DEFINING THE OPENEDGE® REFERENCE ARCHITECTURE - BUSINESS COMPONENTS: BUSINESS TASK

Mike Ormerod
Applied Technology - June 07

PROGRESS SOFTWARE

CONTENTS

DISCLAIMER

INTRODUCTION

This paper is one of a series of papers that define specific aspects or components of the OpenEdge Reference Architecture (OERA). This paper defines the Business Task, discusses why Business Tasks are a valuable concept, its dependencies and some of the key design decisions and consequences to consider.

BUSINESS TASK: DEFINITION

The Business Task is a member of the Business Component layer and is the component of an application that manages a service request that spans more than one Business Entity or Service Providers. The Business Task must be able to complete or error in a single process/transaction; i.e., it does not need to persist content information across a multi-step process for an indefinite period of time.

VALUE

The Business Task serves a valuable role in well-designed application architecture. It provides a number of key benefits, including:

- Loose Coupling of Business Entities

  The Business Task allows the orchestration of multiple Business Entities to complete a service request. Moving this level of control to the Business Task as opposed to keeping it within a Business Entity allows greater flexibility in determining what Business Entities to use, and in which order, as opposed to the task being encoded within individual Business Entities. This loose coupling also removes the potential to create accidental call architecture within the Business Component layer, where any number of Business Entities end up with a dependency on any number of other Business Entities.

- Encapsulation and Re-Use

  Keeping the definition of a process at a Business Task level allows a greater encapsulation and re-use of Business Entities. Due to this level of isolation, the same Business Entity can be used in multiple Business Tasks as required. For those development houses that have multiple applications, this re-use extends across applications. For example, if a you supply multiple applications that have the need for a Customer entity, keeping the Customer Business Entities isolated from other entities that would be specific to each application, such as an Order entity within a Manufacturing System, and a Support Ticket in a Call Logging System, allows the same Customer entity to be used in both the manufacturing and call system as there is no inbuilt dependency to the

Order entity for the manufacturing system and the Support Tickets entity for the call logging system.

- Improved Application Maintainability

  The Business Task improves the maintainability of the application by organizing the definition of the task into a single point. Therefore, it is possible to change the definition of a Business Task without having to change the underlying Business Entities used to complete the task.

DEPENDENCIES

The Business Task has dependencies within the Business Component layer and with other layers of the architecture. *Figure 1* shows the components within the Business Component layer.

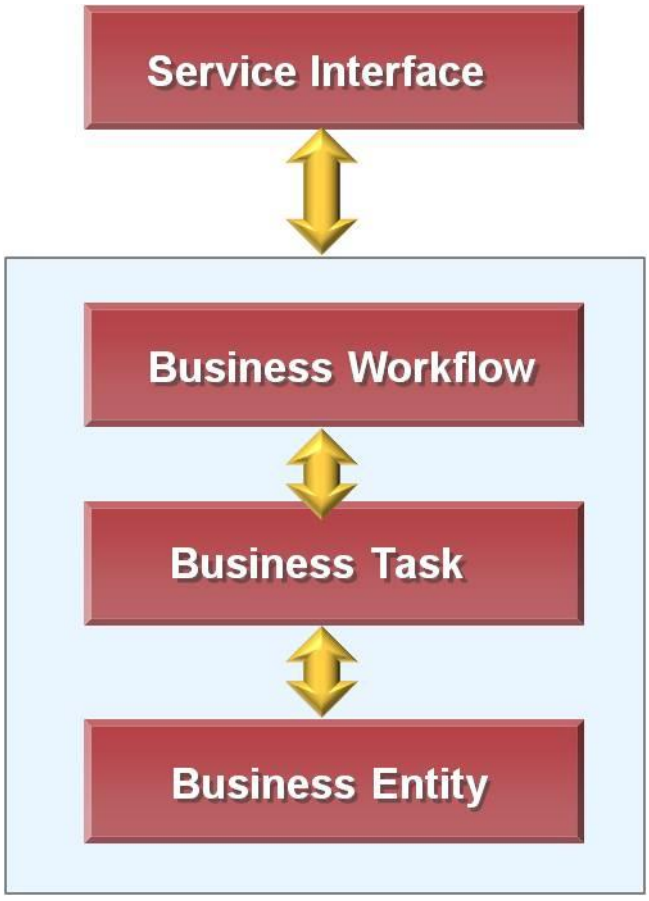

Figure 1: Business Task within the Business Component layer. The grouping highlights the fact that a Service Interface can be for any of the other components.

Figure 1 can be more formally represented as a model shown in *Figure 2* below, showing not only the components but detailing the relationship between them.

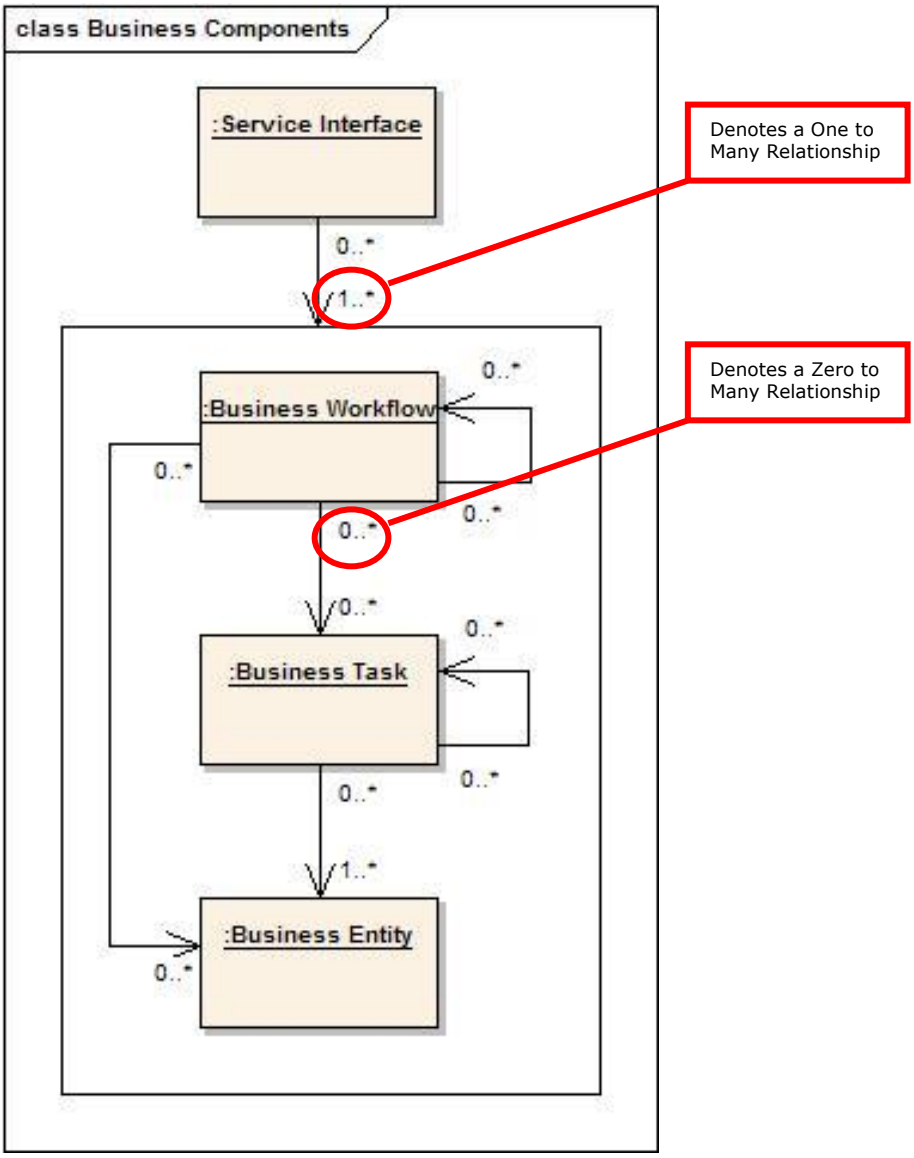

Figure 2: Business Task Dependencies within the Business Component Layer

As shown in *Figure 2,* within the Business Component layer the Business Task can be utilized within Business Workflow or another Business Task as part of a larger process. In these scenarios, the Business Task will expose its methods to another Business Task or Business Workflow component. In order to expose its methods publically, i.e. to a Service Requester, the Business Task is dependent upon a Service Interface, and it is only through a Service Interface that a Service Requester or Client can access the functionality of the Business Task.

Internally, the Business Task has a dependency on one or more (1..*) Business Entities in order to complete its request. (For more information on these dependencies, see the Business Component layer paper.)

Outside of the Business Component layer, the Business Task has two key dependencies, as illustrated below:

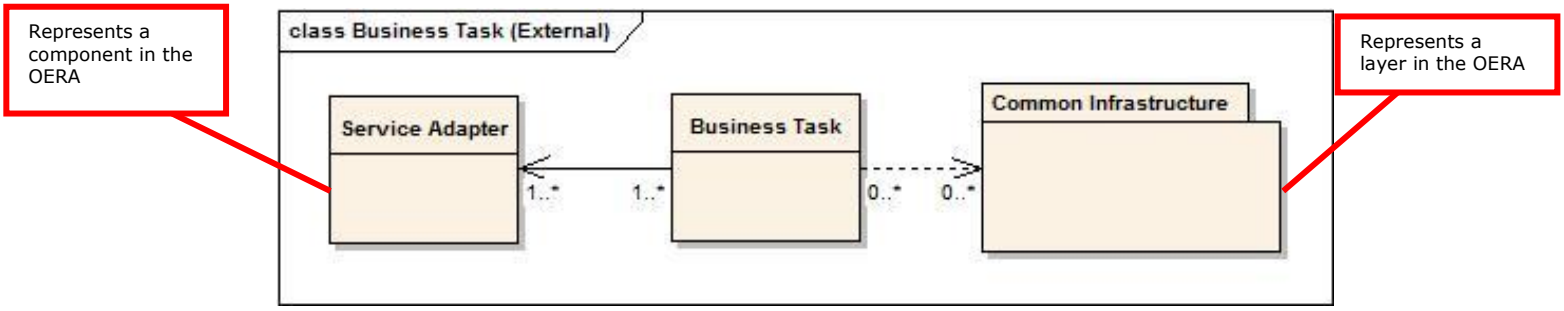

Figure 3: Business Task Dependencies outside of the Business Component Layer

The first external dependency is the Common Infrastructure. The Common Infrastructure layer provides services such as Context Management and Security Management to the Business Task.

The second external dependency is the Service Adapter. This dependency is to cater for those services within the Business Task that need to make outbound service requests, which in effect turn the Business Task into a Service Requester. Given the internal relationships just discussed, it could be feasible that Business Task purely orchestrates services from other Service Providers.

DESIGN & CONSEQUENCE

As *Figure-2* shows, the Business Task is a key component of the Business Component layer.

The design also includes the following other key decisions:

- **A Business Task can access another Business Task**. This allows a Business Task to be comprised of other Business Tasks. However, note that as per the definition earlier, no matter how many Business Tasks a task is comprised of, it must be able to complete or error in a single process/transaction.

- **A Business Task can have one or more Service Interfaces**. This design decision allows for the fact that different Service Requesters can have different data and contract requirements. So, in order to provide a single service definition at a Business Task level, the design allows for multiple Service Interfaces. As a result, an OpenEdge ABL client can use one Service Interface, while an application communicating via Web Services can use a different Service Interface, and each will perform the necessary steps to normalize the incoming request to a single method within the Business Task.

- **A Business Task comprises of one or more Business Entities**. This design decision (see *Figure-2*) is to cater for situations where more than one Business Entity will be required to complete a request. This decision removes the possibility of creating an internal accidental call architecture where Business Entities have built in dependencies between them, which would greatly reduce their potential for re-use, and instead raises the orchestration of Business Entity to Business Entity communication to this higher level. This also means

that a Business Task (or Business Workflow) must be used if more than a single Business Entity is required to complete a request.

- **A Business Task is stateless**. The Business Task receives data in a Data Instance(s) from its corresponding Business Entities in response to an incoming request from a Service Interface, or Business Workflow or as part of the incoming request.

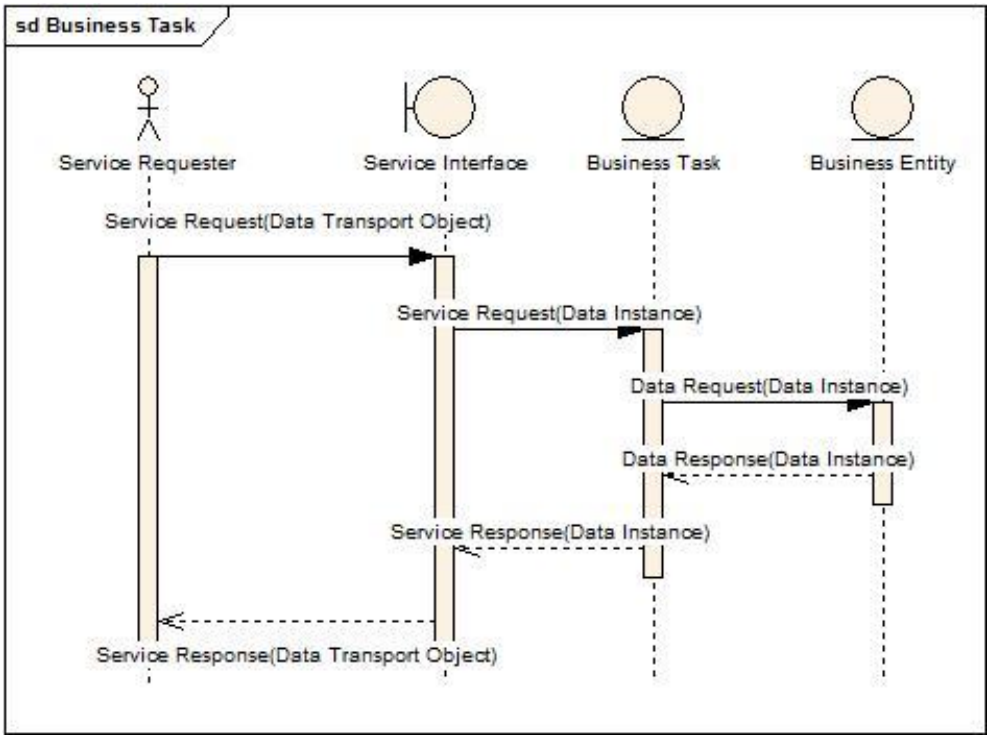

Figure 4: Example model showing sequence of a service request through a Service Interface to a Business Task, which in turns calls a Business Entity

The role of the Business Task is to apply whatever business process logic is required and then pass the Data Instance(s) back to the Service Interface or whichever component made the request of it (see *Figure-4*). As a result, the Business Task is a stateless component, in that it does not maintain any data or state information in between service requests; i.e., it does not "own" the data.

- **A Business Task is a singleton**. Following on from a Business Task being stateless, and since it has been established that it doesn't maintain any data or state in between requests, it would seem a logical conclusion that a Business Task is a singleton, in that one, and only one, instance is active in a session. The result is a performance benefit due to the fact that a Business Task can be started through a Service Manager, and then cached for the life of a session and simply re-used, as opposed to being started and stopped for each incoming request. However, this does mean that one, and only one, instance of the Business Task is running per session, and in some applications this may not be a desirable situation due to existing design decisions.

- **A Business Task can be a Service Requester**. In certain situations, a Business Task may have the requirement to make a call to another Service Provider in order to complete its service request; for example, credit checks. In order to maintain encapsulation and isolate the Business Task from the implementation of the external business application, a Business Task can utilize one or more (1..*) Service Adapters (see *Figure-3*). The consequence of using a Service Adapter is that the Business Task is isolated from the discovery and call to the external business application or service. This allows the potential for the external service to be changed without an impact on the Business Task.

- **A Business Task starts a transaction**. In order to maintain transaction consistency, a characteristic of a Business Task is that it starts a transaction. A consequence of starting the transaction at this level, even though the Business Task deals only with Data Instances and therefore only a logical data model, is that it caters for the situation above where the Business Task becomes a Service Requester. If the transaction were started at a lower level (in the Data Access layer, for example), there would be no knowledge of external service calls, and therefore no mechanism for handling transaction consistency. Since a Business Task is stateless and starts a transaction, a transaction started by a Business Task must complete for a given request. If a transaction cannot be completed in a single request, then a Business Workflow must be used. For Business Tasks that utilize Business Entities, the transaction started within the Business Entity will become a sub-transaction.

  Within the OERA there are two forms of transaction, Managed & Un-Managed. A Managed Transaction is managed by the transaction manager built into the OpenEdge platform. A Managed Transaction can only access Managed Data Sources. A Managed Data Source is a data source that is natively understood by OpenEdge, be that the OpenEdge Database or a foreign data source for which there is an OpenEdge DataServer product.

  An Un-Managed Transaction is a transaction that updates information in an Un-Managed Data Source or Enterprise Service. An Un-Managed Transaction is not managed by the transaction manager of the OpenEdge platform. An Un-Managed Data Source is any data source which is not the OpenEdge Database or a foreign data source for which there is an OpenEdge DataServer product, such as a XML file or a flat text file.

  This is an important distinction. If within an application a Business Task only ever deals with Managed Data Sources, then you are able to take advantage of the built in transaction manager of the OpenEdge platform to handle such issues as roll-back. If however, the Business Task is involved in utilizing Un-Managed Data Sources, then there is a responsibility to provide some form of compensation to perform roll-back. Even though it may not be the Business Task that performs the actual compensation, (that will be performed in the Data Access layer), it is the responsibility of the Business Task to have API(s) to expose through a Service Interface that can be accessed by the Service Requester to initiate the compensating transaction.

SUMMARY

This paper has provided a definition of a Business Task within the OpenEdge Reference Architecture. It has highlighted its dependencies with other components of the architecture and discussed some of the design decisions and their consequences. For definitions of other components within the OERA go to PSDN http://www.psdn.com/library/kbcategory.jspa?categoryID=54.

Corporate Headquarters
Progress Software Corporation, 14 Oak Park Drive, Bedford, MA 01730 USA
Tel: 781 280 4000 Fax: 781 280 4095

Europe/Middle East/Africa Headquarters
Progress Software Europe B.V. Schorpioenstraat 67 3067 GG Rotterdam,
The Netherlands
Tel: 31 10 286 5700 Fax: 31 10 286 5777

Latin American Headquarters
Progress Software Corporation, 2255 Glades Road, One Boca Place, Suite 300 E,
Boca Raton, FL 33431 USA
Tel: 561 998 2244 Fax: 561 998 1573

Asia/Pacific Headquarters
Progress Software Pty. Ltd., 1911 Malvern Road, Malvern East, 3145, Australia
Tel: 61 39 885 0544 Fax: 61 39 885 9473

PROGRESS SOFTWARE

www.progress.com