The BPMN (Business Process Model and Notation) is the industry standard notation for modeling business processes. A subset of this notation is supported by Rational Software Architect.

Business process models are primarily used in early analysis phases, and are often rather informal in their nature. However, the formal basis of BPMN still makes such models formal enough to support simulation. In this article we will investigate how to build and simulate business processes using BPMN in Rational Software Architect.

We will cover three different aspects:

- How to run a simulation session for a business process to check the consistency of the process diagrams.
- Investigate how business processes impact on deployment topologies during simulation.
- How to simulate several business processes that are part of a BPMN collaboration.
- How the different BPMN concepts execute during simulation.

Before reading this article it is a good idea to have a general understanding of Rational Software Architect, BPMN and topology modeling. Some basic knowledge of UML is also useful. It is also recommended to have read the “Model Simulation in Rational Software Architect – Getting Started” article since some of the items we will encounter are introduced in this article.

Note that most of the features we will go through refer to news in version 8.0.2 and are not available in previous versions of Rational Software Architect.
Creating and Simulating a Business Process – an Example

Assume we are running an energy company which supplies customers on the Scandinavian market with different forms of energy (oil, gas, wood pellet etc.). Our company consists of a call center which acts as the contact point for customers who want to order energy from us. When a customer contacts the call center he will specify what kind of energy he wants, and how much. We will then make a delivery of what he has ordered, and also make sure that all information about this customer is saved in our corporate database, to ensure we can serve him quicker the next time he contacts us.

We also support customers to subscribe to automatic deliveries of energy. For example, a customer can tell us he wants 5 tonnes of wood pellet delivered every second month during winter. All information about the customer and his subscription is stored in our corporate database, which enables us to handle such deliveries automatically.

Let's model how our company works by means of a business process in Rational Software Architect. To do this we first create a new model project, using the File->New->Model Project command. Give the project the name ”EnergyCompany” and select to create the model from the standard template Business Process Modeling - Blank Process Model.

The project should look like this in the Project Explorer:

If the BPMN process editor for the ”EnergyCompany” process is not already open, double-click the process node in the Project Explorer to open it.

Now we use the BPMN editor palette to create a business process that describes our company:
As we can see in this diagram there are two ways in which this process can be started; either by a customer that contacts us, or automatically based on some timer that notifies us that one of our existing customers should get a delivery due to his active subscription contract.

Note that this is a very informal high-level description of how our company works. Several details are omitted at this early stage. However, already now we can validate that our business process is reasonable by performing a simulation of it.

We start the simulation directly from the process diagram using the context menu command **Execute As – BPMN Model**

After a short moment the simulation will start and we are asked if we want to switch to the Model Execution perspective.
Let's accept this suggestion.

The first thing that will happen when simulating this process (possibly even before the above dialog has disappeared) is that we are prompted to select which of the two Start Events to use for starting the process. Basically this allows us to either simulate the scenario when a customer contacts us manually, or the scenario when making an automatic delivery based on a customer subscription.

Let's start by selecting the "Customer contact" start event. In the process diagram we can see that the selected start event gets marked with the "next to execute" marker, indicating that this start event is the next element that will execute in the process.

Now take a look at the Debug View which is the main view used during model simulation.
Here we can see our simulated process represented by an instance node "1 : EnergyCompany" (meaning that this is the first instance of the EnergyCompany process).

The Debug View gives us the commands we need to execute the model. For now we only want to check it, so we use the Resume command ( ) that automatically runs through the process from start to end.

In our process there are two exclusive gateways: When these are reached during the simulation we will get prompted to select which of the outgoing flows to select. The first prompt asks us about which energy form the customer wants. Let's pick "gas".

The second prompt asks if the customer is a new customer. Let's assume he is and answer "yes" on this question:
The prompting dialogs that appear during simulation allows us to control the course of action for the simulated scenario. Basically our answers in these dialogs select a particular path in the simulated process. This path can be graphically visualized by turning on an animation preference. Click on the view menu in the Debug view and select *Model Execution – Animation Preferences*...

In the preference page that appears set the "Mark executed elements” preference to "Colorize executed elements”. Click OK. Now we can see the path of executed elements marked by red color in the process diagram.
Marking executed elements during or after simulation is useful in order to ensure that all relevant parts of the business process have been covered. For our process we can conclude that we need to run three additional scenarios to cover the entire process. Restart the simulation of the process by performing the Restart command that is available in the context menu of the session node in the Debug View.

This time select to start the process using the "Subscription timer" start event, and notice how two additional symbols get marked as executed:
As the next step in describing our business process we want to look at the communication that takes place between the different locations of our company. Note that our process diagram contains some service tasks (marked by the icon). These tasks represent usage of some services that are available in our company. The process uses the following services:

- **Lookup subscription.** This service is provided by our corporate database, and allows us to fetch all data we need in order to prepare a delivery based on an active subscription.
- **Search customer data.** This service is also provided by our corporate database. It allows us to find out if a customer that contacts us is a new or existing customer.
- **Register new customer.** Another service provided by our corporate database. It stores all information we need about a customer in the database.
- **Send invoice.** This service is provided by another company, since we don't have resources in our own company for handling everything related to customer payment.

Let us now go back to the Modeling perspective and create a small model of these services and their providers. We will use a UML component model, where each service provider is a component, and each service is represented by an operation.

Start by adding a new UML model to your project by performing **New – UML Model** in the context menu of the project:

Create the model from the **General – Blank Package** template and call the model file "UML Model". Then right-click on the created UML package and add three components to it (use the context menu command **Add UML – Component**). Give the components names as shown in the picture below:
Next create a component diagram in the package (Add Diagram – Component Diagram) and drag the components from the Project Explorer and drop them on the diagram. Use the editor to add operations to the Database and InvoiceCompany representing the services provided by these components. We also add an interface realized by the CallCenter component. This component is what runs the business process and we need the interface to be able to connect the UML component with the BPMN process. The UML component diagram should look something like this when you are ready:

Now we are ready to link this UML model to our business process. This is done from the Properties editor when selecting BPMN elements. First select the process itself (by clicking in the background of the process diagram) and click the Interfaces tab in the Properties editor. Use the browse button to find the ICallCenter interface:
Next, link each of the service tasks to the corresponding UML operation. Select each service task and use the Browse button for the Operation property to find the UML operation to link to:

- **Lookup subscription.** Link to Database::findSubscription().
- **Search customer data.** Link to Database::searchCustomer().
- **Register new customer.** Link to Database::registerNewCustomer().
- **Send invoice.** Link to InvoiceCompany::makeInvoice().

Now let's proceed by modeling the physical topology of our company. Basically we have a call-center where employees respond to customer contacts as well as notifications about subscriptions that need to be handled. We also have a data-center where our corporate database is located. Finally we want to include in the model the partner company that does our invoice handling, because this is of interest for our business process.

To describe the topology of our company we create a topology diagram. In the context menu of the project, perform the command *New – Topology*. Give it the name "PhysicalTopology".

Use the Topology editor palette to create three locations (available in the Logical Analysis tab). Then drag each UML component from the Project Explorer to the location where it resides. The diagram should look something like this:
This diagrams describes that our company is located in two different Swedish cities; the call-center is located in Sundsvall and the data-center is located in Stockholm. We can also see that the partner company that handles our invoices are located in Malmö.

The links between the locations are network communication constraint links, and they represent the fact that we have the ability to communicate from the call-center to the data-center and invoice company through some kind of computer network (for example the Internet).

Now let us simulate the business process again, this time looking at what communication that will take place between our locations. First, open Window – Preferences and select Model Execution – Animation – Topology Animation. Ensure that the preferences called “Animate operation calls in topology diagrams” and “Show arrows for historic messages in topology diagrams” are enabled. Then right-click in the topology diagram and select Execute As – BPMN Model.

When the simulation session starts use the ”Subscription timer” start event and step through the simulation of this scenario step-by-step by repeatedly pressing the Step Event button ( ) in the Debug View toolbar. Arrange the windows so that both the BPMN process diagram and the topology diagram are visible at the same time. Whenever the simulation is about to execute a service task you will see a yellow textbox indicating that a service operation is about to be called. And when you perform Step Event that textbox moves across the constraint link illustrating that the invocation of the service task implies some kind of communication across this link. Also, an arrow will be drawn to show the history of communication.
Visual indication that the "Lookup subscription" service is about to be invoked. This request implies a communication from the call-center to the data-center.

Colored arrows showing the history of service invocations.

In the Debug View's view menu you can perform Model Execution – Show Historic Messages View in order to see the history of all service operation calls that have taken place in the simulation session. You will get a table that acts as a legend where you can see what service operation that corresponds to which colored arrow. For the above example this view looks like this:
Note that it is much easier to see the communication within the different parts of our company in the animated topology diagram, than to look at the BPMN process diagram. This is an important benefit of animating topology diagrams during business process simulation. If you notice intense communication between two remote locations when executing a business process it is an indication that either the process itself should be modified (to avoid too much of such communication) or that the deployment topology should be modified (for example by co-locating the communicating components).
Simulating BPMN Collaborations

A BPMN collaboration describes the interaction between different business processes. The collaborating processes are participants of the collaboration. As of RSA 8.0.3 it is possible to simulate a BPMN collaboration in order to visualize the communication between the participant processes.

A BPMN collaboration is defined using a collaboration diagram. There is one horizontal “pool” for each participant, and typically there are some message flows between elements in different pools. These message flows represent the communication that takes place between the participants. Here is an example:

We can simulate a BPMN collaboration by right-clicking in the collaboration diagram and then perform the Execute As - BPMN Model command:

Note that you need to right-click in the diagram (or on the collaboration in the Project Explorer). If you instead right-click on one of the pools in the collaboration diagram, only the process of that pool will be simulated. In this case none of the incoming message flows can occur since the other participants of the collaboration do not execute. To ensure that you are actually simulating the entire collaboration and not only one individual business process, you can look in the Debug view. You should see one node for each executing business process grouped under a node...
representing the simulated collaboration. For the above example the Debug view will look like this:

![Debug view](image)

The simulation of a collaboration proceeds in much the same way as when simulating a process. When you perform Step Event the participating processes will take turns in executing the element that is "next to execute". As usual you can control the order of execution by Shift-clicking on token decorators within the individual processes.

When a message is sent and received across a message flow, you can visualize this by turning on the following preference: Model Execution - Animation - BPMN Animation - Animate message flows in collaboration diagrams.

Messages are visualized by means of a blue text box that moves from the sender to the receiver element across a message flow. You can also get colored arrows in the collaboration diagram to see all messages flows that have taken place historically during the simulation session. To see these arrows enable the preference Model Execution - Animation - BPMN Animation - Show arrows for historic messages in collaboration diagrams.

Here is an example of what the above collaboration can look like when simulation has run for a while:

![Collaboration diagram](image)

As usual the numbers within the colored arrows are explained in the Historic Messages view:
The communication between the processes that participate in a collaboration can also be visualized in a topology diagram. The requirement for getting this visualization is the same as when simulating individual business processes; each business process in the collaboration must be linked to a UML interface which in turn is realized by a UML classifier (for example a component or an actor). The UML classifier should then be shown by a topology unit in a topology diagram. For the above example we may use the following UML model:

During simulation we will then see the following in a topology diagram that shows the "Client" actor and the "Support Center" component:
In the rest of the article we will focus on different aspects of business process models and how they will affect a simulation session.
Business Process Symbols and Simulation

In this section we will go through the different symbols that can be created in a BPMN process or collaboration diagram and their impact when simulating a model.

A BPMN process is a graph consisting of nodes and edges (corresponding to symbols and lines in the process diagram). Before we go through the details of how individual nodes in a process execute it is important to get an understanding of what happens during the simulation of a business process. This is to a large extent based on the idea of a “token flow”.

A “token” is the notion used in BPMN models to denote a flow of control. During execution a business process has a number of tokens floating around through the nodes and edges. When we simulate the process the tokens are animated as small green circles with a number.

In general there can be any number of tokens present in one business process at the same time. There can also be any number of tokens at any location of the process at the same time. The number in the green token circle corresponds to the number of tokens at this particular location in the process.

When discussing the different symbols below we will usually describe them in terms of tokens on incoming and outgoing connectors for the symbols.

A node is said to be “enabled” if it is allowed to execute. For most nodes the simple rule is that they are enabled if they have at least one token on at least one incoming edge. There are some exceptions to this rule but for most nodes this simple rule holds.

During simulation we can see the node that is selected for execution in the activity as it is marked with a “next to execute” marker, a small green circle surrounded with four black triangles.

This is the node that will execute next if we press the Step Event button. We can however move the “next to execute” marker to any other node that is enabled by holding down the Shift key and then click on a token. This will move the “next to execute” marker to the node that is enabled by the token we clicked on.

If we look at the symbol palette for a BPMN process diagram we can see that there are three sections with symbols that can be added:

- Basic BPMN Elements
• Events
• Gateway Elements
Let's go through these sections one by one.

**Basic BPMN Elements**

The only symbols in this section that have an impact on simulation is the Connector link, the Task symbols and the Call Activity. For collaboration diagrams there is also Message and Pool which are relevant for simulation.

**Connector**

Connectors are what connect the different nodes of a business process. They define paths along which tokens can flow. A connector may be given a name. When there are multiple outgoing connectors from a gateway node you should name the connectors so that it becomes clear which of them to select during simulation. The names of the outgoing connectors may then appear in a dialog that appears at run-time when the gateway executes. For example:

![Diagram showing a connector and its name in a dialog]

If the connectors are unnamed you have to select the entries in the dialog to see to which connector it corresponds (it will be selected in the diagram).

In a BPMN collaboration diagram there is also a different kind of connector: a message flow. This kind of connector connects elements in different pools of the collaboration. Hence, they represent communication between the different business processes that participate in the collaboration. During simulation both the sending and receiving of a message across a message flow will be animated, provided that the preference *Model Execution - Animation - BPMN Animation - Animate message flows in collaboration diagrams* has been enabled.
Task

A task represents an atomic piece of activity that takes place. There are a number of different kinds of tasks to choose from in the editor palette:

- Task
- User Task
- Business Rule Task
- Service Task

From a simulation point of view there is no difference between the different kinds of tasks. When a task executes tokens on incoming connectors will be consumed, and all outgoing connectors are offered a token.

However, if a Service Task is used it is possible to specify a UML operation that represents a service that is invoked when the service task executes. Linking service tasks to UML operations enables you to visualize communication with service components in a topology diagram.

Use the Properties editor to link a service task to a UML operation. All the following must be done to ensure that a topology diagram gets animated during simulation of a BPMN process diagram:

1. In the preference page Model Execution – Animation – Topology Animation the preference called "Animate operation calls in topology diagrams" must be enabled. Calls to service task operations will then be shown using yellow text boxes in the topology diagram. If you also want to see colored arrows representing historic calls to service operations you should also ensure that the preference called "Show arrows for historic messages in topology diagrams" is enabled.
2. The business process itself must be linked to a UML interface which then must be realized by a UML classifier (for example a component). This classifier must then be present as a component in the topology diagram. All communication from this component to service components will be shown.
3. Each service task in the process diagram must be linked to a UML operation. This operation should either belong directly to a component shown in the topology diagram, or it may belong to an interface that is realized by such a component.

Call Activity

A call activity is used for calling another process from a process. It is hence useful in order to reuse a process multiple times from different contexts.
To specify what process to call, select the call activity, click the browse button for the "Called element" field in the Properties editor, and browse to the process you want to call.

When a call activity executes during simulation, the control will be transferred to the called process. When that process has finished its execution (i.e. it has no more tokens available) then control is transferred back to the calling process, and a token will then be offered to each outgoing connector of the call activity.

Note that a process may contain multiple parallel flows. It is therefore possible that the calling process still have tokens available while the called process executes. To control which tokens that will be consumed you can click on a token while holding down the Shift key. For example, you may want to consume some tokens in the calling process before the called process has finished its execution.

Message

A message may be associated with a message flow. From a simulation point of view it does not matter if a message flow has an associated message or not. However, if you choose not to associate a message with a message flow you should give the message flow a descriptive name, since this name then will be used when animating the message communication in BPMN collaboration diagrams as well as in topology diagrams.

Pool

A pool represents a process that participates in a collaboration. When you simulate a collaboration the simulation is actually performed by simulating the processes that are represented by the pools in the collaboration. You can also simulate a pool process in isolation. This behaves the same as to simulate a regular business process, and all incoming and outgoing messages flows are in this case ignored. Simulating the individual pool processes of a collaboration can be a good idea before simulating the entire collaboration, to ensure that the individual participants behave as expected.

Events

Events specify something that happens during execution of a process. All kinds of events that are available in the editor palette can be used, but only the start event and the end event have any special behavior during simulation. The intermediate events (catch and throw) will simply act as intermediate nodes during simulation, relaying incoming tokens to outgoing connectors.
**Start Event**

A start event defines where to start the execution of a process. When a simulation session is started the start event will be marked with the "next to execute" marker, indicating that this is where execution will start.

A process may have multiple start events. In this case a dialog will appear when simulation starts, that allows you to select which one to use for starting the process. In this case it is good practice to give descriptive names to the start events. For example:

![Select a start event to use for starting the process](image)

Note that if you press Cancel in this dialog, none of the start events will be enabled for execution, and hence nothing further can happen during the simulation. However, if the process was started because it was called from a Call Activity, then execution can proceed at the calling process instead.

Any triggers can be specified for a start event; the triggers have no impact on simulation.

**End Event**

An end event indicates that a path of a process ends. This means that during simulation all tokens that arrive at an end event will be consumed, without generating any new tokens. Note, however, that there may be other tokens available elsewhere in the process, so the execution of an end event does not mean that the entire process will finish its execution immediately.

Any results can be specified for an end event; the results have no impact on simulation.
Gateway Elements

Gateways are used for controlling the flows within a process. All kinds of gateways that are available in the editor palette can be used.

Exclusive Gateway

An exclusive gateway can be used to create alternative paths in a process. For example:

MODEL DIAGRAM

When such an exclusive gateway executes during simulation a dialog will appear, that allows you to specify which of the outgoing connectors to give a token to. For the above example the dialog will look like this:

If you press Cancel in this dialog, none of the outgoing connectors will get a token, and the exclusive gateway will not execute. The "next to execute" marker will in this case remain on the exclusive gateway element.

Another way of using an exclusive gateway is for merging several incoming flows to one outgoing flow. In this case the gateway will relay any token that arrives on any of the incoming connectors to the outgoing connector. For example:
Inclusive Gateway

An inclusive gateway can be used for creating alternative, but also parallel, flows in a process. When an inclusive gateway executes during simulation a similar dialog as for exclusive gateways will appear. However, the dialog for an inclusive gateway allows you to select more than one of the outgoing connectors. All selected connectors will get a token, which is why you can get multiple parallel flows from an inclusive gateway.

Here is an example:

Just like for an exclusive gateway it is possible to cancel this dialog in order to not execute the inclusive gateway.

Parallel Gateway

A parallel gateway can be used for creating parallel flows in a process, and also to combine such parallel flows back to a single flow. When a parallel flow executes during simulation all its outgoing connectors will get a token. Hence one flow will be created for each outgoing connector.

If a parallel gateway has multiple incoming connectors it will not be enabled for execution until they all have at least one token. Such a parallel gateway therefore synchronizes the incoming parallel flows.
Here is an example:

Here the upper parallel gateway has executed and yielded two tokens, one on each outgoing connector. You can Shift-click on tokens to decide which flow to execute if there are multiple parallel flows like in this case.

The lower parallel gateway can only execute when there is a token on both its incoming connectors. This means that it won't execute until both the 'Send Mail' and 'Send SMS' tasks have been performed.
More Information about Simulation

You can find more information about the simulation feature in the online help for Rational Software Architect available at http://publib.boulder.ibm.com/infocenter/rsahelp/v8/index.jsp and in other articles available on this wiki.

In particular the following wiki articles are relevant for simulation:

- “Model Simulation in Rational Software Architect: Getting Started” This is a very simple walk-through of how to design an activity and run it in a small simulation session.
- “Model Simulation in Rational Software Architect: Sequence Diagram Simulation”. A fairly detailed walk-through of how to design sequence diagram based models and use them for simulation.
- “Model Simulation in Rational Software Architect: Activity Simulation”. A fairly detailed walk-through of how to design activity based models and use them for simulation.
- “Model Simulation in Rational Software Architect: State Machine Simulation”. A fairly detailed walk-through of how to design state machine based models and use them for simulation.
- “Model Simulation in Rational Software Architect: Business Process Simulation”. This article!
- “Model Simulation in Rational Software Architect: Simulating UML Models” A detailed walk-through of all the commands, views and features available in the simulation support in Rational Software Architect.