EAP: A DOMAIN-INDEPENDENT, MULTI-TIER PLATFORM FOR B2B DEVELOPMENT

Harry Zhou
Department of Computer and Information Sciences
Towson University
Towson, MD USA
zhou@towson.edu

ABSTRACTION

Common features and functions of B2B implementations have been identified as product opportunities and there exist developed products for exchanges, auctions, reverse auctions, catalogs, etc. However, rarely are these products used standalone. Between these products is "glue code" – software that integrates and tailors the products for individual implementation and client project require ments. It is this software that is continually re-designed and re-implemented for each engagement. Each effort implements unique interfaces to the third-party products that it incorporates. This paper describes the architecture of a domain-independent, multi-tire E-market Application Platform in details along with the lessons leaned and benefits observed. EAP can reduce time to market by minimizing both the initial development time and the cost of ownership by amortizing maintenance and evolution costs across multiple digital marketplace engagements. The EAP architecture extends the standard multi-tier enterprise architecture by allowing the integration of discreet layers of program abstraction through well-defined interfaces between the tiers.

INTRODUCTION

An E-market Application Platform (EAP) can reduce time to market by minimizing both the initial development time and the cost of ownership by amortizing maintenance and evolution costs across multiple digital marketplace engagements.

Common features and functions of B2B implementations have been identified as product opportunities and there exist developed products for exchanges, auctions, reverse auctions, catalogs, etc. However, rarely are these products used standalone. Between these products is "glue code" – software that integrates and tailors the products for individual implementation and client project require ments. It is this software that is continually re-designed and re-implemented for each engagement. One reason is that each imple mentation is treated as a unique and isolated product in its own right. Each effort implements unique interfaces to the third-party products that it incorporates. In the extreme, not only does this result in custom designs and code for each project, but also each effort potentially integrates the same products differently. The major observation is that continual use of this approach increases the risk associated with the factors identified a bove.

There is significant advantage that accrues to vendors and Professional Services organizations that take a more planned development approach for their clients. Better project management and process improvement are one level of a planned approach. Another is adoption of a project technical development methodology that addresses underlying causes in key leverage areas. Upstream design activities are one such key area. A second is flexible software. A third is leveraging already developed and tested soft ware components.

The software industry has long realized the state of chaos in software development. Component software, i.e., componentware, is one example consistently identified by many experts as having a profound effect upon the economics and time-to-market for software products. Component-based development is viewed by many as the new paradigm in software development. Further, packaging componentware in a software development framework enables many concrete benefits, and the payoff of such an approach is potentially huge

The benefits of developing e-business applications using such a framework are:

- **Componentware:** A fundamental development effort undertaken in most B2B digital market engagements integrates many e-commerce third-party software products. EAP pre-packages framework-level APIs for these standard components. Furthermore, specific third-party software products may already be supported as a part of the framework. In this way, EAP not only is a digital marketplace integration platform, it is a populated framework with already designed, implemented, and, in some cases, integrated components. For Professional Services projects that can adopt this approach, significant time-to-market advantage can be derived at lower cost and schedule risk. Finally, as Professional Services adds to the componentware, we are developing a repository of proven, tested, and quality integration components.
- **Upstream design:** Component framework software solutions maximize the impact of upstream design. Those components, wrappers, and glue code form larger reusable components. They are puzzle pieces that have already been put together, reliably and with high quality. By reusing the framework components in further implementations, significant leverage in time, cost, and

quality will accrue to the client.

- **Flexibility:** Many of the products on the market have rich sets of functionalities that often are not used. In some cases, extra functionality is needed to meet client requirements. For example, a client may require strong security management system, such as Netegritity. With this approach, clients are not forced to accept a set of pre-specified products that do not meet their needs, but are able to pick and choose whichever they require. EAP offers such flexibility.
- **Leveraging existing assets:** Software development from scratch for each client is costly and risky. Professional Services organizations learn through project implementation via the experience gained by individuals participating in the projects. Adopting and extending the EAP framework through a series of project implementations provides the quality of design and implementation to the next development projects. This componentware framework embodies, over time, more and more of the practical implementations lessons learned in digital market implementation.

E-MARKET APPLICATION PLATFORM

EAP is a programming framework that supports the rapid development and deployment of collaborative digital market services. EAP provides a platform for integrating third-party software. It is based on a modern and proven multi-tier enterprise application model.

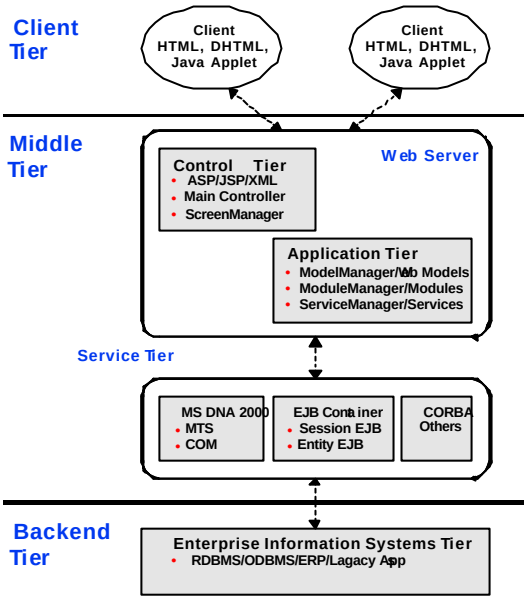
EAP is a high-performance, robust, and scalable digital market platform. EAP is built on open industry standards such as: J2EE (Java 2 Enterprise Edition) Platform – a multi-tier and component-based middleware architecture. This architecture is a fundamental requirement for building high-performance and scalable applications. Furthermore, EAP provides many features to facilitate the integration of digital market service components.

EAP is a programming framework. It provides a clean separation between an application's control logic, integration logic, business logic, and the enterprise information systems on which it relies. APIs and integration among these components are well defined, and, in some cases, prepackaged. As a result, programming specialists in different implementation areas; e.g., control, integration, and business application logic – can work in their specific area with minimum interactions from other areas, thus greatly increasing programming productivity.

EAP can clearly define and divide the tasks among various developers with different skill sets. A web designer created web pages using off-the-shelf third-party web design tool. Those web pages were converted into JSP for dynamic data control by Java programmers with little knowledge of EAP. Developers with some training on EAP implement the integration glue between the control and business application logic. Developers with other backgrounds, such as EJB, can create required, additional service components. With EAP, different pieces can be developed separately and integrated seamlessly. In what follows, the architecture of EAP is described.

EAP ARCHITECTURE

The EAP architecture extends the standard mu lti-tier enterprise architecture by allowing the integration of discreet layers of program abstraction through well-defined interfaces between the tiers:



The Client Tier

The EAP Client Tier consists of web-compliant user interface (UI) devices.

The Middle Tier

The Middle Tier encompasses both user interface controls and business logic. TEAP breaks the middle tier into three parts:

- **Control Tier** contains the user interface logic and defines the look and feel of an application.
- **Application Tier** defines the business logic for an application and modules for interacting with standard services.
- **Service Tier** provides mechanisms to connect to net-sourced services or resources in the EIS tier.

The Control Tier

The control tier resides in the web container and creates the view. It captures the control logic (look-and-feel) of an application. The view can be constructed using modern web tier implementation technologies such as ASP, JSP, Java servlets, and XML/ XSL. Web programmers and graphic designers can use these technologies to develop the user interface independently from the business and integration logic needed to drive the application.

Application Tier

An application's business logic is abstracted in the application tier. The application tier resides in the web container and is used directly by the control tier. Within the application tier, *model web implementations* capture the business data used by the view, *modules* integrate basic services that address the needs of a particular marketplace, and *service stubs* provide access to underlying application services. The implementations of these services can reside locally or remotely.

A module captures the integration logic that defines how basic services can interact to perform sophisticated tasks needed by a particular application. The business logic encapsulated by a module is very specialized and is usually simple and limited in scope. Unlike services, the integration logic encapsulated by a module might vary from marketplace to marketplace.

Service Tier

The service tier contains components that encapsulate the business logic of applications and services, including third-party services. Services can be implemented using a variety of component technologies, such as COM/DCOM, CORBA, or Enterprise Java Beans (EJB), and can be accessed through XML messaging standards such as BizTalk™or ebXML.

The service tier uses connector technology for interfacing with components provided by service providers. The connector technology masks the complexities of particular service components, such as specific implementation technologies, platforms, and API differences. The service tier also provides basic management and configuration utilities for registering and configuring service components.

The EIS Tier

The EIS Tier contains existing enterprise systems such as databases and legacy applications that can be accessed through industry-standard APIs, such as ODBC, JDBC, CORBA, COM/DCOM, SOAP, Java RMI, JNDI, and JMS. EAP also provides sophisticated, scalable support for XML.

REQUEST PROCESSING

The EMarket engine is at the core of EAP. Within the EMarket engine, a WebController processes all incoming HTTP requests and delegates to the model web implementations, modules, and services as necessary to process the requests. The WebController locates the appropriate objects through a set of manager objects: a ModelManager that manages model web implementations, a ModuleManager that manages modules, and a ServiceManager that manages the available services.

Request Processing Flow

When an HTTP request is received:

1. The WebController is called to process the request.
2. The WebController calls
    a. A RequestToEventTranslator to convert the request into an event.
    b. A Module to process this Event.
    c. The ModelManager to update all relevant models.
3. Based on the type of the request, a RequestToEventTranslator is returned by the TranslatorManager. The request is then translated into an EMarketEvent object.
4. The translation decouples the MVC Model from the web presentation. (This enables you to easily change the web presentations without modifying the model. You can even support non-web presentations without modifying the model.)
5. Based on the type of the event, a module is returned by the ModuleManager to process the event. The Module knows which services it needs to process the given event.
6. The event is sent to the designated services via a ServiceStub. Services are managed by the ServiceManager, which can return a service stub by name. Depending the service scope, the ServiceManager might return the same service for the whole application (application scope), or a new service for each session (session scope, which is the default).
7. A service stub delegates the event processing to its real service, which can reside locally or remotely. Each service will return a list of models that it has changed during event processing.
8. The ModelManger notifies all of the interested model web implementations about the changed models. Each model web implementation, in turn, will perform anupdate to reflect the latest changes.
9. An HTTP response that contains the updated content from the data view (ModelWebImpl) is generated and sent back to the user.

Interoperability

EAP supports interoperability in three ways:

- Components at different tiers can interoperate

seamlessly. A component in one tier can be replaced without affecting the rest of the system as long as the replacement supports same functionalities.

- Dynamically configured services can interoperate using a variety of protocols, such as XML, COM/DCOM, CORBA/IIOP, SOAP, RMI, and JMS.
- EMarket engines on different platforms can interoperate through services that they share.

Service Configuration and Management

EAP supports dynamic configuration of service components. The framework also provides a runtime environment for component lifecycle management and service lookup. Request events are directed to the proper services for processing based on the service configuration. When a service is required, the EMarket Engine's Service Manager returns a service stub for the requested service. If the service does not exist or is session-based, the Manager creates a new instance of the service.

Security Management

EAP is designed to ensure that resources are only accessed by authorized users and communications between services are secure. The framework supports:

- Authentication: Users must be authenticated with correct user names and passwords.
- Authorization: Users must have the necessary privileges to be granted access to a particular resource.
- Encryption: If required, data transmitted to remote services can be encrypted.

Internationalization Support

EAP facilitates the internationalization and localization of marketplace solutions. The framework provides a global resource bundle manager to simplify the i18n process for eMarket components. The eMarket Engine and selected eMarket components are already internationalized using standard i18n techniques.

EMarket Component Integration

EMarket components provide complete, reusable solutions that implement common e-commerce features, such personalized user interfaces, catalogs, and shopping carts. These solutions can be seamlessly integrated with the eMarket engine as needed for a particular solution, enabling sophisticated e-commerce solutions to be designed and deployed very quickly.

To build a component (such as a catalog or RFP/RFQ service) that conforms to EAP, you:

1. Build or buy a service component from a service provider.
2. Build a connector to the service component. For services that use a common component framework, the framework provides tools for automatically generating the connector layer.
3. Develop a module that customizes the service to the specific deployment. In some cases, the module might connect to multiple services at the same time.
4. Develop the component data model that will be used by the presentation tier. Make sure that the model can synchronize correctly with the service data model.
5. Design and develop the presentation for the component.
6. Develop an event translator that translates http-based requests to application events.

Before it can be used, a service component has to be configured with the eMarket Engine. The eMarket Engine needs the following information about the component:

- The screens and templates used in the presentation.
- The URLs, the mapping between the URLs and screens (screen flows), and the event translators they use.
- The services, their scope (session or application), and the events that they can process.
-

To plug a new component into the system, you need to:

1. Configure the presentation tier by filling information about screen flows, request processing, etc.
2. Configure the application tier by providing information about event processing and data model registration.
3. Configure the service tier by registering and configuring the services

THE FEATURES OF EAP

The features of EAP are summarized as follows:

- EAP provides a clean separation of a solution's user interface (UI) control from its underlying business logic. It enables its UI to be designed independently and updated quickly whenever necessary. Similarly, separating a solution's business logic from the complexities of interacting with enterprise services enables a solution to focus on the business problems to be solved and reuse standard mechanisms for accessing enterprise services. As a result, EAP significantly improves component reusability, thus reducing the development time and risk.
- EAP helps the division of development work among people with different skills so that each module can be

developed independently. Horizontally, development work can be divided based on service components; e.g., shopping cart, catalog, order management, etc. Vertically, development work can be divided based on tiers; e.g., UI for control tier, EJB/COM+ for service tier, and EAP Integration for application tier. EAP supports a project development framework and timeline that accelerates development velocity.

- EAP frees developers from integration processes as much as possible by providing screen flow management (what to display next), request processing configuration (which service should process a given request), front-end and backend data synchronization (ensures the front-end gets up-to-date information), and other services. Integrators don't have to hard-code these processes for every application. Instead, how they should be done can be described declaratively. As a result, it saves integration time and reduces coding mistakes.
- EAP provides a structured methodology for integration. As a framework, EAP provides integration building blocks and an integration design. As a result, this design reuse greatly reduces the maintenance cost in the long run because all solution integration efforts are done in a similar fashion; even different integrators can implement framework structured solutions reliably.
- EAP interoperates seamlessly with other server-side component-based enterprise application framework technologies such as J2EE or MS DNA. As a result, an application developed using EAP is not tied to a specific server-side technology. This gives more choices to both clients and developers by allowing them to immediately leverage many existing technologies and assets. For future development, EAP supports rapid evolution to new technology as it becomes available so clients are not restricted to the technology in their initial solutions.

CONCLUSION

EAP is a framework for the rapid development and deployment of dynamic, high-performance e-marketplaces that are secure, reliable, and scalable. EAP communication and service interoperability technology enables heterogeneous services to be used together seamlessly. Clearly defined interoperability requirements ease the integration of new components with existing solutions.

At the infrastructure level, EAP will offer sophisticated brokering and matchmaking components to provide dynamic service and process registration, composition, and coordination. Integration with ERP or legacy systems will also be provided through messaging and connection adaptors such as BizTalk™Server and WebMethods. In short, it is hoped that EAP will provide a technical foundation for enabling the next-generation phase of B2B e-commerce.