



***Gen2J2EE:* Conversion from Gen to J2EE**

July, 2005

2750 Prosperity Avenue, Suite 210
Fairfax, VA 22031

703-246-0000 • 888-EVERWARE
www.everware.com

Introduction

The **Gen2J2EE** initiative is a service offering within Everware's **Gen2J** Program (see **Gen2J** Program Overview document **Gen2J WP v1.4**). Whereas **Gen2J** broadly covers various cohabitation approaches using Computer Associates' (CA) AllFusion Gen (also known as IEF, Composer, COOL:Gen, Advantage Gen and referred to in this document generically as Gen) and Java 2 Enterprise Edition (J2EE), the **Gen2J2EE** Program more specifically addresses the transition of software assets from Gen to J2EE using a Service Oriented Architecture (SOA) approach.

This paper is directed at organizations that plan to move away from Gen entirely, both as a development platform and a runtime platform. It is also useful for those replacing individual Gen applications. The goal of this paper is to present the characteristics of the Gen and J2EE environments that impact transition and recommend strategies for making the transition successful.

Everware

Everware has significant experience with both Gen and J2EE. We are recognized as one of the premier providers of Gen consulting services and have built many of the Gen systems in production today. We were one of the early adopters of Component-Based Development for enterprise applications and have an extensive inventory of pre-built Gen components that we have used successfully to deliver Gen applications more efficiently.

We also are an early adopter of J2EE as an enterprise platform for component-based and service-oriented applications. We have leveraged our Gen experience in application modeling and component architecture to create several unique capabilities that include a library of pre-built J2EE components and a rapid component provisioning tool for the J2EE environment. The combined expertise in Gen and J2EE has enabled us to develop specific techniques and approaches for transitioning Gen to J2EE.

The State of Gen

Gen has enjoyed a long and loyal following because it, quite simply, has been the best full-lifecycle application development tool for enterprise applications. The power of Gen's capability to define and generate large-scale, transaction-based applications to a wide variety of target platforms is well known to its many

users. Over the years and under various names, Gen has been owned and supported by multiple companies (Texas Instruments, Sterling Software and CA). Each vendor has faced the ongoing challenge of trying to keep the Gen product reasonably current within a rapidly evolving technical environment and marketplace.

The basic problem is that while the market is moving towards open standards and the plug-and-play approaches of service-oriented architectures using component-based development, the Gen product and its generated applications remain highly proprietary. It is this proprietary result that makes transitioning Gen applications more challenging than one might initially expect.

Since Gen generates source code in several common languages (COBOL, C++ or Java) the first thought might be to simply take ownership of that source code and either manually maintain it or pass that code into conversion products to transform it into another language or port to another platform. Unfortunately, these are not tenable solutions since the generated code is not generated for human consumption and it relies on a significant amount of proprietary, system level code (runtimes) within the execution environment. In other words, you don't have all the source code and what you have isn't maintainable because it's structured for the convenience of generation not for human understanding and maintenance.

The next tactic might be to try to transition the application specification that is maintained as a model in the Gen encyclopedia. Unfortunately, there are no automated full translation solutions for this as these are also highly proprietary and the model objects are incompatible with the way most translators work. There are some products that can translate Gen data models into ERWin models or Rational Rose models, but this addresses only a small percentage of the translation task.

A more significant factor is application architecture. Most Gen applications are architected as tightly coupled procedural models with end-to-end transaction processes. This is a very different architecture from the service-oriented architectures that most of the industry is moving towards. As a result, many organizations will want to re-architect their Gen applications, making automated translation a less attractive option even if there were tools to accomplish it.

The challenge then is to efficiently extract the business value locked in the Gen models and re-architect that functionality into services that can be used throughout the organization to deploy modern, flexible applications. The **Gen2J2EE** Program is Everware's response to this challenge.

Why Gen to J2EE?

We consider the J2EE platform a logical replacement for Gen from a deployment platform perspective. As a motivation for why J2EE is a sensible replacement for Gen, consider the following:

Gen	J2EE
Typically used for medium to large scale enterprise systems	Designed specifically for enterprise computing needs of medium to large systems
Multiple architecture styles supported, from simple on-line to distributed client/server and web-based architectures	J2EE platform supports a similar set of architectures
Multiple deployment platform options through a proprietary , 'Application Execution Environment' ranging over MVS, Unix, Windows, VMS etc.	Multiple deployment platform options through run-time containers (App-server, Web-server, Browser and Java Virtual Machine), based on an open, industry standard specification, that supports a similar set of deployment environments.
Integrated proprietary IDE for full lifecycle development of applications.	Multiple Vendor IDEs can be selected from as required by the organization. J2EE does not imply a dependency on any single IDE.
Proprietary model repository for management of Gen models and business rules. (Note: Generated and compiled code must still be managed through the organizations preferred version management software).	Models and source code are developed within the organization's IDE of choice and artifacts are managed through the organizations preferred version management software.
Supports model based development and somewhat abstracts developers from language and architecture implementation details (i.e. takes care of the plumbing) (Note: The underlying generation language COBOL, C++,Java is abstracted away, however the developer needs to know Gen's proprietary 4GL known as Action Diagramming).	Model based development is recommended when targeting J2EE. J2EE abstracts developers from the plumbing of enterprise scale distributed applications. An understanding of the J2EE architecture and the Java language is required. IDEs assist in further abstracting away implementation and language details to varying degrees.

A few other points worth considering:

- Gen developers are generally considered software engineers rather than programmers. This is an important concept for Gen developers who make this distinction due to their ability to leverage model-based development and structured, repeatable development processes. As a result, we usually recommend replacing their existing model-based tools with new model-based IDEs and replacing their existing lifecycle model with a new lifecycle model.
- Gen developers leverage abstraction and do not often deal directly with source code. We recommend the use of component-based architectures and advanced IDEs to continue the powerful model of abstraction that Gen developers have become used to. However Java language skills are required for the expression of business rules.

Gen to J2EE - the Architectural challenge

Since most Gen applications are built to support enterprise-level business functions, architectural considerations are a critical element of any transformation. The Gen toolset has somewhat insulated developers from architectural issues since the tool has a set of supported architectures that developed applications are generated into (batch, block mode (3270), client server, proxies etc.). When moving to more open technologies like J2EE, the range of architectural options is significantly increased and the selected architecture has a significant impact on the resulting application in terms of performance (throughput, scalability and reliability), agility (adaptability to change) and maintainability.

Everware recommends adopting proven architectural patterns rather than “reinventing the wheel” for each conversion project. In order to devise the appropriate architecture for an application, an intimate understanding of the architectures employed by the existing Gen applications and a thorough knowledge of the architectural possibilities within J2EE is required.

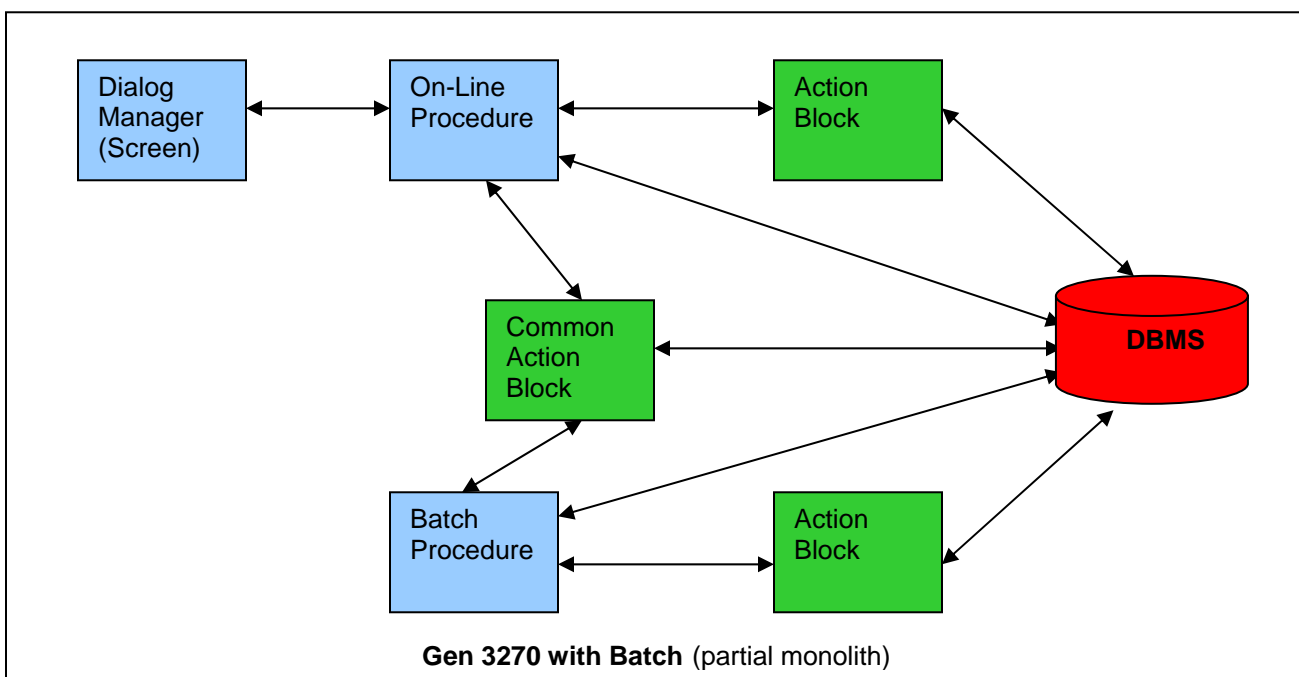
One common architectural pattern that Everware recommends is the Model 2 architecture pattern, also know as MVC (Model, View, Controller). The premise of this pattern is a layering of the application architecture which allows for a clear delineation of responsibility between each of the architecture layers, leverages abstraction within each layer and avoids the confusion resulting from mixing various J2EE technologies into each architecture layer. Within the MVC architecture, we find the following three significant application layers (Note: An application layer does not necessarily equate to a physical deployment tier).

Model: - This layer represents the core business model (both data and process) usually expressed in the form of a suite of available transactions or services within an SOA. The model holds significant properties of the business, namely business objects (data and behavior), significant business rules governing business object interaction and high level business processes. The functionality within the model is exposed via interfaces to the controller layer of the application. These open interfaces, or services, are key to reusing core business functionality across an enterprise.

View: The view layer is responsible both for rendering presentation objects as provided by the controller layer and for providing a mechanism whereby the user may issue events to the controller (e.g. submitting a page of data). The view layer is traditionally some kind of GUI, but it is important to note that the view layer is essentially the outermost interface of your application and as such could be a flat-file, EDI document, XML document, handheld device etc.

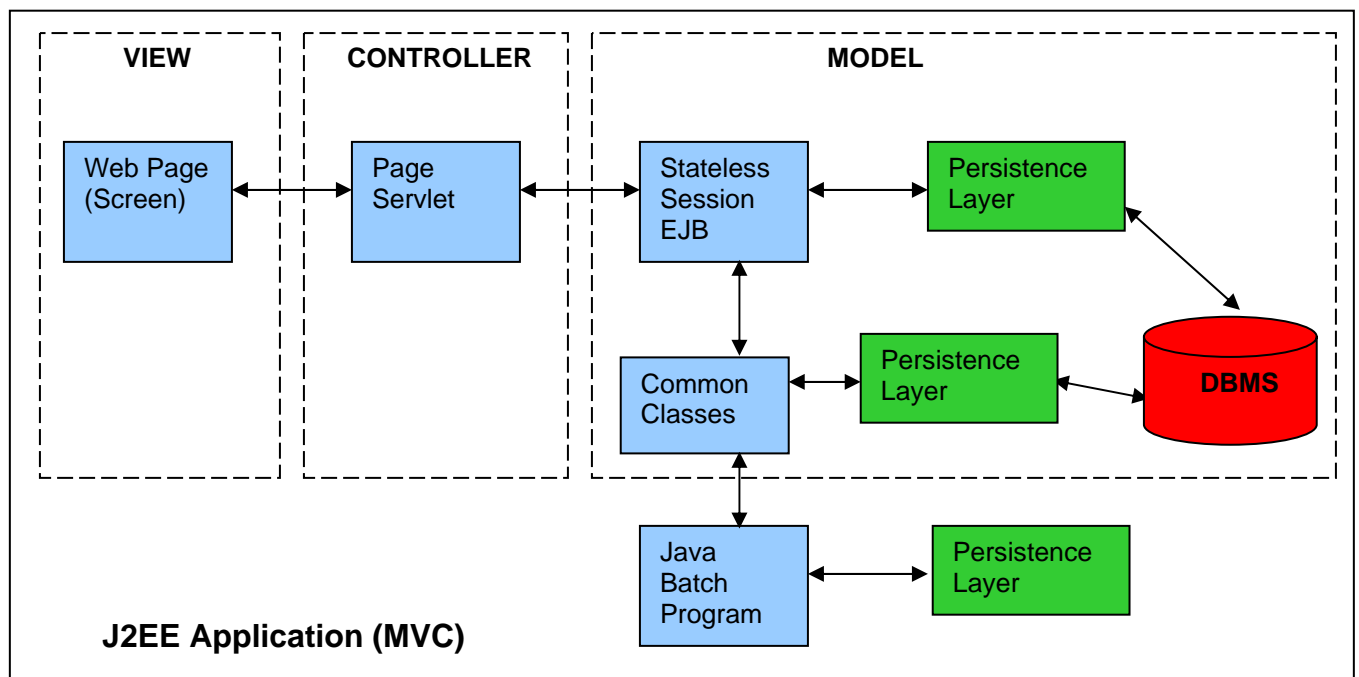
Controller: The controller layer is responsible for intercepting, interpreting and controlling the events that transpire within the application. This layer determines (based on events submitted from both the view and the model layers) what happens next within the application. The controller layer is also responsible for the mapping of model business objects to objects that exist in the view layer. As such, the controller layer embodies a workflow, event handling and object mapping capability. (Note: The controller layer is typically built with a particular application and deployment technology in mind).

As an example of how the Gen application could be re-architected into the J2EE world, consider a possible Gen 3270 block mode application architecture:



The Gen application shown has some factoring of common functionality into common action blocks (CABs) and has some data access occurring via action blocks (ABs). Business process rules, screen event handling and some additional data accesses are executed from the on-line procedure step. We selected a 3270/batch application for this illustration since it generally represents the “worst–case” architecture. Many Gen 3270 applications are total or partial monoliths, meaning that view, controller and model layers are not explicitly separated but are intermingled into a single software layer.

As part of the transition process for Gen applications exhibiting these architectures, Everware conducts a re-factoring effort assigning particular portions of the Gen model to each layer of the to-be application architecture using a service-oriented approach for the model layer elements. The example Gen 3270 architecture (as well as any other Gen architecture) could be converted to the following basic J2EE architecture:



After the transformation, the legacy Gen 3270 application is now an open, modern, J2EE web-application, conforming to an industry standard architectural pattern. The Gen screen, dialog manager and part of the Gen procedure step has been re-factored to form the view (JSP web page) and controller (java servlet), while the remainder of the business process contained in the Gen procedure step has been implemented in a Stateless Session EJB. The common functionality (Gen CABs) has been factored out to common java classes. Data access actions (contained in Gen ABs or in the Gen procedure step) have been implemented using Entity EJBs.

The above discussion illustrates one possible path for converting Gen architectures to J2EE architectures. Other architectural possibilities include Business Process Orchestration, Enterprise Service Bus, and alternate forms of database persistence. The particular path taken on each conversion effort is determined by the existing Gen application architecture style, the desired to-be J2EE architecture and, most importantly, customer requirements.

Gen to J2EE - the Process




The **Gen2J2EE** approach consists of a flexible structure of tasks, techniques, and deliverables that accommodate a wide variety of requirements while clearly defining a roadmap to success. The same four phases, Assessment, Architecture, Implementation, and Deployment, described in the **Gen2J** Program Overview (**Gen2J WP v1.4**) are also applied to the Gen to J2EE transformation.

Once an overall Service-Oriented Architecture and a transition approach is determined, Gen to J2EE conversions are usually conducted in an iterative nature in order to realize the following benefits:

1. Early successes ensure stakeholder buy-in and deliver benefits to the organization faster.
2. Iteration allows smaller, less risky individual projects that can be scheduled according to staff availability, business needs, functional dependencies or other drivers.
3. Effective training and knowledge transfer is ensured through multiple iterations i.e. the Organization's developers practice their newly acquired skills multiple times.
4. Iterations ensure that components provisioned can be leveraged across later projects resulting in asset re-use and associated time, cost and risk reduction.
5. Iteration allows processes, artifacts, standards and lessons learned to evolve and mature in a natural fashion within the organization.
6. Iteration avoids a traumatic big bang approach and allows the organization to more gradually adopt the new paradigms, platforms, processes, tools and software assets.
7. Iteration allows the organization's developers and project management to grow towards conducting these transitions and maintaining the resultant applications with minimal outside support.

To support an iterative development approach, the **Gen2J2EE** Program includes a detailed Work Breakdown Structure (WBS). A sample high level WBS is shown below (support tasks such as Project Management have been left out).

This base WBS is customized as appropriate for each project.

		Task Name
1		<input type="checkbox"/> Gen 2 J2EE - Typical WBS
2		<input type="checkbox"/> Initial Assessment
3		<input type="checkbox"/> Understand Organizational technology objectives
8		<input type="checkbox"/> Analyze Gen Portfolio
12		<input type="checkbox"/> Make recommendations
18		<input type="checkbox"/> Project Iteration n
19		<input type="checkbox"/> Detailed Assessment
20		<input type="checkbox"/> Analyze Application Vision
24		<input type="checkbox"/> Analyze Models
29		<input type="checkbox"/> Analyze Non-Functional requirements
34		<input type="checkbox"/> Analyze Application Dependencies
40		<input type="checkbox"/> Make Detailed Recommendations
47		<input type="checkbox"/> Architecture
48		<input type="checkbox"/> Define Target Architectures
55		<input type="checkbox"/> Architect Notional Components
63		<input type="checkbox"/> Architect Application
70		<input type="checkbox"/> Architect Application Integration Interfaces
77		<input type="checkbox"/> Implementation
78		<input type="checkbox"/> Provision Components
89		<input type="checkbox"/> Assemble Application
94		<input type="checkbox"/> Test Application
101		<input type="checkbox"/> Test Application Integration
106		<input type="checkbox"/> Deployment
107		<input type="checkbox"/> Conversion
119		<input type="checkbox"/> Transition
128		<input type="checkbox"/> Final User Training
135		<input type="checkbox"/> Installation
145		<input type="checkbox"/> On Going Support
155		<input type="checkbox"/> Gen System Decommissioning
168		<input type="checkbox"/> Training
171		<input type="checkbox"/> Mentoring

The Initial Assessment task consists largely of a portfolio analysis to establish a reasonable understanding of the Gen applications to be converted as well as the business and technical objectives driving the conversion. Architecture and technical options are identified during the initial assessment based on the business and technical requirements. The Assessment task is typically followed by an initial iteration (based on the selected options) that includes an appropriate level of analysis and architecture to:

- propose and prove the architecture

- identify and plan the individual iterations
- identify high-level candidate business services
- establish an infrastructure for the full iterations

Each subsequent iteration includes four phases – Assessment, Architecture, Implementation and Deployment. Within the iterations, the Assessment and Architecture phases are focused on driving the initial analysis and architecture work, within the scope of the iteration, to a greater level of detail sufficient to actually implement and deploy. The Implementation phase is where most of the work happens – design and development of software, provisioning of services, hybrid application build (consuming the services), and integration with other applications. If an iteration is deployable – not all are or should be – then Deployment tasks such as installation, conversion and training are performed.

More detailed descriptions of the kind of work performed in the phases are presented below. Since the iteration approach can be viewed as multiple spirals, the work performed for a given phase on a given spiral will vary according to the depth and scope of the spiral, the functionality to be converted and other factors.

- **Assessment** – Since the project objective is a complete replacement of existing Gen functionality by new J2EE functionality, the assessment effort will be focused on the following items:
 - Assessment of the drivers and objectives for the move away from Gen in order to customize a path through the transition roadmap.
 - Inventory of existing applications within Gen and associated artifacts that do not reside in Gen models (EABs etc). Also included are existing services that can be reused to reduce time, cost and risk.
 - Upstream and downstream applications that either feed or are dependent on the Gen applications and the degree of coupling between these various applications.
 - An analysis of the functionality contained within the various Gen applications with an eye to identifying common infrastructural and business services (and corresponding application components) and the degree of affinity between the business objects within the various applications.
 - Specific recommendations regarding the path to be taken for the eventual transition away from Gen to J2EE. This includes recommendations regarding tools, platforms and iterative

processes that will be implemented during the transition. Also included is the approach to be taken for acquiring new skills and knowledge transfer. If necessary, this also includes the selection of a candidate model (or model portion) for a proof-of-concept effort.

- **Architecture** – This effort is focused on architecting replacement solutions that address the business and technical needs of the organization and adhere to the J2EE blueprints and industry best practices.
 - Identify business and technical services required by the business function.
 - Select appropriate architecture patterns for solutions.
 - Identify and model notional components that might provide services and determine their potential for reuse.
 - Model application(s) showing their consumption of services and their interdependencies.
 - Identify and model appropriate solutions for integration between the new application and the associated upstream and downstream applications.
- **Implementation** – This phase is focused on replacing the Gen models and action diagram code with functionally equivalent UML models and java source code and ensuring that the produced application is indeed functionally equivalent.
 - Provision and test re-usable services– We use techniques from years of component based development experience (CBD) in both Gen and J2EE that allow us to rapidly provision common components as identified during assessment and architecture. It should be noted that these components, once provisioned, are accessible to the entire organization through standard java or XML interfaces.
 - Assemble Application – During application assembly the focus is adding organization specific business logic, particularly focused around the preferred presentation mechanism and the specific workflow required by the business process being automated. The resulting application is termed a hybrid application since it is a blend of application

specific software and consumption of provisioned and/or pre-existing services.

- Test assembled application - The assembled application is tested for functional equivalence with the application it replaces. Testing also ensures that the application will meet the required service levels.
- **Deployment** – This includes installing the application in the production environment and associated activities. Replacement of Gen applications in their entirety would require:
 - Integration with related systems
 - Database conversion, as required
 - User training
 - Installation and deployment
 - Performance tuning
 - Ongoing support

For further discussion of the lifecycle phases of a typical **Gen2J** project please refer to the **Gen2J** Program Overview (**Gen2J WP v1.4**).

Gen to J2EE - the Result

Once the Gen to J2EE transition has occurred, the organization is entirely independent of the Gen environment. The business assets that were locked in the proprietary Gen environment are now implemented in the standard J2EE platform using a Service-Oriented Architecture. This allows a much wider choice of development tools, deployment tools, runtime containers, database management systems and hardware and operating system platforms. This flexibility is critical to appropriately scaling and evolving development and operational environments relative to cost, performance and other business requirements. Having implemented a wide range of business and technical services, the organization has a pool of reusable assets that are significantly easier to maintain than the legacy applications and can accelerate the deployment of new capability.

Typically the organization has an equivalent for each item that they previously had in the Gen environment.

Gen	J2EE
Gen Modeling	Organization's preferred UML modeling tool (Rational Rose, Rational XDK, Popkin, TogetherSoft etc.)
Gen Action Diagramming, Window editing, Screen editing	Organization's preferred Java IDE (Sun ONE studio, Borland Jbuilder, Eclipse, Rational Software Architect etc.)
Model repository (Host / Client-Server encyclopedia)	Organization's preferred artifact or version management software (Clearcase, Endeavour, PVCS, CVS etc.)
Gen programs (COBOL /C++)	Java EJBs, classes, JSPs
DBMS Tables	DBMS Tables
Development Process (I.E., Waterfall, Iterative Waterfall, RAD)	Development Process (OOAD, Iterative Spiral, RUP, RAD)
Development Standards	Development Standards
Gen expertise (Modeling, Action Diagramming, Deployment)	OOAD, J2EE and Java language expertise

Summary

The organization looking to transition away from Gen is faced with the very significant challenge of converting enterprise legacy software assets from a proprietary environment. The objective of this transition is to reproduce the existing Gen functionality to the flexible, open, widely supported Java/J2EE environment. The end result is a standards-based Java and J2EE implementation, structured around a Services-Oriented Architecture that is not dependant on any specific vendor's proprietary IDE or runtime components.

There are some commercial tools that assist in the conversion of Gen analysis artifacts into modern UML modeling tools (indeed on large efforts, Everware

exploits these tools). However, to date there are no automated tools to assist in the conversion of Gen code to native 3rd generation languages.

With our **Gen2J** program and specifically the **Gen2J2EE** service offering, Everware assists organizations in rapidly replacing existing Gen functionality through the use of some automated tools and the exploitation of our Gen experience and component based development techniques on the J2EE platform. We fully leverage the service-based architecture inherent in the J2EE specification, which results in a broad set of business and technical services that can then be used to rapidly build new functionality. By following an architected, iterative approach, we can more rapidly deploy functionality to achieve early, steady and incremental ROI.

Other Related Everware Services Include...

J2EE Express and the ***Enterprise Services Capability Development***

For organizations undertaking a large-scale, enterprise-wide migration to J2EE or enterprise component technology, Everware offers two complementary service programs:

- ***J2EE Express*** – a short, focused project intended to exercise the J2EE environment and provide knowledge transfer to in-house staff while demonstrating the rapid delivery potential of J2EE and service-oriented components by building a small, yet usable, application targeted to the business.
- ***Enterprise Services Capability Development*** – a set of services designed for large organizations establishing a J2EE-based component services development and assembly framework. This offering addresses all of the technical, environmental, and organizational factors necessary to achieve the benefits of enterprise services in an SOA environment.