UML and Simulation

The Unified Modeling Language (UML) is a language standard for developing and documenting the systems model of a software intensive system. The UML defines several types of diagrams to view the dynamic aspects of a system. Two of these diagram types are Activity Diagrams and Use Case Diagrams.

Activity diagrams may be used to document workflows in a system, from the business level down to the operational level. An activity diagram is essentially a flowchart, showing flow of control from activity to activity over time. Activities ultimately result in some action, which results in a change in state of the system or the return of a value.

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the analysis phase of software development to articulate the high-level requirements of the system.

UML diagrams are appropriate for system analysis, design, and development, but not business process workflow analysis. The dynamic business modeling capabilities for predicting the business metrics of a workflow design, including queuing and variability, are non-existent in the UML framework. To design the functional and business work flow, a discrete event simulation modeling tool such as Arena is often used.

The gap between business analysis and systems analysis is the basic motivation for developing new Arena to UML interfaces. If Arena can export appropriate information from a simulation model in UML format (e.g.,UML Activity and Use Case diagrams), then UML modeling tools can import that information to support any ensuing software development. Likewise, users may wish to import UML activity diagrams into Arena for process simulation purposes.

Using Arena's Export to UML Add-On

The objective of this feature is to generate abstract descriptions of a business process definition from an Arena simulation model in Unified Modeling Language format.

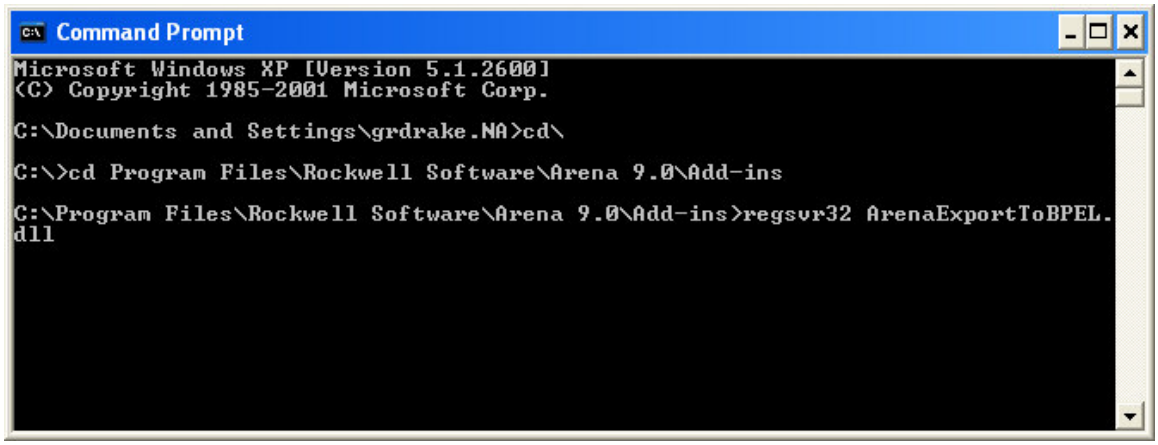
The *Export to UML* add-on prototypes an integration mechanism between Arena and applications which can read UML-specific XML files.

The *Export to UML* add-on is a prototype and is intended for research and evaluation purposes only.

The *Export to UML* add-on was prototyped against Arena 9.0.

Installation

1. Place the file *ArenaExportToUML.dll* in the Arena programs *Add-Ins* folder (e.g., *C:\Program Files\Rockwell Software\Arena\Add-Ins*). Note that the add-ins folder location can be changed via **Tools > Options > Folders** dialog.
2. The dll must now be registered. Open a windows Command Prompt window and navigate to the add-ins folder where the file was placed. Then execute the following command: *regsvr32 ArenaExportToUML.dll.*

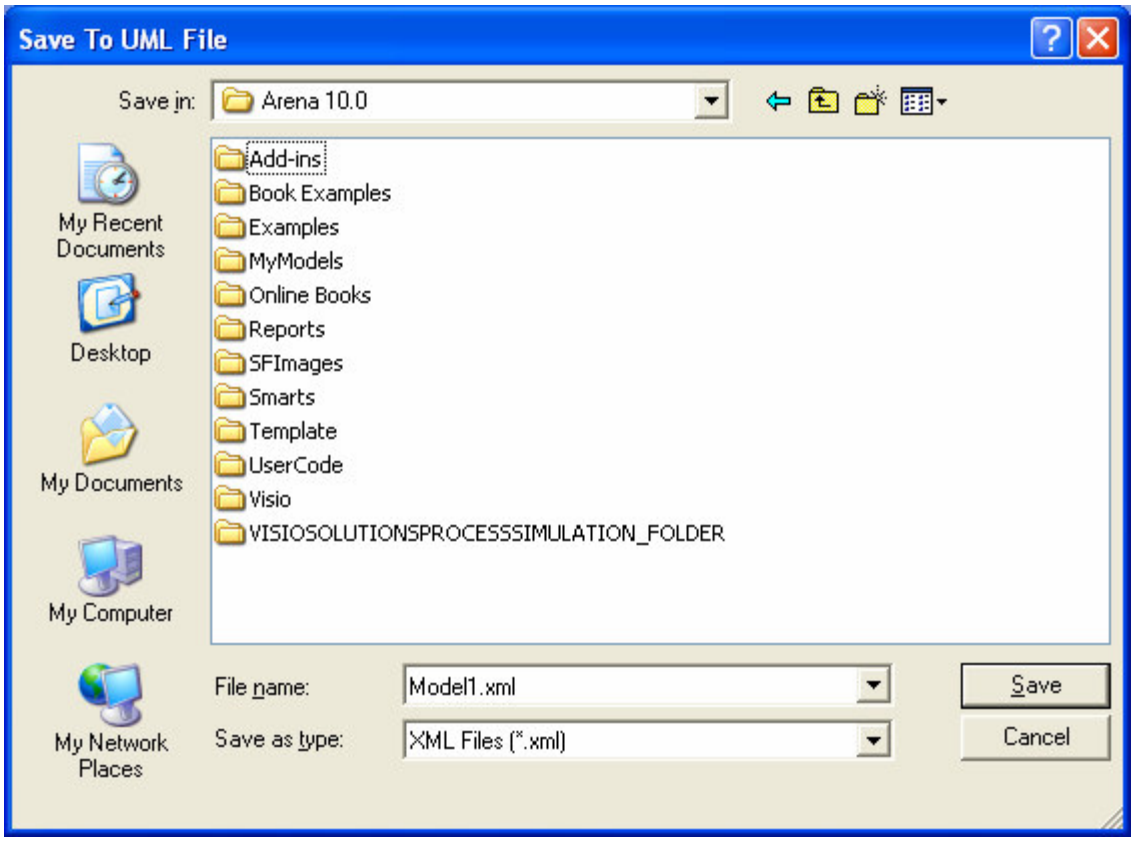


3. Start the Arena program. There should now be an *Export to BPEL* menu command available from Arena's *Tools* menu.

Running the Add-On

1. Open the Arena model that you would like to export UML from, and click on the Export to UML menu item from Arena's *Tools* menu.

The following dialog is displayed:



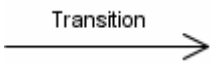
2. Specify the BPEL *File Name* that will be created and the folder to create it in.

Background on UML

UML: Activity Diagrams

Activity diagrams are used to document workflows in a system, from the business level down to the operational level. When looking at an Activity diagram, you'll notice elements from State diagrams. In fact, the Activity diagram is a variation of the state diagram where the "states" represent operations, and the transitions represent the activities that happen when the operation is complete. The general purpose of Activity diagrams is to focus on flows driven by internal processing vs. external events.

Notation

Activity/Action States	Activity states mark an action by an object. The notation for these states are rounded rectangles, the same notation as found in Statechart diagrams.	
SubActivity States	SubActivity states represent another activity graph. Can be used for functional decomposition.	
Transition	When an Activity State is completed, processing moves to another Activity State. Transitions are used to mark this movement. Transitions are modeled using arrows.	
Initial State	The Initial State marks the entry point and the initial Activity State. The notation for the Initial State is the same as in Statechart diagrams, a solid circle. **There can only be one Initial State on a diagram.**	
Final State	Final States mark the end of the modeled workflow. There can be multiple Final States on a diagram, and these states are modeled using a solid circle surrounded by another circle.	
Synchronization Bar	Activities often can be done in parallel. To split processing ("fork"), or to resume processing when multiple activites have been completed ("join"), Synchronization Bars are used. These are modeled as solid rectangles, with multiple transitions going in and/or out.	

More on Transitions…

A transition is a relationship between two states indicating that an object in the first state will perform certain actions and enter the second state when a specified event occurs and specified conditions are satisfied. On such a change of state, the transition is said to fire.

A transition has five parts:

1. Source State – the state affected by the transition.
2. Event trigger – The event whose reception by the object in the source state makes the transition eligible to fire (e.g., signals, calls, the passing of time, or a change in state), providing its guard condition is satisfied.
3. Guard condition – A Boolean expression that is evaluated when the transition is triggered by the reception of the event trigger; if the expression evaluates to True, the transition is eligible to fire;
4. Action – An executable atomic computation that may directly act on the object that owns the state machine.
5. Target State – The state that is active after the completion of the transition

UML: Use Case Diagrams

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis phase of software development to articulate the high-level requirements of the system. The primary goals of Use Case diagrams include:

- Providing a high-level view of what the system does
- Identifying the users ("actors") of the system
- Determining areas needing human-computer interfaces

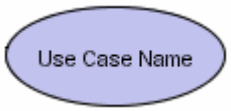
Use Cases extend beyond pictorial diagrams. In fact, text-based use case descriptions are often used to supplement diagrams, and explore use case functionality in more detail.

Graphical Notation

The basic components of Use Case diagrams are the Actor, the Use Case, and the Association.

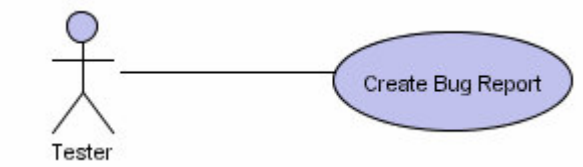
Actor An Actor, as mentioned, is a user of the system, and is depicted using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor.

Use Case A Use Case is functionality provided by the system, typically described as verb+object (eg. Register Car, Delete User). Use Cases are depicted with an ellipse. The name of the use case is written within the ellipse.



Association Associations are used to link Actors with Use Cases, and indicate that an Actor participates in the Use Case in some form. Associations are depicted by a line connecting the Actor and the Use Case.

The following image shows how these three basic elements work together to form a use case diagram.


Create Bug Report
Tester