# System Automation vs. Human Facilitation

BPMN defines the look of a process, but not how to draw it.

**Keith D Swenson, VP of R&D,**
**Fujitsu Computer Systems, USA**

## INTRODUCTION

Business Process Modeling Notation (BPMN) is a graphical notation standard broadly adopted across BPM vendor products. BPMN defines the way that a process "looks." Some people think that his means that a given process is modeled the same way in all such products, but this is not true. Some people jump to the conclusion that a process drawn a given way will execute the same on all process engines that support BPMN, but this is also not true. The purpose of this article is to give a concrete example of a process that will be drawn different ways depending upon assumptions about the target of the diagram.

There is a strong desire for a *lingua-franca* for BPM—a single consistent language for describing a business process, which would work in all situations for all purposes. As organizations expend more and more resources to define processes to run their business, a *lingua-franca* would preserve their investment and allow reuse of process models across a variety of situations. Many people have hoped that BPMN would be that single common language for a process.

This article dispels the myth that BPMN defines a single way to draw a process. This is not what BPMN was defined to be, and there are good reasons why it cannot be as such. BPMN is a notation and not a language. The distinction between these is sometimes elusive. BPMN was designed to be methodology agnostic, so that it can be used in both business situations as well as technical situations, and this design goal is probably the reason for its success. This article will explore in some depth two different methods for designing processes, both of which use BPMN, aimed at two different audiences, and for implementation on two different process platforms.

This article will help everyone gain an appreciation that these differences in modeling are not flaws due to arbitrary differences in approach, but necessary differences due to the many different aspects of process modeling. Two approaches are compared. One is a data-centric methodology that focuses on how bytes are passed from one system to another. The other is a human-centric methodology that focuses on the activities that people do. It should not come as a surprise that a model of what people do will look different than a model of what data is passed between computer systems, even though the process is the same.

This article will cover:

- Setting the Stage—Brief overview of BPM standards and popular assumptions about their use as well as a few myths that need busting.
- Defining the Human Activity—how this differs from an activity in a Business Process Execution Language (BPEL) process and why this matters

- A Method for Defining Human Processes—an approach that focuses on activities of people and how they interact, instead of focusing on how data flows through the system
- The Email Voting Process for BPEL—diagrams from the process as it is presented in the BPMN specification
- The Email Voting Process for Humans—how BPMN can be used to define a human process that is the same process as the one in the BPMN spec, but looks completely different
- Discussion and Conclusion

In the end we will see that a diagram of a human process focuses on the human activities, and that this is useful for training people, as well as explaining where they are in the process. This diagram can be drawn using the BPMN standard, but it does not look the same as a diagram drawn for a BPEL. The surprising conclusion is that the exact same process is drawn more than one way using BPMN. Again, this should not be surprising because BPMN is not a complete language, but only a dictionary of notations that can be used in a diagram. As such, it still provides a huge benefit because a reader of the diagram can recognize things and know that the meaning is consistent.

## SETTING THE STAGE

At the dawn of 2008 we find a wide variety of standards that are relevant for BPM. Three of them stand out as part of the BPMN-XPDL-BPEL value chain:

- Business Process Modeling Notation (BPMN) is widely accepted in the 1.0 release from the Object Management Group (OMG), and there is a 1.1 version being privately circulated to address limitations in 1.0 as well as plans for a 2.0 version, which is still in the far future.
- XML Process Definition Language (XPDL) is a widely implemented file format for exchanging process definitions between various process tools. The 2.0 release is two years old. A 2.1 version is ready for public review and will probably be ratified before this article is published.
- Business Process Execution Language (BPEL) is a widely discussed language for implementing web service choreography. The market has generally become aware of the limitations of the format: it focuses on exchange of data between servers, which is fine if that is what you are doing, but it misses the largest part of what most people view as being business-related. There are planned extensions to add human activities, subprocesses, and other missing features to this language.

The holy grail of BPM is a way to define a business process once and be able to deploy and run that process on any BPM server. The benefit of this is obvious: organizations spend tremendous resources on defining their processes, and they would like to be able to store and exchange them in a format that preserves their investment over time. In 2002, BPEL was introduced as a new standard for process engines. Unfortunately, most people did not realize that the intent of BPEL was simply to be able to write programs to send and receive XML data—something better known as Web Service Orchestration. Media hype around BPEL led the public to believe that this could be a common language for all business process needs. BPEL has largely failed to live up to the marketing hype for two reasons. First it is unable to represent human activities directly. The effect of this will be made clear later in this article. The second reason is that since every vendor has to resort to proprietary extensions to do all but the most basic things, the resulting BPEL output is not really portable between vendors. For these and other

reasons, most vendors have stepped away from the language for human BPM uses, while it retains its position as a web service orchestration language.

It then seemed that the only hope for a common language would be BPMN, the modeling notation. Almost all vendors have endorsed BPMN and there are no large competitors. The theory was that if all vendors could support BPMN semantics consistently, we would finally have a way to draw the process once and run it everywhere. However BPMN does not fully define the semantics behind the elements as they are composed together; it was never intended to do this. The designers of BPMN knew that there were many ways to model a business process, and that the success of BPMN would lie in its ability to be used with multiple different process modeling methodologies.

Different approaches to the same problem can yield a different drawing—while still conforming to BPMN—because the approach to drawing a diagram makes implicit assumptions about the capabilities that can be used in that diagram. We will demonstrate that our two different approaches, two different methodologies, and two different sets of assumptions, will get two very different results.

### *The "Automator" Approach*

"Human Process Automator" is a term I use for someone who is taking a manual business process and is attempting to replace humans with computer systems. Many tedious and routine business tasks could be automated by software, freeing up people up to focus on more creative tasks. The ultimate goal is to completely automate the process and remove all manual labor from it.

An Automator focuses on the inputs and outputs of human activity, and writes software to produce the same outputs automatically. BPEL is a language that is strongly oriented to support the needs of an automator in service oriented architecture (SOA). BPEL offers powerful capabilities to send data, receive data, and to transform data. BPEL's native support of XML allows it to work independent of the specific hardware and software platforms it communicates with.

An activity within BPEL is an activity that the computer performs. That activity might be to send information to a person. A later activity might be to collect the response from that person. This naturally can be used to implement a business process, or any other communication-oriented activity, but the BPEL itself is about sending and receiving bytes.

Much of the work on BPMN has been focused on how to make diagrams that can be converted to BPEL. The examples in the BPMN specification are of this flavor, and in some cases even provide the desired BPEL result of the conversion.

### *The "Facilitator" Approach*

"Human Process Facilitator" is a term for someone who diagrams a business process that uses people to do things that cannot be automated. People make decisions based on criteria that are not formally defined. For example, which print advertisement should be used for this month's ad campaign? Or who is the best person to fill this open position? These activities will never be automated. The facilitator is not trying to replace the people in the process, but rather to facilitate people working together.

Since it is people who must perform the tasks, the facilitator needs a process diagram that describes what the people do, not what the computer does. This process diagram is composed of human activities, and a human activity is fundamentally different from a computer activity in a number of ways. People need reminders and ways to prioritize their tasks. People need to have the data presented in a

human readable way. The characteristics of a human activity are expanded in the next section.

Process design patterns also differ for human processes. People do not like to be prodded a number of times in quick succession for the same case, so it is best to group everything that a person needs to do at one time into a single activity instead of creating series of small tasks as you might do for a computer.

A BPM system for human facilitation has built-in capabilities specifically designed to support typical human interaction patterns. If a facilitator did not make use of these capabilities, the resulting process diagrams would be many times more complex because they would have to include all the details of fairly standard interaction patterns. Such increase in complexity would make the diagram less useful for explaining to people what needs to be done.

A person trying to facilitate human business processes has very different goals from someone trying to automate a human process. Both approaches are valid, and different forms of BPM have evolved to handle these different goals. We should not assume that one technology will meet the needs of both, even though there is overlap. The approach detailed in the BPMN specification represents the automator approach. The facilitator approach is further defined below. We use this approach to create a facilitator-oriented process and then compare the results to the sample process using the BPMN specification.

### DEFINITION OF A HUMAN ACTIVITY

Before we talk about a method for facilitating human processes, we need to be clear about what a human activity is. Clearly it is work done by a human. It is not work that is done by a computer on behalf of a user. In order to focus on the human activity, we need to consider those things that facilitate the work as part of the task itself.

Before anyone will perform a task, that person certainly must be informed that the task needs to be done, given the details of the particular case, and have a way to communicate the results of the activity. These are part of any human activity. When modeling human activity, we focus on the work to be done: decide the menu for dinner, wash the dishes, feed the dog, or write a blog entry. Naturally, for a group of people to coordinate these tasks, there must be communication among them, *but a facilitator does not model the communications*. If I want my son to wash the car, clearly I have to tell him that I want him to wash the car, but I don't write that as a separate task on my task list. Instead, it is part of getting the car washed.

It is no surprise that systems designed for facilitating human activities allow you to model the work that is to be done at every step in a process, without being buried in the details of how you will tell that person to do the work, or how the results are collected. Such systems often include customizable ways that each user can decide how they wish to be informed: some users prefer email, others like to receive an SMS message on their phone, etc. A facilitator focuses on the task to be done (e.g. review this document) and lets the system take care of how that user is informed about the task. Similarly, the facilitator knows that an activity may be concluded with a decision (e.g. to either "accept" or "reject" the document), and that may effect the path that the process takes, but does not want to be concerned at the high level with how the system collected that response.

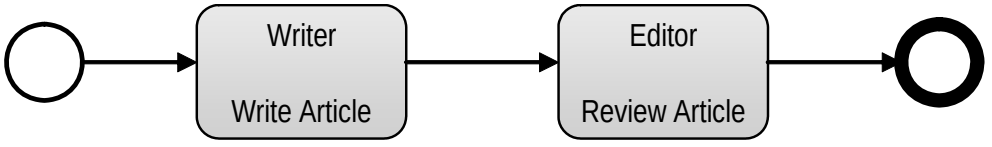
In order to allow a process diagram to be drawn that describes human activity, we assume that the following capabilities are inherently part of a human activity step:

| | |
|---|---|
| ***(a) inform that the task needs to be done*** | This is known as "notification" to the user. Different people have differing requirements on how this notification must be accomplished. In many cases email will be sufficient. Some people have Blackberries and are informed quickly about such notifications. Sometimes SMS or IM is preferable. Some systems use phone paging. Email-based notification is not perfect because the lifespan of the email is different from the task itself: an emergency task may be resolved in a few minutes, but the email will last until the user deletes it. There is a usually mechanism called a WorkList that the user can check at any point in time to see what tasks remain to be performed. |
| ***(b) give the details of the particular case*** | This is the presentation of the information. There is usually a "form" involved, but the term does not to denote a particular technology, but rather any technology that can take information and display it for the user to view and manipulate. There are many options from a Visual Basic or Java programmed UI, to Xforms, to PSP web pages, to any number of options. |
| ***(c) have a way to record the results of the activity*** | There must be a way to indicate that a task is completed. The thing about human activities is that the completion of an activity can be compared to the "declaration" speech act. A declaration is a speech act that in the performance of it, changes the state of a group or organization. Announcing that a document is "approved" or "rejected" is a speech act that is relevant to many people involved in the process. It is quite common that the conclusion of a human activity is also a decision, that is, that the activity can be concluded more than one way and the process should continue on a different route depending on how this activity was concluded. |
| ***(d) have a deadline date for an activity*** | Because people do not work on one thing exclusively until completion and often have many tasks in progress simultaneously, a deadline is a way to make sure that tasks are not dropped too long. Any task might need a deadline. |

| | |
|---|---|
| ***(e) provide reminders about the activity*** | Reminders help in the managing of many tasks, since one task might easily get lost among a number of others. A reminder might be used as a warning that the deadline is approaching. |

A process built out of human activities is dramatically less complex than one that uses more primitive activities. One might say that human activities hide a lot of detail. If your goal is to present the actions that people do, then the detail about how that person is notified that something is to be done is not important. Similarly, if you include all the details of how everyone is notified about every activity, then the entire diagram is likely to be too complex to be readable.

The effect of complexity on readability can best be demonstrated with an extremely simple example of a human process: two human activities that represent a writer writing an article for a reputable publication, and an editor who must review that article to make a decision on whether the article is suitable for publication or not. This is a real process that when used in a real organization provides real value. A facilitator would draw it like this with two human activities:



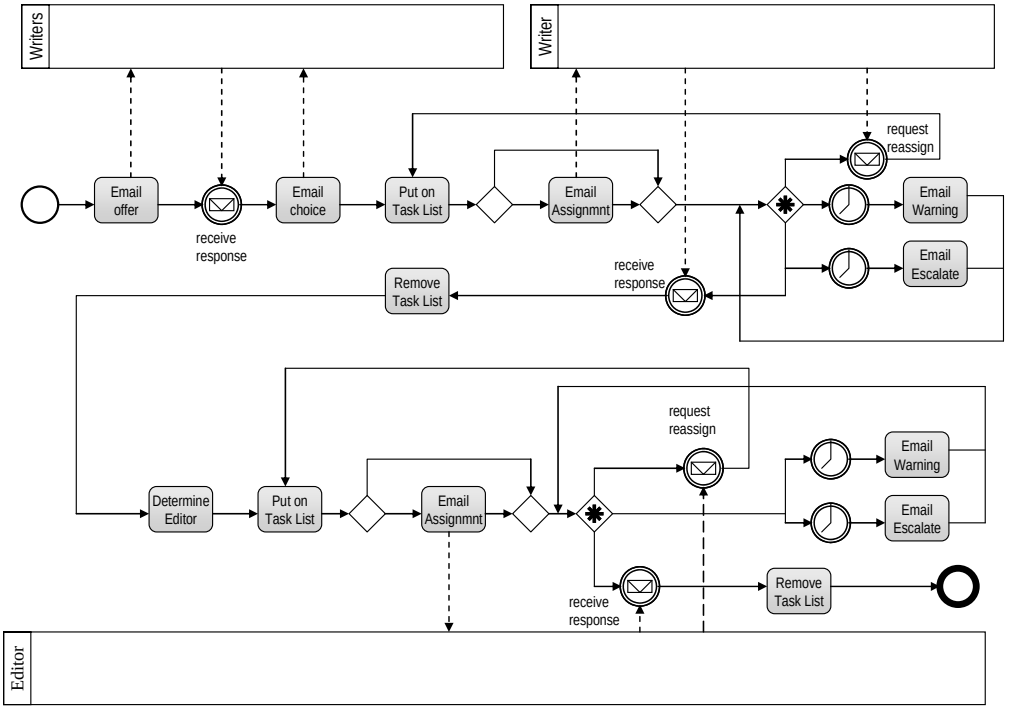

***Process diagram for Human Facilitation***

At the low level, a lot more is going on. If you model all the data that has to be exchanged between systems in order for these people to perform this, you would typically include all of this detail:

- send an email to a writer offering the chance to write an article
- receive an email from the writer agreeing to the task (presumably you ignore any later email message to this effect)
- send email to the editorial team informing them that the task has been accepted
- put the task on a task list that the writer can browse
- send an email (to writer and associates) prompting for the actual writing
  - send reminder email to writer
  - send escalation email to associates if it gets late
- receive the email message back with the document in it
- remove from writer's task list
- determine who should be the editor of this kind of article
- put the task on a task list that the editor can browse
- some editors also want an email message prompting them
  - send reminder email to editor
  - send escalation email to wider group of people if it gets late
- receive the email message back with the publication decision in it
  - OR receive an email saying to reallocate to a different person, loop back a couple steps.
- remove from editor's task list

In a real human facilitation system, there would be a lot more flexibility than this, but this should be enough to show how complexity affects readability. BPEL is a

language that is oriented toward this level of detail: the sending and receiving of bytes. A diagram that is drawn in order to implement the above process using BPEL, might look something like the following diagram.




***Process Diagram for System Automation***

The system programmer would welcome this diagram of the entire detail of exactly what chunks of information are sent at what time. But for the writers and editors, it is much harder to tell exactly what it is that the people are expected to do in the process. In fact this model never actually shows what the people are doing. It shows only what the systems are doing. This is a little like describing the work of an organization in terms of the telephone calls that people make. While you might be able to take a record of all the calls made, and deduce the work of an organization, the records of the calls are not a direct representation of the work. This is, again, fundamentally a tradeoff. If you are a facilitator, and want to model and facilitate human activity, then you want to make a diagram that shows human activities. If you are automating that work by transmitting information around the system using BPEL, then a lower level representation is appropriate.

## A METHOD FOR DESIGNING HUMAN PROCESSES

So keep in mind that a human activity is a description of actual human work to be done, and that each activity is assumed to have (a) notification, (b) information, (c) conclusion, (d) deadline, and (e) reminders built in. The following 9-step method can be used to create a process for human facilitation.

### *Step 1: Identify Human Work*

Start by enumerating the tasks that must be done by people. Ignore for the moment the paper form, the data on the form, or how that form is passed around.

Those who expect this to be a programming exercise may be tripped up by this because of the tendency to focus on the artifacts that help people coordinate their work. We need at this point to look at work itself. These are tasks that depend upon a human skill to do.

There are three reasons why an activity might *not* be able to be automated:

- In some cases there are decisions to be made that cannot be automated and must be made by a person. For example, the determination of whether an article is fit for publication is a task that depends upon recent current events, suitability of the writing style, and the editorial preferences of a particular publication. Another example, the decision about which candidate is the best fit for an open position, is a task that depends upon the personalities of the candidate and the team they would join, as well as an assessment of skills and ability to perform the job. These decisions must be performed by a person because the most relevant attributes may not be able to be expressed in a quantitative way, like political correctness or personality. The rules behind what constitutes acceptable quantities are tacit and are not consciously known by the people who evaluate such rules. But indeed there are people who are very good at making such decisions. This is work that will never be automated.
- The second category is of tasks that might one day be automated, but to do so would require additional prep work that has not been done. For example, you might need someone to enter figures from a financial report that is received either on paper or in an electronic format that is not easily consumable. For the time being, it is simply less expensive to pay someone to do this than it is to pay a programmer to write the code that automatically converts the information. Eventually, these will be automated.
- The third category consists of physical tasks that must be done outside an information system. For example, driving a forklift to load goods from a truck into a place in a warehouse. Or to perform maintenance on a piece of equipment. It might be possible in the far future to automate these tasks with robots, but there are significant barriers to automation due to the physicality of the task. For the time being, we must treat these as human work.

These human tasks are made explicit so that people with the right skills can be identified, or so that people can be trained to do those tasks. Everyone involved in the process needs to know what they do—not just those performing the task—so that everyone gains an understanding of how the tasks they do fit in with what the others are doing. The human tasks need to be described in a way that the people themselves will understand using the specific vocabulary that the people in that organization use. There will normally need to be additional documentation associated that contains detailed information that is useful for training or skills identification.

Avoid including activities that do not involve humans. For example, running a query on a database is something that might be needed at some point in order to support a human task. At this point in the process, you simply assume that the right information is available. There is a later step that defines what information must be available, and a final step that defines how that information is retrieved, but those should be defined at the right point, which is much later in the method.

***Step 2: Determine Activity Conclusions***

Human tasks can be concluded in more than one way. For example, the decision of whether to accept or reject an article for publication will be concluded in two ways: "accept" or "reject." The conclusion of an activity is an explicit part of the activity itself. In many situations, there may be a third conclusion to this example activity, which is something that means more or less, "I am not qualified to make this decision." That is a possible way that an activity might be concluded. Some activities will have acceptable time limits, and may be concluded simply by the passing of time. Each conclusion is given a name.

Conclusions are important communication events. When you model a human process, you are modeling things that need to be communicated to the people involved in the process. Take for example the process of writing a book where many people are involved in various roles, such as writer, reviewer, editor, etc. The writer will at some point declare that the book (a particular draft) is ready for review. While this concludes one phase of writing, more importantly it tells others that they may start their activities of reviewing and editing the current copy. The conclusion of a human activity is most often a speech act known as a "declaration," a statement that in the act of uttering it changes the state of a group of people. Declarations often redefine what many people are expected to be doing. Therefore in a modeled human process the completion of one activity redefines what other people in the process are expected to do.

A conclusion should be considered a distinct conclusion only if it matters to the group. Take for example a task "Answer Question." You might think of the answer to the question, as being the conclusion of the activity, and there are one (or more) answers to every possible question that might be placed. Clearly it is nonsense to consider every possible answer as a possible conclusion of the activity. Conclusions are grouped into sets that affect the flow of the process further on. To be specific, if the flow of the process does not depend at all on whether the task is completed or not, then it is sufficient to say that there is only one conclusion: "done." The president is given the choice to "sign" or "veto" a piece of legislation, and the process continues in different directions depending upon how this task is concluded. However, there is a time limit, and if Congress adjourns before the bill is signed, then this situation is called a "pocket veto." A "pocket veto" is considered to be completely identical to a "veto" as far as the process is concerned, so we would not need a separate conclusion for the pocket veto: the timeout rule would simply be another way to conclude the activity as a normal "veto."

***Step 3: Put the Tasks in Order***

The work and conclusions should be identified without getting overly involved in the sequence of activities. In many cases it is clear that a particular task needs to be done before or after another related task. There will also be branches, and certain tasks that are done only on certain conditions. This is where a diagramming tool becomes useful, but only if it can describe activities at the human level. If one activity must be completed before another, and that other activity can start as soon as the first is completed, then an arrow is drawn between them.

If an activity can be concluded in more than one way, and if each conclusion would cause the process to proceed in a different direction, then there can be an arrow coming out of that activity for each possible conclusion. Clearly, if the point of an activity is to "accept" or "reject" an article for publication, the process that continues after that point will be very different. Because this decision is the very point of the activity, the process becomes easier to read if there is a direct connec-

tion between the activity and the direction that the process goes. Some engines cannot represent this in this way, and instead save the conclusion into a variable which is then tested at a following branch gateway. This is an accepted and common practice, but because the branch is removed from the human task, it is harder to see the direct causal link.

The result is a network diagram of the human activities that must be performed properly set in a process that indicates the conditions and order of the activities.

***Step 4: Determine Performers***

After the tasks and order are identified, one needs to determine who should do the tasks. This is highly dependent upon a particular organization. It also changes from case to case. In some cases, there will be a pool of people who would be qualified to do the task, and anyone from that pool might be picked. What must be determined at this point is what set of rules will be used to determine who should do a particular job. It might be that a person with a particular skill is needed, and if a directory exists that lists all the people with that skill, then the rule is to find those people and pick one. More often the requirement will be that a particular person is chosen because of their responsibility in a particular part of the organization. For example, there may be a person designated to handle requests from a particular customer. Or there may be a person who is designated as handling all the purchase requests for a particular department.

Unfortunately such a rule cannot be specified without specific consideration of the organization that will be using the process. Each organization will have unique organizing principles, some of which are based on historical precedence. Even across a single organization, the rules to determine who does a particular activity may not be consistent. Any organization that grew by mergers of other organizations will have some "special" parts of the organization that are not like other parts. There also needs to be consideration about the specific representation of the organization in an organizational directory. If skills are not tracked, then that cannot be used to determine the person to perform the activity.

There generally will need to be an expression of some sort that can be evaluated in the context of the organizational structure that resolves the assignee of a particular task. This expression will usually make use of pre-existing groups and/or job titles in the organizational directory. It may require new groups or job titles. There may need to be multiple levels of groups, that is, groups that include groups. In some cases it may not be possible to determine *a priori* who will be performing a particular task. In some cases the assignee expression will narrow it down to a group of people, but immediate circumstances (who is available) may determine the final assignee. It might be necessary for the users to self-select for a particular job. There may need to be case-by-case adjustments, since it is not possible to know everything in advance.

***Step 5: Determine the Information in the Process Context***

Here you specify a schema or a set of schemas that carry the information context within which all the activities take place. If the process is for a customer to open a bank account, then there is specific information that needs to be carried for that process, such as the customer name, address, and references to other accounts or credit history. The context schema needs to be a superset of all information needed for every activity. For example, if there is an activity to assess the property value of a house, then clearly the details about the home address, prior sales information, and various reports about the locale are necessary to perform this activity. If one activity produces a result which is necessary in a later activity, such

as the assessed value of a house, then there must be a variable that will hold that information between activities. By considering the information requirements of every activity in the process, you can compile a complete context schema required by the process.

The information content will be modeled differently by different implementation engines. For some there is a single schema for the context that is shared by all activities (effectively a union of all schemas required by the individual activities). Others have a collection of schemas that are transformed back and forth through the process. Either way, the idea at this point is to identify the information requirements of the entire process.

***Step 6: Define Access to Information at Each Activity***

At some points in the process, certain parts (variables) within the shared context can be read and updated. At other points information can be read, but not updated. There are also points in the process where information is completely hidden because it is either not yet specified at that point in the process, or not relevant to that particular activity.

***Step 7: Determine Timeouts***

An activity may have a requirement to be performed in a particular time period. What happens when that time period is exceeded? Does the process continue without the activity being completed, or does the process "fail" and go down a different path. There may be reminders that are additional notifications to the user that the task has not yet been completed. There may also be escalation to other people or management if the task is nearing the deadline without being completed. At this point for each activity, all time-dependent behaviors should be considered. Some tasks may have no time dependency at all, and may be allowed to remain uncompleted indefinitely.

We know that time equals money; so it is worth considering at this point the cost of every activity, as well as the cost to the organization of either delaying the activity, or not performing that activity. If you are simulating the execution of the process, these costs entered into the model can be accumulated across a simulation run in order to guide the further design of the process.

***Step 8: Design the Presentation of the Information***

This puts a face on the context information, mapping the schema to a visual presentation. This presentation might be specific to a given activity, or might be the same presentation over the entire process.

Humans don't read XML directly. Instead, the information has to be displayed in a way that is meaningful to the user. To be effective, the display should be organized for ease of use. Some of the information may be keys or links to other information, and the display should provide an easy way to access those external sources of information.

Technology to present the information is often described as "forms" in the BPM community, but you should keep in mind that any technology that can take data and generate a user interface can be used. The choice will depend on many factors outside the BPM system. Some organizations will choose Visual Basic or Java Swing because they have programmers experienced in these areas. Some might choose PHP or other web techniques. They might have powerful forms software designed specifically for this purpose. The process definition method should not get bogged down at this point in the specific requirements of the technology to be

used. Instead, this step should focus on the look and feel of the displayed information.

***Step 9: Integrate with Information Services***

This is where the information needed in a process can be picked up from various sources and sent to various destinations. I use the term "service" in the generic sense of an SOA. This might be through web service calls or any other means to access other service types. The point simply is that there is a human activity that needs a particular piece of information, and so this is where you specify how that information will be retrieved for that human user.

It is this step where you finally consider how data is sent and received between computers. Many process designers start by considering how data is transferred through the system, and it leads them to a communications-centric view of the work. It can lead to activities that are optimized for computer communications, instead of being optimized for human work. Since human costs far outweigh computing resource costs in most business processes, it is important to start with the human tasks, and then work down to the integration tasks.

To access information from a web service, some of the process context information will need to be transformed appropriately into that XML that is needed as input to a web service. The resulting XML may need to be similarly transformed to be put back into the process context. For example, in an account application, the process may need to access a "credit rating" service to retrieve the applicant's credit rating for consideration in the application process.

Services are used not only for retrieval of information, but also for how the results of the human tasks will be sent out to destinations outside of the people directly involved in the process. For example, if the decision is made to approve a loan of a particular amount to a customer, then there are various parties that may need to be informed about this decision (e.g. by email) and there would also be calls to services to actually set up the account and initiate the sending of a contract to the parties involved.

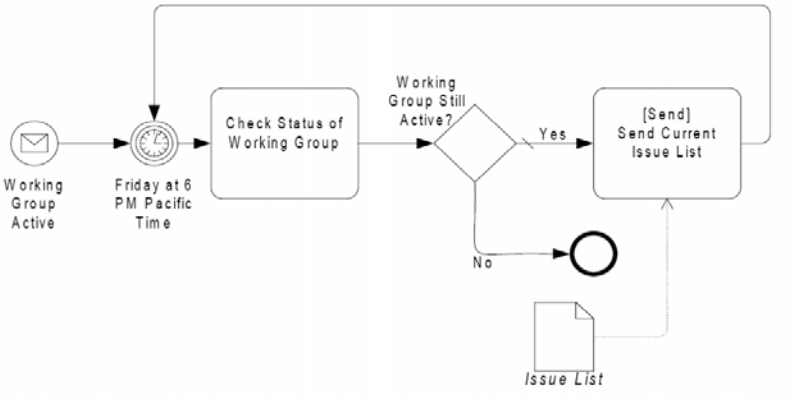
There it is; nine steps that lead to a model of a human process. Not a complete methodology by any means, but still useful. The steps are repeated iteratively, with reviews at various points. Usually after each step there is some segment of the organization that is interested in reviewing the progress. It is also true that later steps will turn up details that were left out of earlier steps, and so there is some iteration through the method multiple times. A good system will allow simplistic execution of the process before you are complete, so that you can try out the process along the way. After Step 3 (Put the Tasks in Order), you should be able to run simulations of the process in order to gain confidence on the correctness of the process. After the process is implemented and deployed, you can collect statistics on how well it is running and cycle back through these to improve the process. We call it "Business Process Management" because you are never completely finished designing the process. This method is repeated as long as the process can be improved, and there are always new ideas on how to improve the process or to respond to external changes.

### EMAIL VOTING PROCESS—AUTOMATOR APPROACH

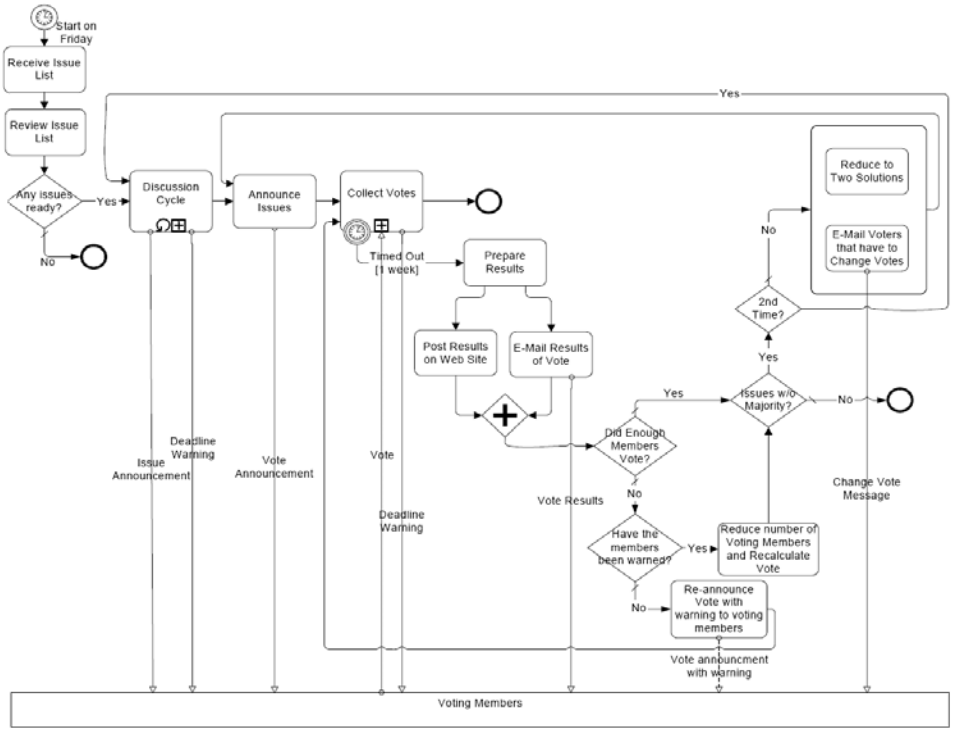
In order to show how a human facilitation process would look different from a BPEL-oriented process, we need an example process that is done both ways. The BPMN 1.0 specification includes such a sample process. It is known as the Email Voting process. Bruce Silver, arguably the most respected analyst in the BPM

field, has suggested that this be used as an example process to test interoperability between different process diagramming tools. One point in favor of this is that it is fairly well fleshed out and documented. Also, it is a real process that would be reasonable to use in real life.

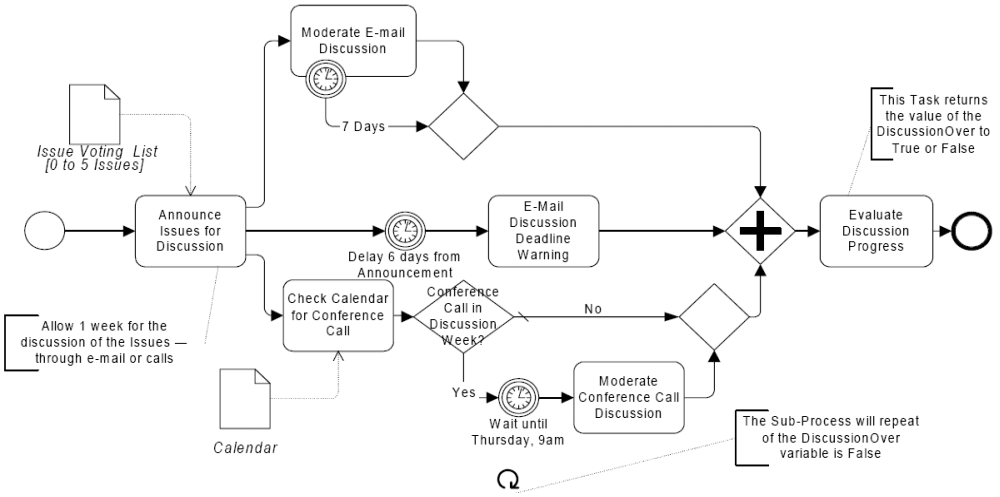
The version in the specification was drawn explicitly to be converted to BPEL for implementation, so the process is designed around things that can be done in BPEL. As BPEL is a language for automating processes, the form of the process in the specification is an excellent example of an automator oriented diagram. Space restrictions prevent us including everything, so please refer to the BPMN specification itself for a complete description of the process. The process diagrams are included here as a more convenient reminder of the processes:



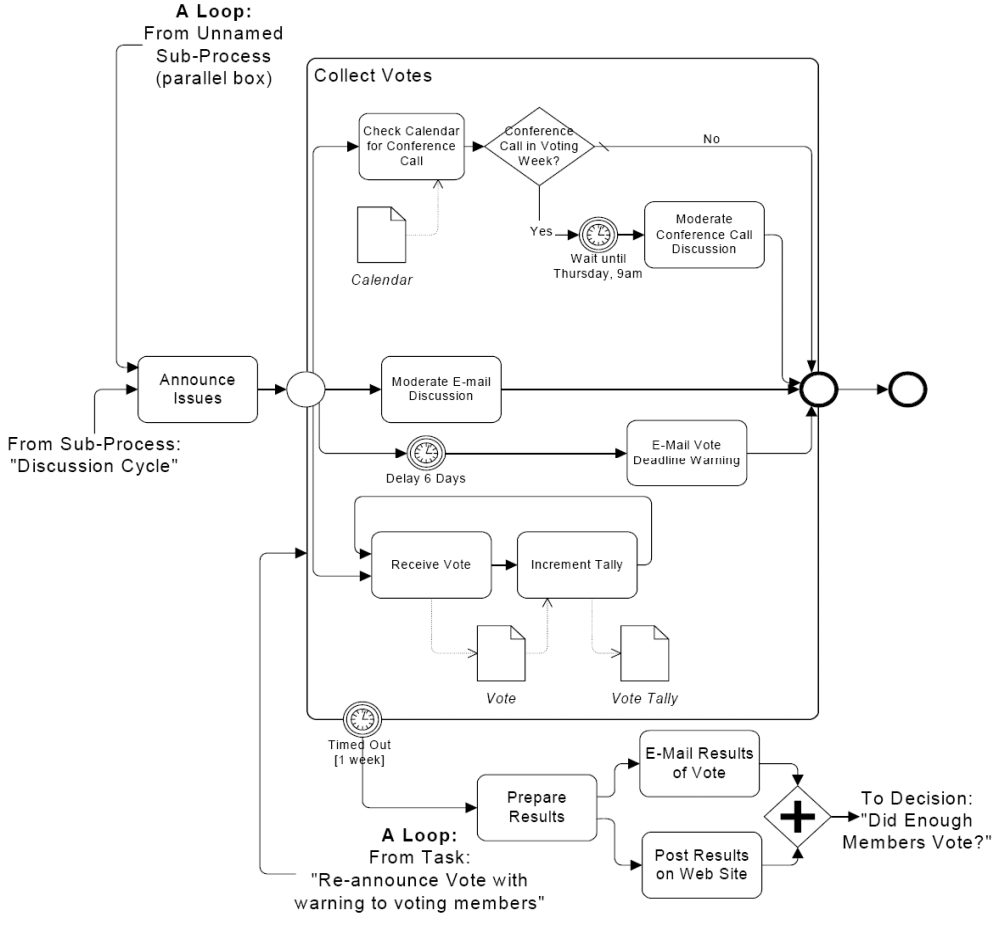

***The Starter Process***




***Top Level process, there are two subprocesses***




***First Subprocess***




***Second Subprocess***

## EMAIL VOTING PROCESS—FACILITATOR APPROACH

As I set out to implement this process, it struck me how dramatically different the process would be drawn if you had an implementation engine that supported

human activities directly. Many of the things that are broken out into separate activities would be combined into a single activity. Even at the basic level, the model would be different because you would be modeling the human activities instead of the flow of data back and forth.

The first step is to get a list of the actual human activities that appear in the process. To do this we fly up to the 40,000 foot level and look down, ignoring as much as possible the details that came into the process due to the need to execute it in BPEL. We start with the activities that have to be done by people, because they require human intelligence and really cannot be automated.

| | |
|---|---|
| ***Activity 1: choose the issues for this week*** | The scenario assumes that there is an open ended list of possible issues that are being continually added to. It is not possible to talk about all issues every week, so someone must pick the most important issues. It is not possible to automate the determination of what the most "important" issues are, so this task of picking 1 to 5 issues for discussion and voting this week must be done by a person. There may not be enough issues this week, or it may be a holiday week, or any number of other possible reasons not to discuss issues this week, so the issue manager must choose whether there are "Issues" or "No Issues" this week. |
| ***Activity 2: discuss issues*** | There is a period of time within which people are allowed to discuss the issue, and that means having access to the issue descriptions, access to comments that others have made, and the ability to make comments on the issues. This is performed simultaneously by a large number of committee members. |
| ***Activity 3: determine if discussion is over*** | The issue list manager is asked in this process to take a look at the results of the current discussion, and decide whether the discussion needs to continue or not. Since this is a decision that is not automatable, we have to have the issue manager do it. There are two choices: "Continue the Discussion" or "Finish the Discussion." |
| ***Activity 4: vote on the resolutions to the issues*** | Mainly just recording the results of the activity. In this step, we can make use of a parallel activity called a "Voting Node" to allow everyone to be informed about the vote and to collect their responses in a convenient way. |

| | |
|---|---|
| ***Activity 5: remedy voting*** | This is a human activity in the case that voting "fails." Voting can fail when not enough people vote. The original BPEL-oriented process had a number of steps, most automated, to handle this situation. When considering the human tasks, an analysis of the process leads to the conclusion that a large number of the possible paths can be automated, but a few of the paths require a person to address the situation by either reducing the required number of voters or reducing the number of options being voted on. The key here is that in those situations a human must be brought in. What is important is that the person be notified that the voting has not worked and be given a couple of options to remedy the situation. So for the purpose of modeling the human process and for making the decision of how to proceed, I have collapsed this to a single activity node for a person to perform. The automated aspects are ignored at this point. |

Then we have human activities that are necessary because the technology is somewhat limited, and not completely automated, and so there are some tasks which people do manage and maintain the systems

| | |
|---|---|
| ***Activity 6: moderate the online discussion*** | Unfortunately, the discussion is not completely self-moderating, so someone must have special access to delete spam or inappropriate messages, and to help keep the discussion going. |

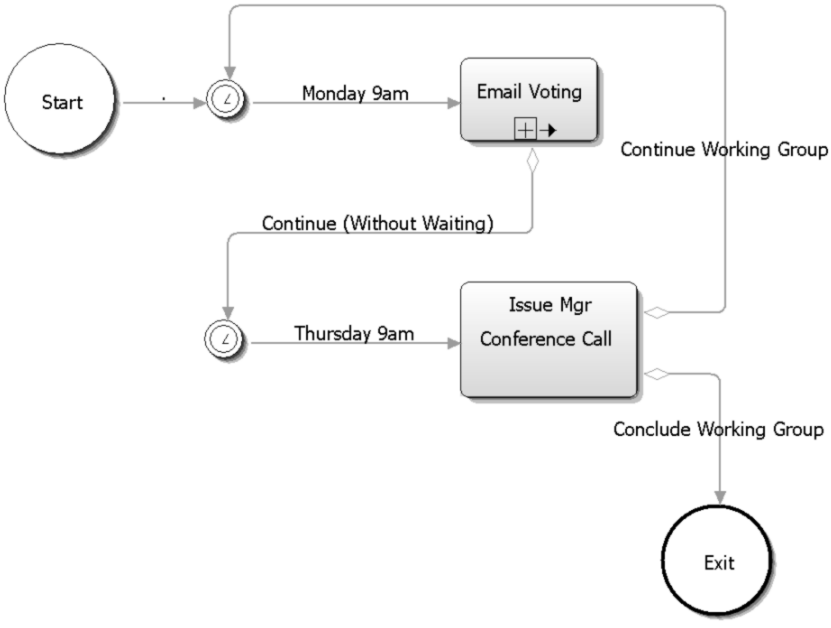
| | |
|---|---|
| ***Activity 7: hold or attend conference call*** | The discussion is not completely an online discussion. Conference calls are used to help people come to a common understanding of the issues and resolutions. Someone needs to set up the conference call and moderate it while it is happening. |

These seven activities account for everything that is done by people in the process. There are only two roles in this process. The issue manager is the person managing the issue process. This could be multiple people, but only one of them would act at a time. The other role is the committee, which is everyone involved in discussing and voting on issues. The committee members are responsible for discussing issues and voting, while the issue manager is responsible for all other activities in the process.

The email voting process actually consists of two processes: a single starter process that loops and starts an instance of the actual issue voting process every week. The issue voting process then has two subprocesses. Because we have only seven human activities, I decided to keep it simpler and not use the subprocesses, and to do everything in the two main processes. If you wanted to, however, you could use subprocesses to group things differently.

In doing the analysis, I became aware of a strange behavior of the original process as written. You could very easily be in the position of being asked to run multiple

conference calls at the same time. Remember that the issue handling process is started every week, which drops into a subprocess for decision handling and another for voting. Each of those subprocesses has logic for determining if there will be a conference call during that time period, and to trigger an activity for handling the conference call. The discussion subprocess may loop, potentially for multiple weeks. It is possible that you will have multiple instances running at a time, and if this is the case, each instance will prompt for a conference call. The issue manager might manually avoid having multiple instances running at the same time, by deciding not to discuss anything in the new process instance as it is started, but this rather begs the whole question of why there is the possibility of multiple issue resolution processes in the first place; why not simply have the single starter process do all the work? Finally I settled on the idea that issues are grouped, and it makes sense to have the possibility of multiple issue groups being discussed and/or voted at the same time. The conference call on the other hand is scheduled every Thursday, and takes place regardless of the number of issue groups currently and therefore should be associated with the single starter process, and not the instances of the issue voting process.

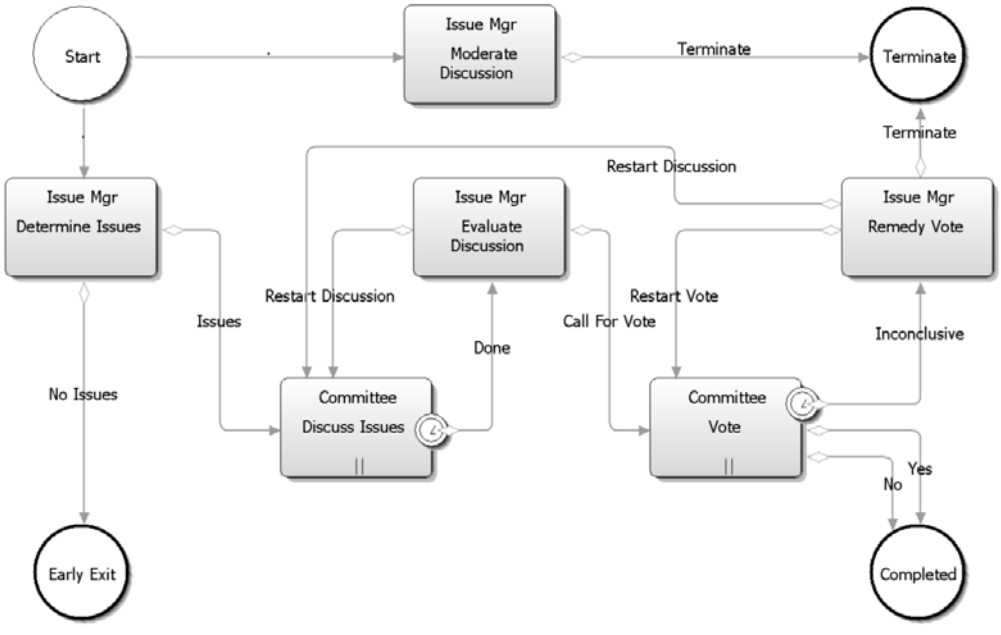



***Email Voting Starter Process***

There is a single instance of this process, which loops every week. There are timer delay nodes that cause things to happen at specified times. At 9:00 a.m. Monday morning, an instance of the issue list process is started. This is done with an "asynchronous" subprocess node; this is a node that starts a subprocess, but continues without waiting for that process to finish. The ability to start a subprocess and not wait for it is commonly available on human process engines, and is a standard part of the XPDL file format. This is one clear area where the capabilities of the engine will certainly affect how you draw the process. If you think about it, the purpose of this starter process is to create instances every week of the email voting process, but because in BPEL there is no operation for express-

ing this, the original process would "send the issue list" and the "receipt of the issue list" would cause the creation of the process. This is quite strange when you consider that the issue list itself is probably on a server that is already accessible by everyone, and does not need actually to be "sent" anywhere! The sending and receiving of the issue list is an artifact created just to get around the problem that BPEL does not have an operation to create a subprocess! This is also partially because BPEL is not modeling the work that people do, but only modeling the sending and receiving of data. Of course, it is possible to model the whole world in terms of sending and receiving bytes, but it is far clearer and easier to read a process diagram with a node that creates the subprocess directly.

Then, on Thursday at 9:00 a.m. there is an activity to hold a conference call. This is assigned to the issue manager. I embellished the process a little bit to also give that manager a choice of whether to conclude the working group or not. The original process had a branch node that checked something to see if the working group was still active, but then someone would have had to have taken some action somewhere to indicate that the working group was completed. Drawing it this way ties both the human indication that the working group is finished together with the completion of the process. Every week, the conference call task will be concluded with the choice "Continue Working Group," which requires no effort beyond simply saying that the conference call is done. But, when the time comes that the working group is no longer active and there is no need to ever have another conference call or another issue vote, then simply choosing "Conclude Working Group" provides an easy way to complete the process without using the administrator's console.




***The Issue Email Voting Process***

This process starts two activities from the start. "Moderate Discussion" is designed to remain active the entire time the process is running. Most of the time the process will be in the Discuss Issues step or the Vote step, so really this step allows a place where the issue manager can moderate both discussion and voting.

I embellished it a little to add a "Terminate" option that would shut down the process before discussion or voting was finished.

The main process is the lower five activities. First the issue manager decides the issues to be discussed. Second, the issues are discussed by the committee. Then the issue manager decides if the issues have been discussed enough and completes that by either "Restart Discussion" or "Call for Vote." If it is the latter, the committee is asked to vote. If the vote is not conclusive, then the issue manager must Remedy Vote, where he will be able to reduce the number of voters or the number of resolutions to vote on, and choose either to Terminate the process, go back to Discussion, or go back to Vote.

How does the "Vote" work? Here we cheat a little bit by using a capability of the engine that presents the vote options to a number of people simultaneously. It gives the options of "Yes" and "No" to all committee members. People choose one, and the process engine tallies the results. On the outbound arrows, you can specify how many votes, or what percentage of the votes, are necessary to consider an option successful or not. As soon as the success criterion is met, the voting activity is concluded. There is also a built-in timeout so that if a vote is not concluded in a period of time, then the activity can be concluded down a third arrow which leads to the Remedy Vote activity. I know this seems like cheating to pull something like this out of the hat, but I did not make up this process. Voting steps are relatively common in the workplace, and this is a real node type available in a number of process engines designed for human facilitation. The voting node is a clear example of how a BPMN drawing is dependent upon the underlying technology.

How are people notified about what to discuss? How are they told what to vote on? When you assign a human activity to a person or group of people, that activity automatically includes a notification mechanism that tells them what they are to do, and includes information about the specific case. Simply assigning the "Discussion" node to the Committee means that everyone in the committee is automatically told what it is they are to discuss at the time that the discussion is to start. Simply assigning the vote node to the Committee means that they will be notified what the voting options are at the time they are to vote, and their votes are automatically collected and tallied.

What about the nodes to send reminders? Again, this does not need to be explicitly modeled because every human activity automatically includes (a) notification, (b) information, (c) conclusion, (d) deadline, and (e) reminders. We just set up the discussion activity to have a reminder after six days. Similarly on the voting node, we have a reminder after six days. That reminder on the voting node can be considered a "warning" since if the vote has been concluded, then the reminder will not be set. The BPEL-oriented process had to stop the voting, and then go back and restart the voting after the warning, but that is not necessary when the voting node can send reminders without stopping. Even the nodes for the issue manager can have warnings and reminders if the issue manager goes for a time without taking care of the task.

What is this human process good for? By showing the process, you see what everyone is expected to do. At each step in the process, a person is told that something must be accomplished, and given a set of choices. The diagram is oriented around what it is that people do, not what bytes are sent and received. As such, it is useful for explaining what will happen next in the process depending upon which choices are made. This is useful for training people, and it is useful to dis-

play the current running status of the process instance to people. But some will point out that it is not complete; it does not contain a representation of all the automated actions. This is true, but no process map is entirely complete. For example, the complete list of process variables is not displayed in either version of the process diagram.

## DISCUSSION AND CONCLUSION

This article has demonstrated that using two different methodologies to define a single given process has produced two very different process diagrams. Neither diagram is more correct than the other, but the diagrams are useful for different things. Both diagrams use BPMN. The key point is:

> *The way that you model a given process in BPMN depends upon the methodology you use, and the assumptions you make about the technology that will support the process.*

There exists a strong desire for a *lingua-franca* of business processes. It would be great to be able to draw a diagram of a process on any drawing tool, and then execute it in any process engine. It would be great if BPMN could provide this, but it cannot, and indeed should not. BPMN does not define the full semantics of the process, and was designed to be used in more than one methodology.

> *A given BPMN diagram can execute only on a process engine that corresponds to the assumptions that are built into the diagram.*

"What good then is BPMN?" some will ask. BPMN is appropriately named as a notation, and it provides exactly that. BPMN provides a dictionary of symbols with associated meaning. To the extent that different vendors use the symbols for the same meanings, there is great value in this. The one who benefits is the business person who takes the time to learn the basic symbols in order to be able to read the process diagrams.

There is an analogy that one can use in the symbols of geographic maps. A set of symbols has evolved over time with clear meanings on maps: a school, a church, or an airport. The reader who understands those symbols is in a better position to interpret the map. Because most people have learned the associations between the symbols and meanings, maps are easier to read in direct proportion to the extent that a producer of a map sticks to the standard symbols. A map that makes use of proprietary symbols would benefit neither the producer nor the readers. So standard symbols are a benefit, but that is not to say that all maps are the same and have the same meaning. You might make a map to get someone from point A to point B, but that map will look very different depending upon whether the person is walking, riding a subway, or taking an airplane. There is no question that a walking map of a city looks completely different from a subway map of the same city. The map incorporates implicit assumptions about the people who will use the map. And while you might make an infinitely detailed universal map of all modes of transport, you will find that that map will be far less useful than the mode-specific maps.

It is the same with BPM. The business process diagram embeds implicit assumptions about the audience and about the capabilities of the process engine it will be running on. We have seen specifically that the approach of the process "automator" will lead to a process that focuses on how information is exchanged between

systems, and result in a process that describes this. If your goal is to convert the process to BPEL in order to run it, then that process will tend to be designed in that way. We also have seen how a "facilitator" approach will focus on what the people do, and less on what the computers do, and result in an entirely different diagram. Again, if you can assume an engine that supports human activities directly with powerful concepts like "voting activities," then you are freed from most of the tedious details and can focus more on what the people are doing.

I should mention briefly here that there is a move to make BPEL more suitable for processes that facilitate human work by extending the specification to include a new human activity and by defining standard worklist services which can be invoked through standard BPEL means. This work is still very preliminary at the time of writing this, so it is impossible to say where it will lead. If the BPEL standards group was to add complete support for human activities as described above, it would be improper to categorize BPEL as an automation-only language, and we must also assume in that case that BPEL would be useful for facilitation diagrams. This would make BPEL far more useful, but would not change the message of this article. It would still be the case that a diagram drawn for use by humans which focuses on human tasks, would look different from a diagram drawn for use by a system programmer which shows all the data exchange.

In this article I contrast two methodologies, but I don't mean to imply that there are only two. There is, presumably, any number of such. BPMN should be used whenever it makes sense to be used to give common meaning to such diagrams. It should be further expected that BPMN will grow over time to become more suitable for more different situations. While many vendors have implemented BPMN, they almost always include proprietary extensions suitable to their particular approach. It is a benefit to all of us for common meanings not yet specified by BPMN, to be included into the standard in the future.

Though BPMN is not a universal language for describing business processes, it is still a great notation standard and is as relevant as ever.

## METHOD FOR DESIGNING HUMAN PROCESSES (SUMMARY)

***Step 1: Identify Human Work***
***Step 2: Determine Activity Conclusions***
***Step 3: Put the Tasks in Order***
***Step 4: Determine Performers***
***Step 5: Determine the Information in the Process Context***
***Step 6: Define Access to Information at Each Activity***
***Step 7: Determine Timeouts***
***Step 8: Design the Presentation of the Information.***
***Step 9: Integrate with Information Services***