Business Object State Transition Controller

RUSS RUBIS, FLORIDA ATLANTIC UNIVERSITY

DR IONUT CARDEI, FLORIDA ATLANTIC UNIVERSITY

ABSTRACT

Businesses constantly consume, generate and share data. Without data, a typical business process would come to a halt. But for the data to be useful to businesses, it must be accessible, manageable, and current. The data supporting daily business operations does not just materialize out of thin air. There exists a number of patterns dealing with data object presentation, manipulation and storage, among them are the Model-View-Controller (MVC), the Document-View-Presentation (DVP), and the Presentation-Abstraction-Control (PAC) patterns, just to name a few. However, the life cycle of a business object is more complex than just its presentation and persistence. In a business environment every business object throughout its life cycle must adhere to certain rules, regulations, approvals, validations, and constraints. The pattern of managing business object life cycle is the subject of this paper.

KEYWORDS

Business object, data object, business document, workflow, document status, common business object, dynamic object, business object life cycle, managing business object life cycle, transition states, controllers, transition state controllers

---

1. INTRODUCTION AND OVERVIEW

Business data is arguably the blood line of a typical business. Data is vital to just about every business process, internal or external, and the data management is just as important as the data itself. There are a number of forces which mold the life cycle management of the businesses data. There are economic forces, which allow businesses to stay competitive by the way they use and maintain their data. There are legal forces, which bind businesses to certain laws. There are privacy and accounting practices businesses must adhere to. Finally, there's a growing social pressure on the ethics of usage and dissemination of personal data by government agencies and business entities. In this paper we introduce the business object life cycle management pattern, our attempt to address and abstract the process of data acquisition, creation and update.

2. THE BUSINESS OBJECT LIFE CYCLE MANAGEMENT PATTERN

2.1 Intent

Ensure the business objects' integrity and availability throughout their life cycle by establishing control gates around business objects' transition states, such that checks, validations, and verifications are possible.

2.2 Example

Within the business environment, every business object that is being created (or updated) must pass certain scrutiny before it becomes available for general consumption. Consider, for example, a process of adding a new vendor to the system. It is not just a matter of entering the new vendor information in the vendor database. After the new vendor data has been entered in the system, it must first pass certain approvals and validations before it can be made available for consumption by other processes. The new vendor must first be approved by the buying organization of the business, who signs off on such things as who asked for the new vendor, why is the new vendor being added to the system. Then there is legal department approval and validation which must take place before the new vendor can be made available system-wide. Are there any obstacles to doing business with the new vendor? Has the background check been completed? Has the vendor's tax payer ID been verified? There could be, and often are, other approvals, validations and verifications which must take place before the new vendor business object can be made available for usage. Only after the new vendor business object has gone through and completed all the required checks, can it be then made available for consumption withing the business system.

2.3 Context

Often changes to business objects could have legal as well as economic consequences, and must therefore follow strict corporate and often government procedures and restrictions [1]. Most, if not all, business objects follow a certain life cycle, and are frequently modified, shared, deleted or copied.

You are designing a new business object which will be incorporated into your company's system. It is expected that the new business object will be accessed by the current applications and processes within the system. However, when the new business object is first created (or subsequently updated) within the system, it must pass some predefined approvals, validations, verifications and crosschecks before it becomes available for usage by the rest of the system.

2.4 Problem

As a business object transitions between states during its life cycle, how do you control changes such that its integrity and availability is ensured throughout the life cycle?

2.5 Forces

- All business objects should follow a certain life cycle process throughout which they are created, updated and shared.
- Business object cannot and should not be created, updated or shared without an underlying process which guards its integrity.
- Managing the life cycle of business objects is often ignored or not followed, resulting in inconsistent data at best, and missing or incorrect data at worst.

2.6 Solution

The Business Object Life Cycle Pattern describes an extensible approach for managing the life cycle of a business object and ensuring its integrity.

When a business object is created within a system, it must pass certain scrutiny before it can be considered a valid and complete business object within that system. During its life cycle a business object may go through various transition states which determine its availability within the given system. In this paper we propose that at a minimum a business object must to go through three stages in its life cycle. The three basic stages are:

1. Create/Update Stage. This is the initial stage, during which the business object is created, if its a new business object, or updated if it an existing one.
2. Workflow Stage. During this stage the business object undergoes vigorous validations, verifications, and crosschecks.
3. Ready Stage. Once in this stage, the business object is ready for consumption.

Naturally there could be, and often are, more stages that a business object must go through in its life cycle. The additional stages are often due to industry specific requirements, application constraints, or business process demands. But the additional transition stages are the extension of the three original stages proposed in this paper: Create/Update, Workflow, Ready.

For example, a purchase requisition might go through the following stages in its life cycle:

Created → Submitted For Approval → Approved → Converted to Purchase Order and Sent to Vendor

Although the transition states are specific to a purchasing requisition, they are in a way an extension of the original three stages. The Created and Submitted For Approval corresponds to the Create/Update Stage of the business object life cycle. During this stage the purchase request is created and the items that need to be ordered are added to the purchase request. The Approved corresponds to the Workflow Stage. During this stage the purchase request might go through the approval process, as well as validation and crosscheck, such funds availability and verification, order amount limit, etc. Finally, once purchase request has been fully approved, it is in Ready Stage, at which point it is "ready" to be converted into a purchase order and sent to the vendor for fulfillment.

Creation/Update Stage

During this stage the business object is either created anew (for example when we create a new vendor), or is updated as the result of some data changes to the given business object. To reference above example, a new vendor business object must first be created and populated with vendor specific data. An existing vendor object can be updated during this stage also. While in Create/Update stage, the business object is not available for consumption by other entities or processes. For example, a business object representing a new vendor ABC which is in the process of being created is not accessible by other business object, such as purchase requisition or ordering process, because vendor ABC is not yet "ready" to be consumed. It is important to note that if the business object is being updated, the update is performed on a copy of the original, and not on the original itself. This way the original business object is still available in the system, and only its copy being updated. Once the copy is fully updated and enters the Ready stages, it replaces the original. Thus the state transition controller is also a business object version controller.

Workflow Stage

Once the Create/Update stage has been completed, the business object enters the Workflow Stage. During this stage the workflow process takes over the state of the business object. As per above examples, a newly created vendor must pass vigorous validations, verifications and crosschecks before it can be available for consumption in the system. Note that within the system, the workflow process can be either manual or automated process. The important thing to understand is that it is the workflow process that contains the logic (or rules) which must be applied to the business object. The business object itself does not (and should not) contains any logic (or rules) on how it should be processed. The business object simply contains a set of related data which together comprise a specific business object (i.e. vendor, purchase request, etc.).

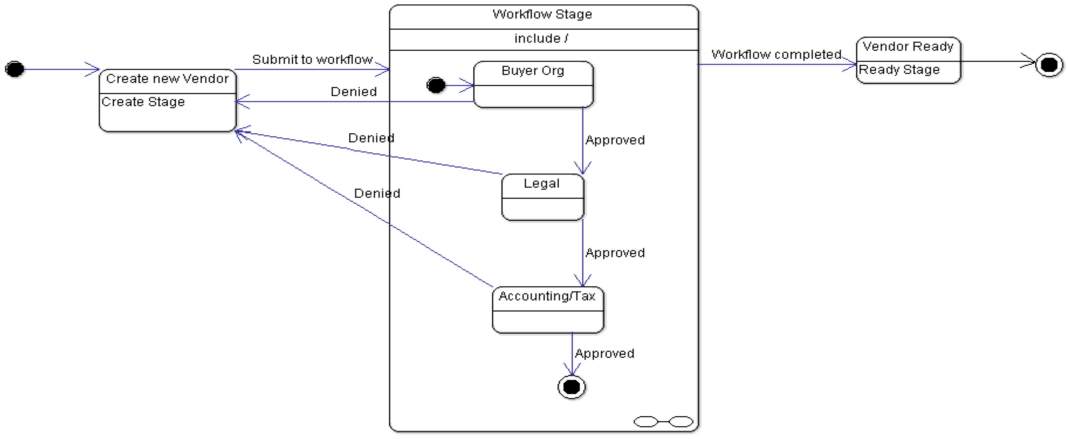
It should be noted that not all business objects must pass a complicated workflow stage. For example, a change to vendor contact data should not require high level approval or rigorous validation and verification. Thus a change to vendor's contact information would require an update (Create/Update stage), some sort of simple approval (Workflow stage), and finally the change available system-wide (Ready stage).

Of the three stages, the Workflow is the only optional stage. Frequently business objects need to be created in the system which do not require any validation or verification. An example of such business object would be a new customer. New customers need to be created quickly and frequently, and often do not required any validation or verification. Thus a new customer business object would only need Create/Update and Ready stages.

It is important to note that although a new customer business object does not require validation/verification, it should still go through "an empty" Workflow stage. Thus if in the future a need arises to add some business rules to a new customer business object, they can readily be added to the existing workflow.

Ready Stage

Once the logic (or rules) of the Workflow Stage have been completed, the business object enters the Ready stage. At this stage no further updates to the business object are allowed. From examples above, once the all the approvals,

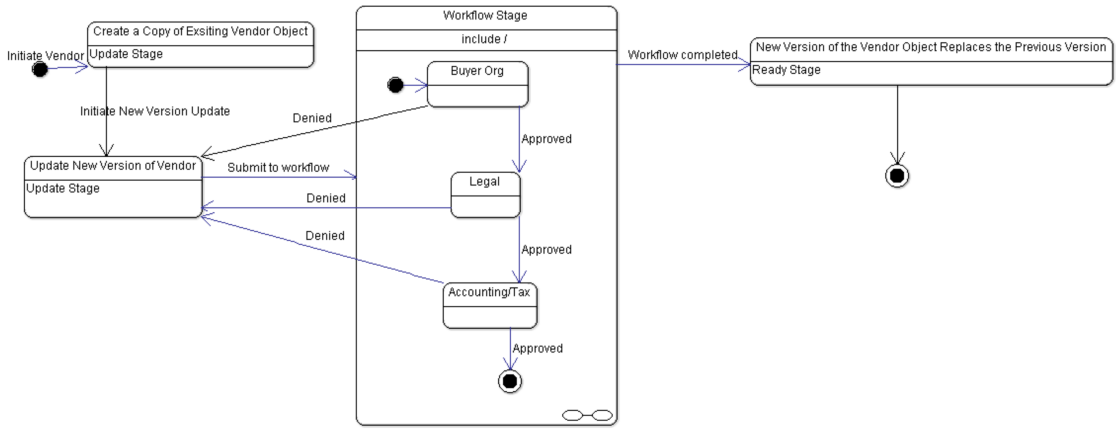


Sequence Diagram 1: The 3 Stages of Business Object State Transition Controller

validations, verifications and crosschecks have been performed on the vendor object, no further changes can be made to it while the business object is in Ready stage. When the business object enters Ready stage, it is available for consumption by other processes and entities. A new vendor business object can now be used to purchase supplies from.

When a business object needs to be updated (as opposed to being created new), a slightly different sequence of events is followed. Prior to making any changes to the business object, a copy (new version) of that object must first be made. There are a number of advantages to this approach:

1. While the new version of the business object is being updated, verified and validated, the original version is still available for consumption within the system.
2. Updates to the new version are made independent of the original version, which allows for audit trail and historical records keeping. Not only the changes themselves are tracked, but the owner of the changes (who made the change) and when the changes were made can be recorded.



Sequence Diagram 2: The 3 Stages of Business Object State Transition Controller for Update

The Business Object Life Cycle Management Pattern is not only applicable to data or documents created by humans, and should be applied to the data generated, exchanged or modified between processes. For example, a supplier might send a catalog file to its customer for use in the customer's procurement system. Upon receipt of the catalog file from the supplier, the customer's system should first submit it for some sort of review and acceptance process, before making the catalog available for general use by its employees. By following our pattern, this process would take place, and the supplier's catalog would go through the process of creation, validation and verification,and finally approval to become part of the customer's procurement system.

2.7 Consequences

- By adhering to the business object transition states, we can enforce as many or as little business rules and processes as needed, without changing the underlying structure of the business object.
- Employing the state transition controller pattern we can improve business object concurrency and synchronization, which can be applied using different rules for each state, thereby providing a find-grained implementation of business object concurrency and synchronization needs.

- The state transition controller pattern allows for version control or change management implementation of business object. For example, for business object updates, a copy of the existing business object could be used instead of the original. This way the original is still available and accessible in the system, while its copy is being worked on. Only when (or if) the copy reaches the Ready stage, would it replace the original version with its new changes.

3. KNOWN USES

Most, if not all major Commercial Off The Shelf (COTS) Enterprise Resource Planning (ERP) developers such as Oracle and SAP, to name the two biggest, utilize extensive business object life cycle management processes to ensure the business data integrity. Ariba Inc., which has recently been acquired by SAP [14] uses similar data object process in its architecture and software products [13]. Ariba uses the concept of an Approvable object, which is a representation of a business object. The Approvable object is similar to the business object presented in this paper. Within Ariba architecture the Approvable object goes through a three stage process before it is fully approved and can be used within the system. In Ariba, the three stages are: Composing, Submitted, and Approved. The Ariba architecture includes a workflow component which manages the Approvable object through the business life cycle. Similarly, only when Approvable object is fully approved does it become available to other processes and entities withing the system.

4. RELATED PATTERNS AND FRAMEWORKS

4.1 The Document-View-Presentation (DVP) Pattern

DVP [7] separates an application into three components: document, view, and presentation. The document component holds business logic and data. The view component is responsible for service requests and supplying the data to the document. The presentation component processes the events and provides data to the view component.

The Business Object Life Cycle Management Pattern differs from DVP in that its main goal is to address the document life cycle, and not the particulars of its presentation.

4.2 The Model-View-Controller (MVC) Pattern

MVC [15] has become one of the most popular and widely accepted patterns in the market today. Its main benefit is that it separates the presentation of information from the business logic and data storage. The model component in MVC is responsible for business logic and data storage. The view component handles the end user presentation of the requested data. The controller component handles the input from the view component and converts it into instructions for the view and/or model.

Unlike the Business Object Life Cycle Management Pattern, the MVC can only support a request/response mode, and does not provide the means for handling a business process in the form of a workflow.

4.3 The Presentation-Abstraction-Control (PAC) Pattern

PAC [16] defines a structure for interactive software systems in the form of a hierarchy of cooperating agents. In PAC, every agent is responsible for a specific area of the application's functionality and is comprised of three components: presentation, abstraction, and control. The abstraction component in PAC is similar to MVC's model component. The presentation component in PAC can be viewed as a combination of view and controller components in MVC pattern. The control component is responsible for facilitation between PAC agents.

The job of business processes is handled by the workflow component in the Business Object Life Cycle Management Pattern, whereas in PAC it is the responsibility of the agent itself.

4.4 Workflow Patterns

Workflow Patterns [8] describe various patterns of the workflow process, covering Sequence, Parallel Split, Synchronization, and many other patterns.

The Business Object Life Cycle Management Pattern differs from the Workflow Patterns in that it is at a more abstract level than the inner workings of a particular workflow, and is not concerned with the type of workflow pattern that is being employed in the data object's life cycle, but rather how it is being employed.

5. CONCLUSIONS AND FUTURE WORK

The three stages pattern can be applied to just about any data-driven business object within business environment. At the same time, it can be extended to address specific and more complex business needs. For example, a purchase request, before entering the Ready stage, might go through other interim stages (i.e. Sent to vendor, Shipped, Received, etc.). We believe that the Business Object Life Cycle Management Pattern is simple enough to be applicable to most business scenarios, yet easily extensible to add much more complex and more realistic business data and processes.

6. ACKNOWLEDGMENTS

We would like to thank Hans Wegener for his invaluable feedback and support during the shepherding of this paper.

7. REFERENCES

[1] A Guide To The Sarbanes-Oxley Act, The Sarbanes-Oxley Act of 2002, DOI=http://www.soxlaw.com/

[2] Fowler M., 1997. Analysis Patterns: Reusable Object Models, Addison-Wesley Longman.

[3] Daum B., 2003. Modeling Business Objecs with XML Schema, Morgan Kaufmann Publishers.

[4] Eriksson H.-E., Penker M., 2000, Business Modeling with UML: Business Patterns at Work, OMG Press / Wiley Computer Publishing

[5] Adams J., Koushik S., Vasudeva G., Calambos G., 2002, Patterns for e-business: A Strategy for Reuse, IBM Press

[6] Buckl S., Matthes F., Monahov I., Roth S., Schulz C., Schweda C.M., 2012. Enterprise Architecture Management Patterns for Company-wide Access Views on Business Objects, ACM Transactions on Applied Perception, Vol. 2, No. 3, Article 1, Publication date: May 2012.

[7] Chang K.Y., Chen L.S., Lai C.K., 1999. Document-View-Presentation Pattern, Department of Electrical Engineering, National Cheng-Kung University, Taiwan.

[8] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, 2004, Workflow Patterns, Department of Technology Management, Eindhoven University of Technology

[9] W.M.P. van der Aalst1, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, 2005, Advanced Workflow Patterns, Department of Technology Management, Eindhoven University of Technology

[10] Fortis A., Fortis F., 2009, Workflow Patterns in Process Modeling, European Conference on Computer Science & Applications

[11] Nick Russell, Arthur H.M. ter Hofstede, David Edmond, W.M.P. van der Aalst, 2005, Centre for Information Technology Innovation, Queensland University of Technology, Department of Technology Management, Eindhoven University of Technology

[12] Jianrui Wang and Akhil Kumar, 2005, A Framework for Document-Driven Workflow Systems, Smeal College of Business, Pennsylvania State University

[13] Ariba Inc., 2013. DOI=http://www.ariba.com

[14] Wong K. and Bass D., May 23, 2012. SAP to Acquire Ariba for $4.3 Billion in Push Into Cloud. Bloomberg.

[15] Glenn E. Krasner and Stephen T. Pope, A cookbook for using the model-view controller user interface paradigm in Smalltalk-80, Journal of Object-Oriented Programming, Volume 1 Issue 3, Aug./Sept. 1988, Pages 26 – 49

[16] Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. Pattern-Oriented Software Architecture: A System Of Patterns. West Sussex, England: John Wiley & Sons Ltd., 1996