A Proforma White Paper

Integrating Business Processes, Workflows, and Object Models via Use Cases

Bridging Business Models to Systems Design

proforma

26261 Evergreen Road, Suite 200
Southfield, Michigan  48076
248/356-9775  FAX 248/356-9025
E-mail proformacorp@proformacorp.com
Web site www.proformacorp.com
# Table of Contents

## INTRODUCTION .............................................................................................................................. 2

## THE APPLICATION DEVELOPMENT LIFECYCLE .............................................................................. 3

## ENTERPRISE MODELING .................................................................................................................. 4

### CONCEPTUAL MODELING AS THE STARTING POINT ................................................................. 4
- Workflow Model................................................................................................................................. 4
- Use Case Model................................................................................................................................. 5
- Workflow Models to Use Case Models............................................................................................. 6

### FROM CONCEPTUAL TO LOGICAL ................................................................................................. 7
- Use Case Model................................................................................................................................. 7
- User Interface Design....................................................................................................................... 7
- Functional (Component) Design..................................................................................................... 8
- Data Design.................................................................................................................................... 8
- System Interface Design.................................................................................................................. 8
- TO CODE ....................................................................................................................................... 9

## PROVISION WORKBENCH .............................................................................................................. 10

### MODELS, NOT SIMPLY DIAGRAMS .............................................................................................. 11

## SUMMARY ...................................................................................................................................... 11
Introduction

Our industry has long pursued a formalized, continuous application development lifecycle. We want to understand our business and gracefully evolve that understanding into the design and implementation of business systems. We want the deliverables of each lifecycle stage to flow into those of subsequent stages with forward and backward traceability. And we do not want to retool our development staff.

Manual attempts to record and manage the lifecycle details have proven to require more effort than our project deadlines will support. In response, organizations often employ shortcuts such as discarding business requirements at the beginning of design. Past efforts lacked a single methodology, especially in the area of analysis and design. As a result, it was difficult to organize teams that could maintain a single methodological focus.

Many of the better solutions required the use of a single CASE tool to which many organizations could not or would not conform. These monolithic tools required substantial investment in equipment and training to arm a relatively small portion of the organization.

Today, modern computers are abundant and powerful. Organizations do not have to invest in special high-powered workstations and boot-camp training to roll out a concept-to-code solution. Applications can more easily exchange data within the same machine or among machines with a drag and a drop. Everyone is ‘connected’ and routinely exchange data in electronic form.

New modeling techniques and approaches have evolved through the 90's, primarily due to extensive BPR and OO initiatives. Today, Workflow models, the Unified Modeling Language (UML), relational technology, Object-orientation (OO) and Use Case modeling represent standards that are largely unchallenged. Collectively, these are the key components of a continuous lifecycle.

This paper describes the ProVision Workbench™ approach and technology for using an enterprise modeling approach in the modern application development lifecycle.
The Application Development Lifecycle

The Application Development ‘lifecycle’ starts with the business (or enterprise) and ends in production code. As we traverse the lifecycle, we find numerous points of iteration, but ultimately the lifecycle progresses through three major (classical) stages: Conceptual, Logical, Physical.

The *conceptual* stage of the lifecycle involves understanding the business. Sometimes this is done to achieve improvements in business processes, other times it is used to guide the development or purchase of application systems. Work here is often called ‘Enterprise Modeling’ or ‘Business Analysis’. A key work product is often ‘Business Requirements’.

During the *logical* stage, business requirements help guide the engineering of system components. This stage is commonly known as system architecture and design. The key work product is ‘System Specifications’.

The *physical* stage commits engineering decisions of the logical stage to a particular platform. It involves the low-level engineering, coding, testing, deployment and maintenance. The key work product is a production system.

In essence, application development lifecycle involves two major elements, the business and its systems. Business Process (Re)Engineering (BPR) and complementary approaches represent the modern methodology for improving business processes and systems, and thus the business. BPR challenges current practices to invent new ways of conducting business which meet the demands of the market and embrace the capabilities of modern technology.

To effect changes in the business we employ technology. The modern methodology for analysis, design and development of systems is Object Orientation (OO). Use Case modeling, a very popular analytical technique adopted by OO, further adds an effective means of describing business situations in terms of interactions between people and system functions.

This paper describes how to integrate Business Processes and Workflows, Use Cases, and Object Models to provide an enterprise with a highly effective way to understand and represent both business and system requirements.
Enterprise Modeling

Conceptual Modeling as the Starting Point

Enterprise modeling implements the conceptual stage of the lifecycle. This is where the enterprise is scoped into manageable pieces. Select pieces are then studied to improve business processes (BPR or Business Process Improvement) and/or to formulate business requirements (Business Analysis).

Enterprise modeling does not imply studying the entire enterprise. In many cases, it is conducted within a defined scope in preparation for new systems development or the purchase of enterprise software.

The following discussion focuses on two models, the Workflow model and the Use Case model, which play a major role in enterprise modeling.

Workflow Model

A key tool for enterprise modeling is the Workflow model (or process map). This model describes the precedence of business activities within a business process. It enables analysts to identify and correct bottlenecks and inefficiencies. In addition to precedence, the Workflow model can express which department is responsible for each activity. This is done by placing the activities of a given department within the ‘swimlane’ that represents the department.

![Figure 1](image-url) – The Workflow model describes the flow of activities within a business process
The above Workflow model (Figure 1) depicts the activities involved in fulfilling orders and the departments or systems responsible for each activity. Since business processes can be very complex, it is often useful to nest Workflow models rather than place all details on one model. This creates a hierarchy of models each of which describes the details within a single activity.

To achieve process improvement, activities and workflows (the links) carry substantial information such as the time and money they consume. By analyzing process costs and timing, business analysts can reroute workflows, consolidate or eliminate activities, etc. to design a better business process.

**Use Case Model**

The Use Case model is a very popular technique for expressing how people, departments and systems interact to accomplish a portion of the business (or system). It is a simple model yet provides a powerful means to express continuity from the business to supporting systems.

The essential idea of a Use Case model is to represent how actors (job functions, departments and external systems) interact with recognized business or system functions. A completed Use Case model represents a prototypical interaction from which numerous usage scenarios can be derived.

Simplicity is a key reason for the popularity of the Use Case model. It often depicts a concrete situation observed in the physical world.

Figure 2 depicts the interplay required to complete an order. This involves both validation and prioritization. The Order Entry clerk and the Order Entry System both participate in order validation. The Order Manager and Sales Manager participate in prioritization. Occasionally, an order will require special handling. The Use Case model depicts this with ‘Prioritize Special Order’ shown as an extension of the normal ‘Prioritize Order’.

The informality of the Use Case model contributes to its simplicity but hinders overall consistency. It is normal to produce many Use Case models, each representing a different perspective of the business.

Use Case models require careful management. These models express details within a rather small scope and pay little attention to other Use Case models. On a large project, Use Case models often contradict each other. It is also easy to overlook areas where the use case should be used, leaving holes in the requirements.
Serious projects benefit from a more global model to help avoid overlap and ensure the Use Case models collectively represent the entire scope of the effort.

**Workflow Models to Use Case Models**

The Workflow model is a natural host for Use Case models.

Each lower-level activity in a Workflow model depicts a significant but well-scoped portion of the business (or portion of a system). A Use Case model is used to expose how users interact with functions to accomplish the activity.

Figure 3 depicts the ties between the Workflow and Use Case models. The Workflow model covers a wide scope involving numerous departments performing a variety of activities. By itself this model provides both a broad and detailed view of the business.

The Use Case model complements the Workflow model. Each activity on the Workflow model is a candidate subject for a Use Case model. The Use Case model describes what happens inside of the activity in terms of functions (sub-activities or use cases) and the business and/or system entities which interact with these functions.
From Conceptual to Logical

The transition from conceptual to logical involves recognizing system components to implement the business components of the conceptual model. Use Case models provide an excellent vehicle for making this transition.

Use Case Model

The Use Case model plays a major role in the transition to the logical stage. It provides an abstract view of how the system users (and external systems) interact with automated business activities. This provides a structure for identifying and designing user and system interfaces.

User and system interfaces are a characteristic piece of the system architecture. Historically, this is also a point where lifecycle continuity broke down.

When developed under the structure of a Workflow model, the Use Case models collectively describe all key user and system interactions. Designing the user interface for the system involves inspecting each Use Case model to identify candidate user and system interfaces. A candidate user interface occurs whenever a business entity (e.g. a role or department) interacts with a function. The Use Case model of figure 4 states three candidate user interfaces and one candidate system interface.

User Interface Design

A candidate user or system interface provides a problem scope for design. The design process involves understanding the user interface in terms of the data and function provided to the user.

Several models come to play when describing an interface. The models and their usage depend largely upon the complexity of the interface. In many cases, a designer would first express the user interface as a class model. The class model depicts the classes, attributes, associations and methods involved in the
user interface. In complex cases the designer would produce an interaction model (or collaboration model) to describe the dynamic behavior of objects within the interface. In all cases, the designer would provide detail to the user interface in textual and/or storyboard form. This detail is primarily used to communicate design intent to the developer of the user interface.

The interface in figure 5 between the Order Entry Clerk and the ‘Validate Order’ function is further detailed in a class model and then as a storyboard. The class model depicts the data and function made available to the Order Entry Clerk during this interaction. The associated storyboard provides a graphical ‘sketch’ or rough idea of how the user interface should appear.

![Figure 5 - The Use Case provides a structured means of identifying user and system interfaces](image)

**Functional (Component) Design**

The first stage of design produces a class model to clarify the components which implement the system activity. Where needed, additional models such as collaboration or interaction models describe the dynamics of how the components interact to perform the activity.

**Data Design**

The data architecture is always designed for some data storage technology. Whether the underlying storage technology is Relational or Object-oriented, designers must deal with the management of persistence. For Relational technology, the engineering of elegant persistence mechanisms starts by mapping the persistent data for all components of the Function Architecture into corresponding relational tables. Subsequent steps include classical relational table design and providing a clean abstraction for interfacing with the function architecture.

**System Interface Design**

The interface architecture, like the presentation architecture, starts with the Use Case models. Each interaction with an external system is a candidate interface. Figure 4 depicts an interface between Validate Order and the Order Entry System. The design for this interface involves describing the net data exchanged via a class model followed by low-level engineering based upon the technology employed (e.g. CORBA, batch files, inter-process communication).
Integrating Business Processes, Workflows, and Object Models via Use Cases

… to Code

The purpose of an application development lifecycle is to produce a working system which accurately supports the business.

Systems are developed (appropriately) from a design. The design specification includes the architectures and the design of the key components (user/system interfaces, functional components, relational tables, etc.).

Our industry has an incredible number of authoring/development tools. These tools support the low-level design and development of code. Appropriately, these tools focus on code and do not attempt to provide a monolithic approach covering the entire lifecycle. A practical concept-to-code solution is one that enables conceptual and logical components to interface with the varied authoring environments in use by the development organization.

ProVision Workbench™ (PVW) is designed to complement the existing base of authoring environments. PVW provides an integrated repository which provides intimate traceability from Enterprise models through the generation of C++, Java and DDL code.

The transition from logical to physical essentially involves generating code that reflects design intent. From this point, developers employ authoring environments to conduct detailed algorithmic design, coding and testing.

Since development often reveals changes in design and business requirements, a two-way interface is required. The key technology for interacting with authoring environments is the Microsoft COM interface. Developed components from authoring environments are reverse-engineered to extract design information and automatically update the design models in the PVW repository.
Integrating Business Processes, Workflows, and Object Models via Use Cases

ProVision Workbench

ProVision Workbench™ (PVW) is the award-winning product that enables concept-to-code using established code-authoring tools. It is the only analysis and design tool on the market that explicitly ties Workflow models to use cases and class diagrams, providing an easy transition from business analysis to system design.

PVW supports simple to sophisticated enterprise models using the most popular workflow methodologies. In addition to the Workflow model, models exist to represent other dimensions of the business. The enterprise models are:

<table>
<thead>
<tr>
<th>Business Models</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Interaction Model</td>
<td>Provides a strategic view of the business. Profiles and scopes the business showing organizational boundaries and interactions with internal and external organizations.</td>
</tr>
<tr>
<td>Workflow Model</td>
<td>Represents a business process in terms of component activities and the flow of work among these activities.</td>
</tr>
<tr>
<td>Goal Model</td>
<td>Show the hierarchy of business goals associated with business processes and their activities.</td>
</tr>
<tr>
<td>Process Model</td>
<td>Shows the decomposition of the business domain into process and activities.</td>
</tr>
<tr>
<td>Organization Model</td>
<td>Represents the organizations and supporting roles of the enterprise in a hierarchical fashion.</td>
</tr>
<tr>
<td>Location Model</td>
<td>Shows the various geographic locations of interest to the enterprise.</td>
</tr>
<tr>
<td>Event Model</td>
<td>Organizes key business events and sub-events.</td>
</tr>
<tr>
<td>System Model</td>
<td>Represents the systems and subsystems which support the enterprise.</td>
</tr>
</tbody>
</table>

The transition from conceptual to logical is best characterized by the Use Case model. However PVW supports additional models to enable a very detailed description of business requirements:

<table>
<thead>
<tr>
<th>Analysis Models</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Model</td>
<td>Depicts prototypical scenarios involving the ‘actors’ in the business (e.g. roles, departments, systems) and the functions they use to accomplish business activities.</td>
</tr>
<tr>
<td>Business Object Model</td>
<td>Shows the properties and interrelationships of the business objects (e.g. Order, Customer) of interest to the business.</td>
</tr>
<tr>
<td>Subtype Model</td>
<td>Depicts the fundamental hierarchies of business objects exposing the supertype / subtype relationships.</td>
</tr>
<tr>
<td>Interaction Model</td>
<td>Shows how objects collaborate, via message passing, to accomplish a given function.</td>
</tr>
<tr>
<td>State Model</td>
<td>Shows the lifecycle of individual objects for a given class in terms of states (key stages in the life of an object) and transitions among those states.</td>
</tr>
<tr>
<td>Method Model</td>
<td>Shows the delegation, in terms of messages, required to implement a single method.</td>
</tr>
</tbody>
</table>
Models, not simply Diagrams

A diagram is a picture. Each diagram can convey thoughts graphically, but offers no support for:

- integrity
- properties beyond the generic name and description
- sharing or reuse of objects on other diagrams
- intelligent functions such as completeness checks and traceability
- integration with other diagrams (reflecting a single change across all diagrams)

Each PVW model is a view into a single integrated meta-model. Changes from one perspective are automatically reflected in others. Each model consists of a (logically) unlimited number of objects and links among these objects.

Each PVW object is intelligent. It knows how it participates in models and how to interact with other objects. In addition, each object carries characteristic information far beyond name and description.

Intelligent, integrated objects enable quality models which work together to effect transitions from lifecycle stages and provide automatic forward and backward traceability.

Summary

ProVision Workbench supports an integrated enterprise analysis and development lifecycle which co-exists with existing code development/authoring environments. It supports the most common Workflow and Object-oriented methodologies with primary focus on the Unified Modeling Language (UML). This support extends across the enterprise using the multi-user capabilities inherent in the product.

The intent of this paper is to explain the basic ideas behind enterprise modeling and application development solution of ProVision Workbench™. To remain brief, the discussion omitted many capabilities of the product such as multi-user, methodology customization, and automatic model interpretations. For further information, please contact our website at http://www.proformacorp.com/.