Business Use-Case Modeling

Mike McCormick
August 2011

A visual model of a business can provide important insights into whether it is doing the right thing and how it might be improved. The Unified Modeling Language (UML), the de facto standard visual modeling notation for the analysis and design of software systems, can be used effectively to create such a model.

At a time when businesses are becoming more and more automated -- when oftentimes a computer system makes up the largest part of a business -- understanding the business and how it works is critical to successful business case. Existing businesses evolve and change; new businesses can require many complex interconnecting pieces. In both instances, a visual model of the business can provide important insights into whether it is doing the right thing and how it might be improved.

Business analysts can use the same notation and tools to document business processes that software architects and designers use to document software systems. By "speaking the same language" the two groups can communicate better, ensuring that software systems really meet business needs.

Software teams also need business models for other reasons. The role of software has changed. It is no longer about **cool features or dashboards**. Instead, commercially driven software projects are becoming more business focused, and the emphasis has shifted from **technical innovation** to **Return on Investment (ROI)**.

Software must be delivered rapidly, in increments driven by business value rather than technical needs. In this environment, it is crucial for an IT team to have descriptions of the business that allow them to make informed decisions. They need an unambiguous description of how the business looks that specifies where the value and cost factors are associated.

A good business model provides a software-independent description of the business processes to be automated, thereby promoting a good understanding of priorities and risks prior to technology selection.

Business Use-Case Model

So what does a good business model look like? First, it consists of two major parts: a business use-case model and a business object model; you can create both using the UML.

Let's begin with the **business use-case model**. Business use-cases describe business processes. These processes are documented as a sequence of actions that provide observable value to a business actor.

Another way to think of a business use-case is that it documents a particular business workflow. The main use-cases in ***Figure 1*** are "individual check-in" and "group check-in." The diagram also shows "business actors": the stick figures labeled "passenger" and "tour guide." To fully understand the purpose of a business, you must know who the business interacts with; that is, who puts demands on it, or is interested in its output.

The different types of "interactors" are represented as business actors. These are most often customers, but they could also include suppliers, partners, potential customers (the "market place"), local authorities, or colleagues in parts of the business not modeled.

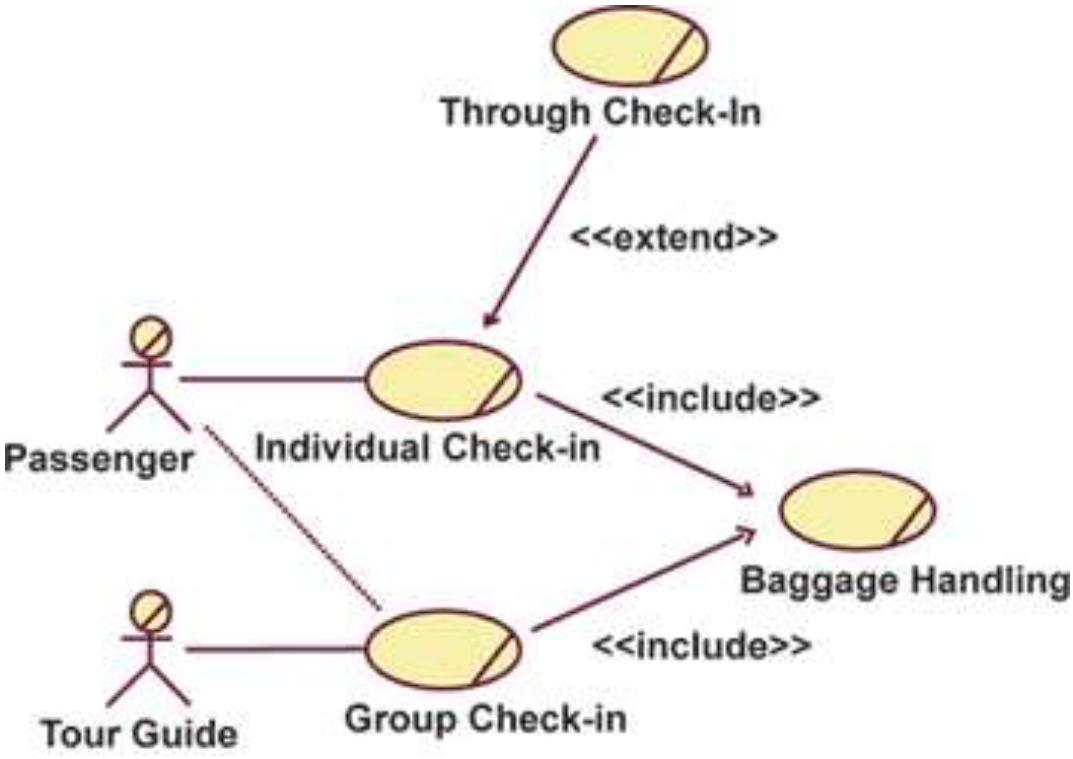

Figure 1: Use-Case Diagram for Several Business Processes

The detail associated with a business use-case is documented in a business use- case specification. This will include text as well as one or more UML activity diagrams and possibly system use-case diagrams. The following items are normally included in a business use-case specification:

- Name
- Brief Description
- Performance Goals
- Benefit / Value
- Workflow / Flow of events
- Special Requirements
- Extension Points
- Relationships
- Activity Diagrams
- Use-Case Diagrams

The key item is the workflow/flow of events, which describes ***what*** the business does to deliver value to a business actor, not ***how*** the business solves its problems. The description should be understandable by anyone within the business.

The structure of the workflow is described graphically with the help of a UML activity diagram. It provides a pictorial representation of the workflow structure described in text in the business use-case specification. ***Figure 2*** shows a sample activity diagram.

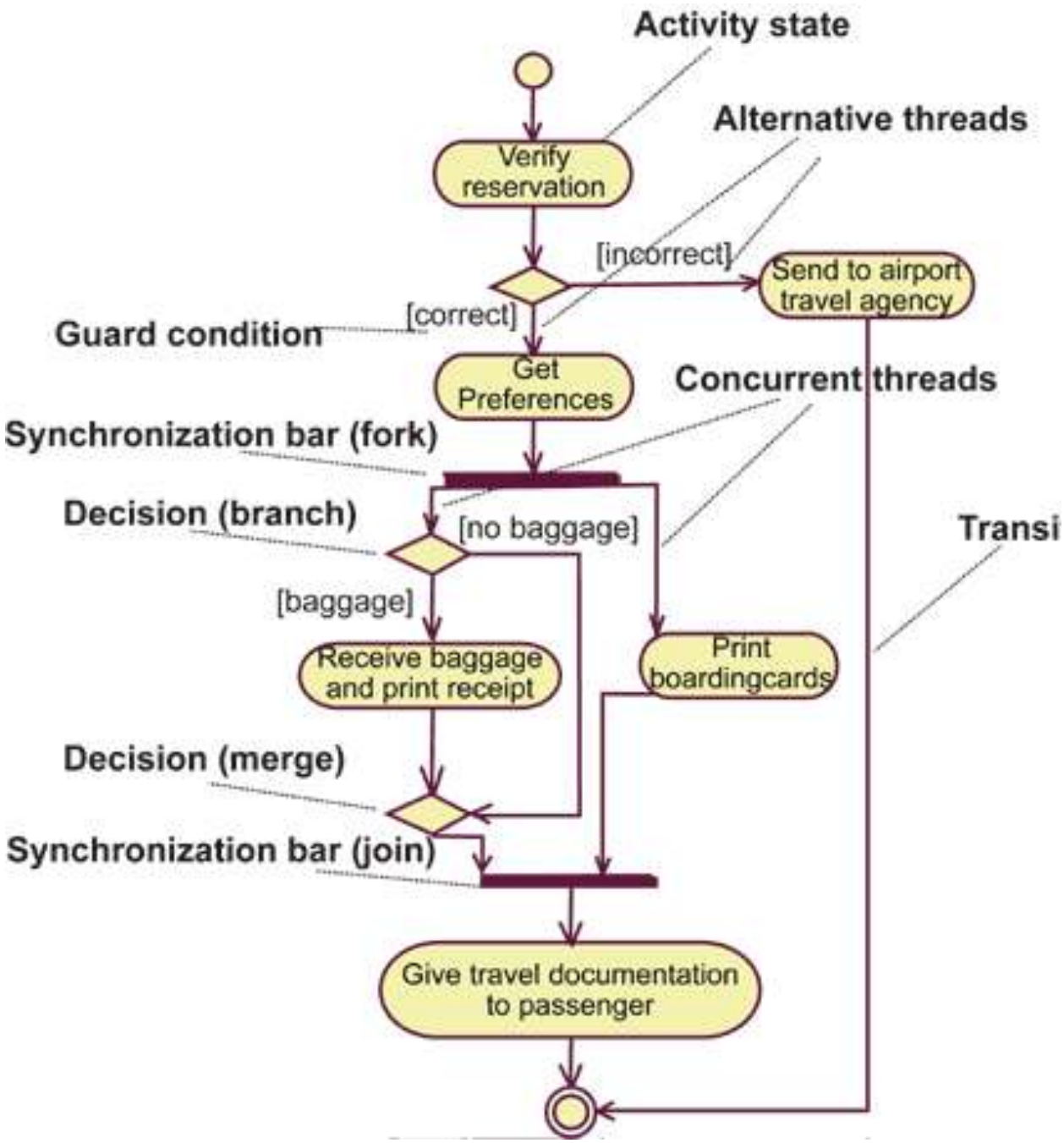

Figure 2: UML Activity Diagram Depicting Workflow Structure

The components of this diagram are:

- Activity states that represent the performance of an activity or step within the workflow.
- Transitions that show what activity state follows another. This type of transition is referred to as a *completion transition*. It differs from a transition because it does not require an explicit trigger event; instead, it's triggered by the completion of the activity that the activity state represents.
- Decisions for which a set of guard conditions are defined. Guard conditions control which transition, of a set of alternative transitions, follows once the activity is complete. You may also use the decision icon to show where the threads merge again. Decisions and guard conditions allow you to show alternative threads in the workflow of a business use-case.
- Synchronization bars that are used to show parallel sub flows. Synchronization bars allow you to show concurrent threads in the workflow of a business use-case.

The activity diagram shown in ***Figure 2*** provides a high-level, "macro activity" view of the business use-case. It does not indicate who performs a given activity or what is produced by an activity; that is part of the business object model described below.

To recap, the first part of a business model is a business use-case model. It consists of one or more use-case diagrams that contain one or more business use-cases. Business use-cases are documented via specifications that are partly text (most important: a workflow description) and partly graphical (activity diagrams). The business use-case model provides the big picture from a business actor's perspective.

Business Object Model

The second part of a business model is the business object model. Whereas a business use-case model tells *what* a business process will do, a business object model tells *how* it will be done. It serves as an abstraction of how business workers and business entities need to be related and to collaborate in order to perform the business. Figure 3 shows part of a business object model. It is a business class diagram showing business workers (the circles with stick figures) and business entities.

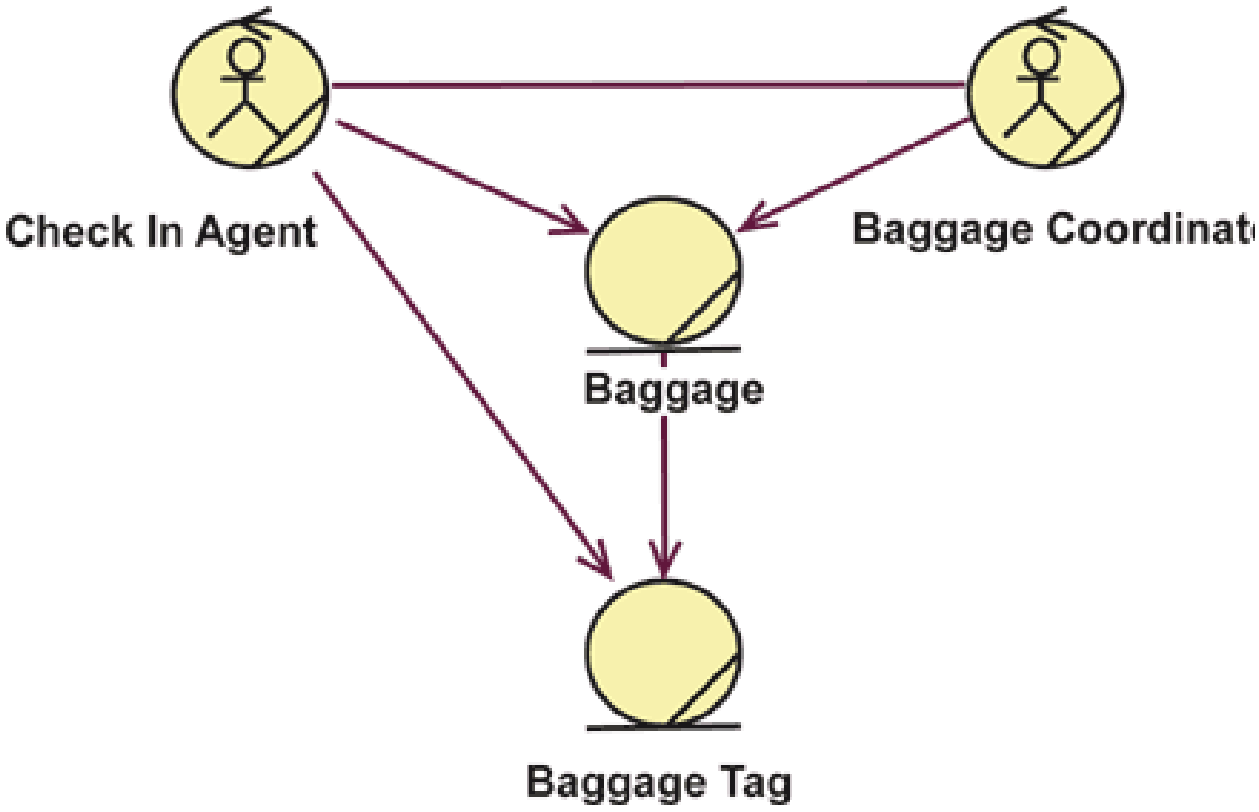

Figure 3: Business Class Diagram Showing Relationships Between Workers & Business Entities

It indicates how workers relate both to each other and to "things" within the business.

Another type of diagram used in a business object model is a variation of the activity diagram in ***Figure 2***. In this case, ***Figure 4*** the diagram also contains "swim lanes," or columns, that show which business worker is performing each activity. It "drills down" into the details of how a given business “use-case” is implemented.

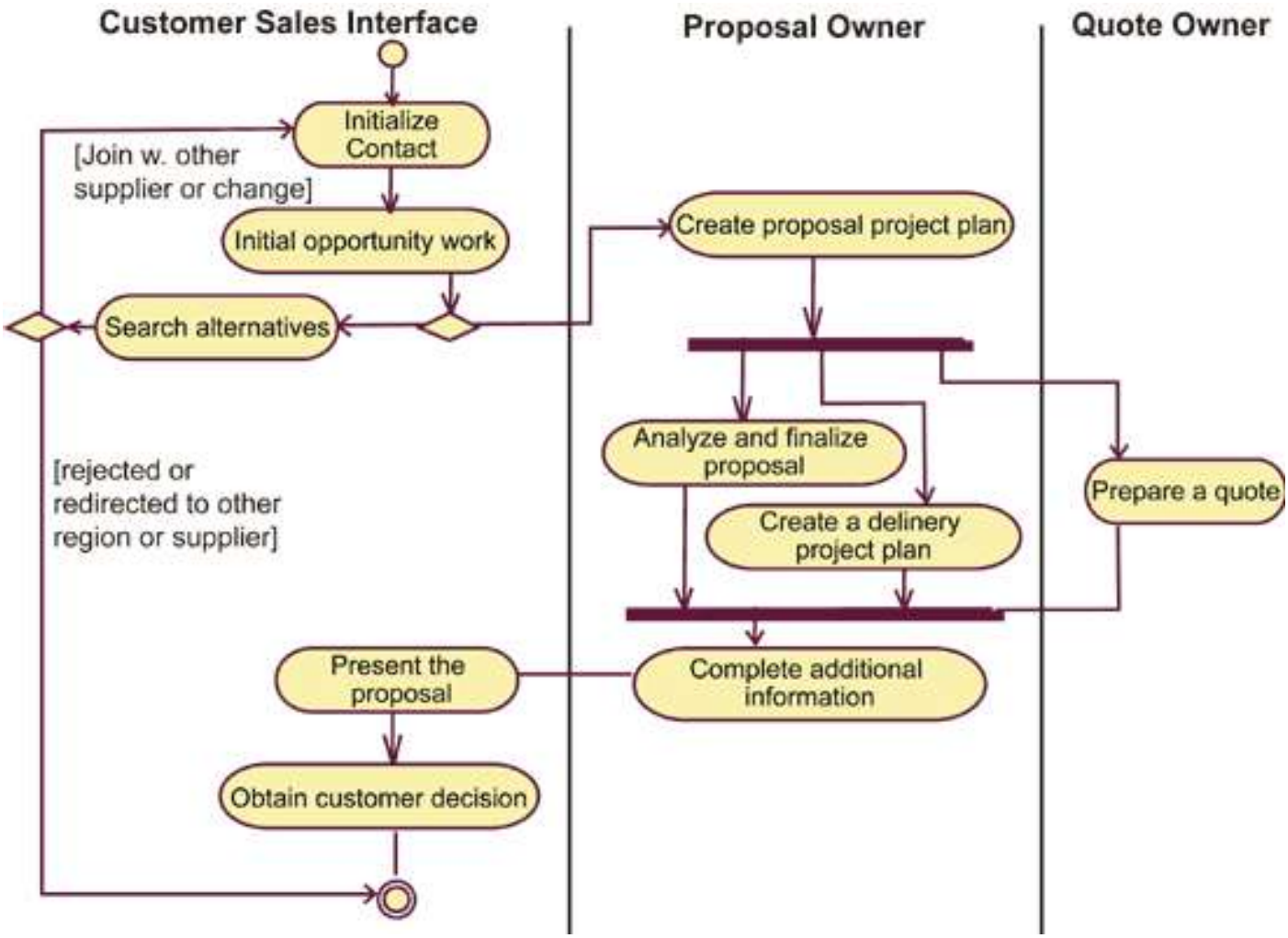

Figure 4: Activity Diagram Showing Who Performs the Activities

A third type of diagram used in a business object model is the business sequence diagram. A sequence diagram graphically depicts details of the interaction among business workers and business actors, and also shows how business entities are accessed, during the performance of a business use-case.

A sequence diagram like the one in ***Figure 5*** briefly describes what participating business workers do, how they communicate by sending messages to one another, and how relevant business entities are manipulated. Sequence diagrams may also show the interaction of a business actor with the business.

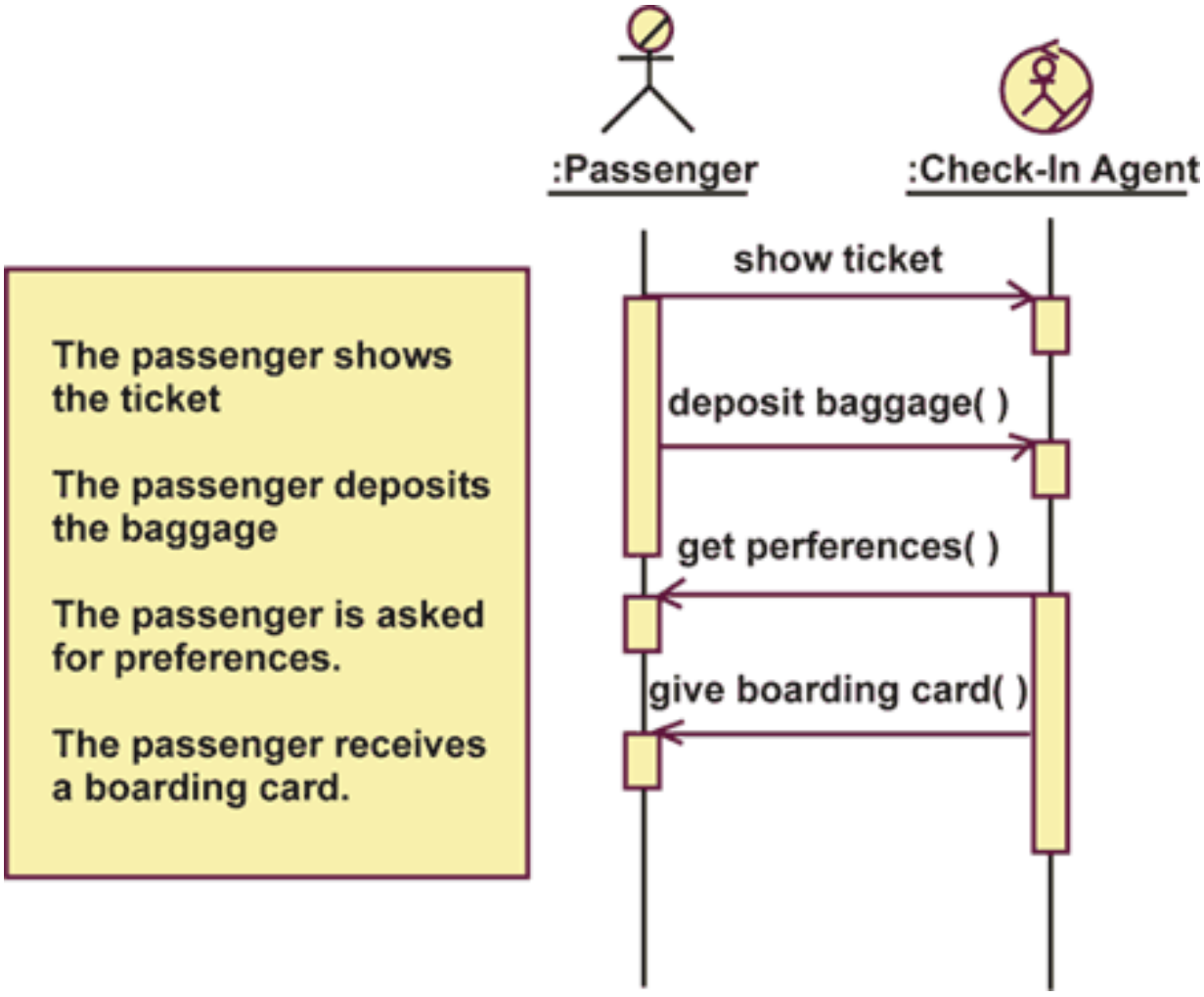

Figure 5: Business Sequence Diagram Showing a Business Actor Interacting with a Business Worker

Together, all of these various diagrams make up the business object model. This model provides detailed information on how the business process is implemented.

Business Modeling and the Software Development Lifecycle

While business modeling can be done either independently or as part of a business process reengineering effort, we are interested in how it can help improve the software development process. Clearly, it helps to define system requirements. For a development team, part of the requirements process is to analyze the problem being solved, and doing that analysis in the greater context of the business can help ensure that the system they build will really meet business goals.

In fact, a UML-based business model can be a direct input to a requirements model. The Rational Unified Process (RUP) defines direct mapping between the two models, and if you map a business model to an analysis model:

- A business process that is to be automated will be represented as a use-case;
- A business use-case will become a subsystem;
- Each business entity will become an entity class.

This mapping provides a head start for the requirements and analysis and design workflows.

Although in well-understood business situations, business modeling is often not needed, when an organization is complex and trying to automate significant functions, it can be invaluable. Similarly, if you are starting a new business, then defining a model can provide valuable insight into where automation can provide the greatest benefit. In today's competitive market, making sure that you solve the right problem in the business context can mean the difference between success and failure for your entire business and using the UML to model your business and requirements can help you get there.