Human-Readable BPMN Diagrams

Refactoring OMG's E-Mail Voting Example

Thomas Allweyer

V 1.1

1 The E-Mail Voting Process Model

The Object Management Group (OMG) has published a useful non-normative document for BPMN modelers [BPMNExamples]. It is called "BPMN 2.0 by Example" and can be downloaded at: http://www.omg.org/spec/BPMN/2.0/examples/PDF/10-06-02.pdf

While the specification of the BPMN standard describes the BPMN diagrams, elements, and their meanings, the examples document provides suggestions of how to use BPMN for modeling real processes. The reader can get valuable insights and hints for his own modeling practice.

This paper discusses one of the models, the E-Mail Voting Example [BPMNExamples, p. 35]. The E-Mail Voting Example describes how a distributed working group discusses issues and votes on them by e-mail. This process was used during the development of BPMN. The authors claim that "This process is small, but fairly complex […], and it will help illustrate that BPMN can handle simple and unusual business processes and still be easily understandable for readers of the Diagram" [BPMNExamples, p. 35].

The reader is recommended to download the examples document and to have a look at the diagram on page 36. The process as such may indeed be small, but the diagram is not small. It covers an entire page, and when it is printed out, the font size is about 4pt which makes it rather hard to read. The diagram indeed looks rather complex. So, if this is a small process, it is easy to guess the dimensions of a large process's diagram.

Is this diagram easily understandable? Judge for yourself. If you take the effort to follow the sequence flow step by step you will be able to sort out how the process actually works. So yes, it is understandable for experts – but it is certainly not easy to understand. I would not dare to give this diagram to any business expert. This picture is not worth a thousand words, but it requires a thousand words of explanation in order to understand it. Actually, the explanation in the document contains a bit more than 1.500 words.

It may be unfair to criticize this model, because its main purpose is to illustrate the application of many different BPMN concepts. However, it is still an interesting exercise to re-design this model in

order to make it more comprehensible. How should the model look like if its main purpose was to explain the process to one of the participants, maybe to a newly elected workgroup coordinator who needs to carry out that process in the future?

It is recommended to print out the original model from the OMG paper and refer to it when reading this paper.

2 Partitioning the Process

In order to cope with a large model, it can be divided into sub-processes. When the e-mail voting process is analyzed, four basic phases can be identified:

1. A discussion of the issues by e-mail and by a conference call
2. The submission of the votes by the voting members
3. The dissemination of the voting results
4. An evaluation of the voting results and a decision about how to proceed. For example, if not enough members have voted, there will be a second chance for submitting the missing votes.

So the process could be split into four sub-processes. However, the dissemination of voting results would be a rather small process, and the decisions in the fourth phase can also be modeled in a more compact way. There are also strong connections between phase 2 and phase 4, since there are two back-loops from phase 4 to phase 2.

For these reasons it may be a good idea to define only two sub-processes: “Discussion” and “Voting”.

Usually, BPMN sub-processes are part of a higher-level process. A first draft of such a higher level process is shown in Figure 1. This is a simple diagram of the basic structure of the process: First, there is a discussion, then the voting.

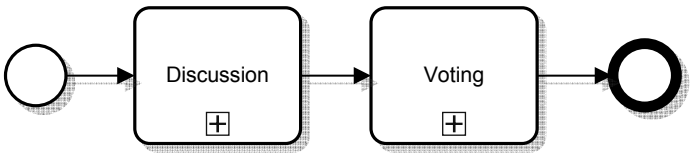

Figure 1: High-level diagram with inaccurate sequence flow

However, the diagram in Figure 1 is a little bit oversimplified. A look at the original diagram reveals that the process is not triggered by a “none” start-event, but by a timer event. And there is also a back-loop to the discussion, because it is possible that the voting cannot be completed, but the discussion needs to be re-opened. In order to reflect this, the top-level diagram needs to be modified, as in Figure 2.

Now the diagram is correct, but it is cluttered with details which are not really appropriate for a high-level diagram. Is it important that the process is started every Friday? No, this is a detail. If the process would be started every Thursday instead, it would not be important for the big picture. Is the back-loop important? Probably not, since a detailed analysis of the original diagram’s sequence flow shows that the discussion is re-opened only in exceptional cases.

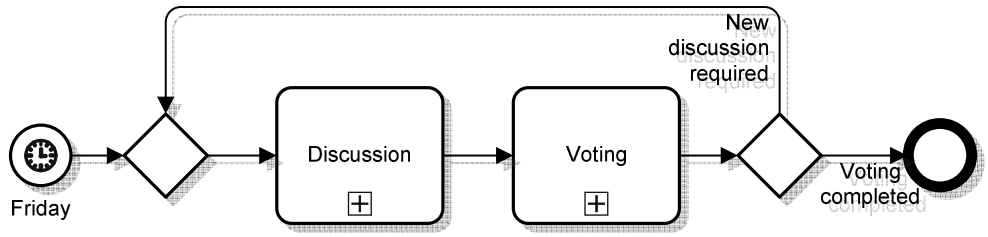

Figure 2: High-level diagram with accurate sequence flow, but with too many details

So, how can you model the simple fact that the process consists of a discussion process followed by a voting process? If you strictly follow the BPMN specification, you cannot.

One possibility is to relax the BPMN rules a little bit for high-level diagrams and simply use the diagram from Figure 1. If you decide to do so, you should define in your modeling conventions at which points you relax the BPMN rules.

The other possibility is not to use BPMN for this purpose, but another notation or free-form diagram, as in Figure 3. Here, arrow symbols have been used which often can be found in value-chain diagrams or process landscapes.

Figure 3: Non-BPMN high-level diagram

3 Simplifying the Process Diagrams

A first attempt to make the two sub-process's diagrams more readable is shown in Figure 4 and Figure 6. First of all, the data-objects and the message-flows have been removed. In order to understand what is going on in the process, these details are not required. They are only confusing. Since there are only two roles – the workgroup coordinator who carries out the process, and the voting members – it is rather clear from the activities' descriptions which e-mails will be exchanged. Therefore, the message flows do not add valuable information for understanding the process. Nevertheless, a simplified diagram with message flow will be shown later on as an alternative representation.

The data objects, on the other hand, are more interesting. They represent the various documents and lists which are used and created in this process. The workgroup coordinator can seen from the original model which documents he requires for each activity, and where the resulting information is used. In the first step, however, it is easier to understand the sequence flow without the data flows. In a second step such additional information may be added to the model. A variant of the model with data flow will also be shown later on.

In the original diagram, the tasks have been marked with icons as "user tasks" or "send tasks". These icons have been removed for two reasons. The first reason is that the definitions of these task types in the BPMN specification are rather restricted: A user task is defined as "a typical 'workflow' task where a human performer performs the task with the assistance of a software application and is

scheduled through a task list manager of some sort" [BPMSpec, p. 167]. Since this process is apparently not supported by a workflow management system, the tasks are not strictly user tasks. A send tasks, on the other hand, only has the purpose of assigning data to a message and sending that message – but nothing else [BPMNSpec, p. 163 and 444]. In this process, however, the tasks with envelope icons apparently require the workgroup coordinator to combine information from various lists with predefined text, before an e-mail is sent. Due to these additional steps, the tasks are not pure send tasks.

Of course, it would be possible to relax the strict task type definitions and simply use the icons for identifying tasks which consist mainly of user actions or which have the main objective of sending out a message.

The second and main reason for omitting the icons is that they do not provide much help for understanding the process. All of the activities involve the user, and the activities' names or descriptions clearly indicate whether messages are sent.

4 First Part: The Discussion Process

Just as in the original diagram, the process starts with a timer event. The first original activity, "Review issue list", has been removed, since it is rather trivial. It is not necessary to emphasize that the following decision "Issues ready?" requires a look into the issue list. This decision is followed by an exclusive gateway. It merges the sequence flow with two backwards looping sequence flows.

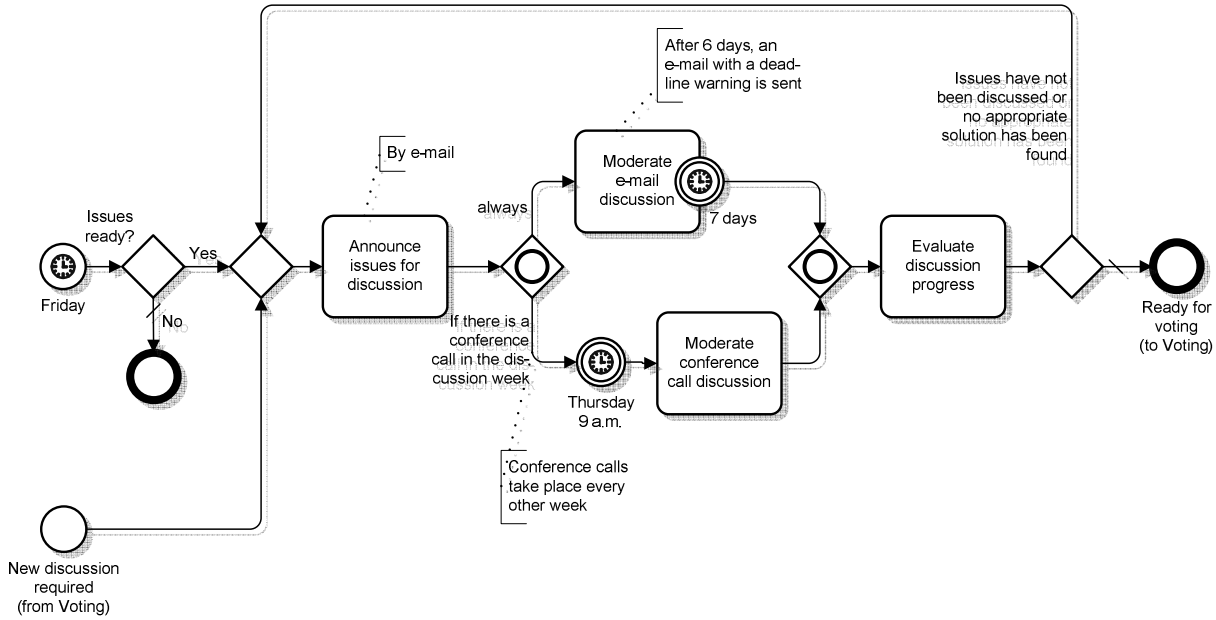

Figure 4: Discussion process, first version

In the original diagram, the discussion cycle is shown as a sub-process with a loop indicator. Since the loop is more clearly visible when modeled with a gateway and a backwards looping sequence flow, this alternative structure has been chosen. This is also more consistent with the representation of the other loops in the voting process.

The e-mail discussion and the conference call discussion have been reduced to one task each. The details concerning the e-mail deadline warning and how to determine whether there is a conference

call in the discussion week, have been removed from the model. They are explained in annotations. In this way, the model structure becomes clearer. It is very easy to see what is going on: An e-mail discussion and – in some cases – a conference call discussion. It would be possible to model the omitted details in sub-processes. However, in this case the annotations should be sufficient, so there is no need for the reader to figure out this information from additional models.

Originally, the fact that there is always an e-mail discussion and only sometimes a conference call, has been modeled with parallel sequence flows. One of these parallel flows is split into two alternative flows: One with the conference call, the other without any activity. This rather complex solution for a simple problem has been remodeled with an inclusive gateway. The upper sequence flow is labeled “always”, indicating that the condition of this sequence flow is always true and the e-mail discussion takes place in any case.

After evaluating the discussion progress, either the loop is repeated, or the voting process can start. When the discussion process reaches the end event is reached, the start event of the voting process is triggered. There is no obvious way of modeling this connection between processes. If the two processes are regarded as two independent processes, it would be possible to use signal or message events. Both types of events are not entirely satisfying. Signals represent broadcasting behavior: Any process could react to that signal. This is not intended here, since only one specific process should be addressed. This can be expressed by sending a message, instead. However, the sending of messages between internal processes rather than between different partners is not very intuitive.

Note that link events cannot be used in this case, since we have two separate processes. Link events are intermediate events which can only be used within one process. If the undivided original process just had been printed onto two different pages, then link events could have been used for providing the connection between the two pages.

If the two processes are not independent processes, but sub-processes of a high-level process (as the one in Figure 2), then the connection is provided by the parent process. When the first sub-process is finished, the parent process moves the control to the second sub-process and thus triggers the start event in this sub-process. Within the sub-processes, the connection to the next sub-process is usually not visible. It is therefore indicated in the label of the end event (“to Voting”).

Note that the discussion process also has a second start event: “New discussion required (from Voting)”. This label indicates that this start event is triggered from the backwards looping sequence flow originating from the voting process. This loop is also shown in Figure 2.

Unfortunately, the connection between the parent process of Figure 2 and the discussion sub-process is still not modeled correctly. When a sub-process is activated by its parent process, the sub-process’s ‘none’ start event is triggered. So every Friday, the parent process would activate the discussion sub-process, but the process would not start with the timer event, but with the ‘none’ start event, so that the decision “Issues ready?” would be omitted. The sub-process’s timer event “Friday” is independent from the parent process’s timer event. Therefore, the sub-process would be started a second time. This second process instance would not be in the context of the parent process, so that it would finish without triggering the voting process.

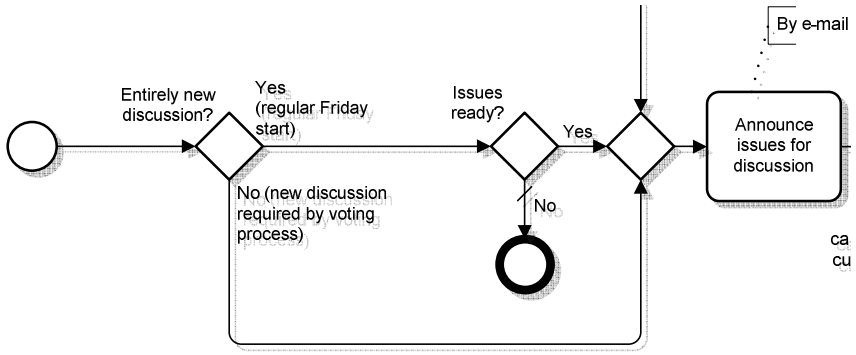

Figure 5: Alternative beginning of the discussion process with only one start event. This makes the model consistent with the parent process of Figure 2.

The correct behavior would be expressed by one single 'none' start event in the sub-process (Figure 5). This would be followed by an exclusive gateway. This gateway would split the flow based on whether the sub-process was triggered via the sequence flow from the start ("regular Friday start"), or via the sequence flow from the backwards loop ("new discussion required by voting process").

Since Figure 5 is less self-explanatory, and since the business reader may not be shown Figure 2 (but rather Figure 3), Figure 4 still seems the better solution.

5 Second Part: The Voting Process

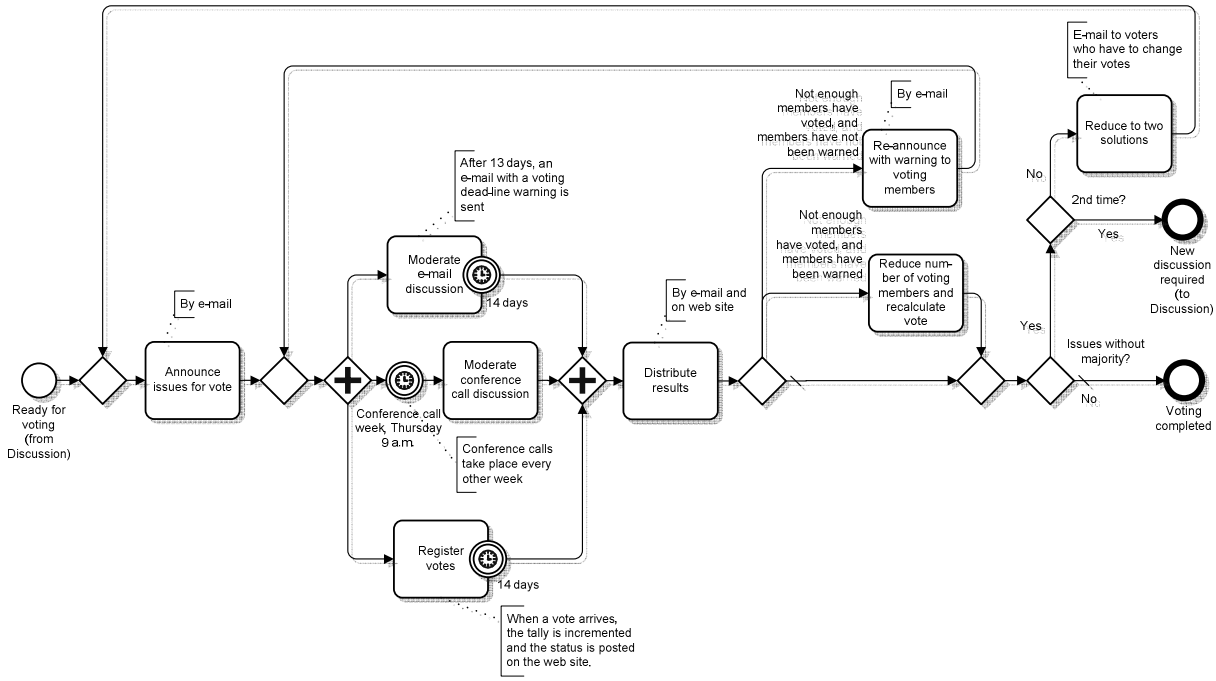

Figure 6: Voting process, first version

The beginning of the voting process is similar to the beginning of the discussion process. Parallel to the voting there is another e-mail discussion and another conference call. Since the voting period is 14 days, there is always one conference call during that period. Therefore a parallel gateway has been used. In the original diagram, registering the votes has been modeled with a rather unintuitive

event sub-process. This has also been simplified and reduced to one task which is carried out in parallel to the e-mail and conference call discussions. Both the e-mail discussion and registering the votes take 14 days. Like in the original diagram, this is modeled with attached intermediate timer events. Again, the details are explained in annotations. The new task “Distribute results” also replaces several smaller tasks.

The following sequence of several gateways has also been simplified. The number of gateways and different paths could be reduced by creating combined conditions (e.g. “Not enough members have voted, *and* members have not been warned”). The graphical layout and the consequent separation of merging and splitting also improve the readability.

6 Further Re-Design of the Models

The re-designed models are certainly easier to comprehend than the original model. However, there are still some details which may not be required for getting a good understanding of how the process works. First of all, the timer intermediate events can be omitted. The details concerning durations and times can be given in the annotations without losing significant information. In the discussion process, the decision at the beginning can be combined with the timer start event, so that the process is triggered on Fridays with ready issues. The combined event is not a pure timer event any more. BPMN contains an event-type “multiple” which could be used here. However, the “multiple” information is not important for understanding the process; so a simple “none” event is sufficient.

Since the connections between the processes are still a bit difficult to see, this information can be emphasized by placing it into annotations to the respective events. The resulting diagram is shown in Figure 7.

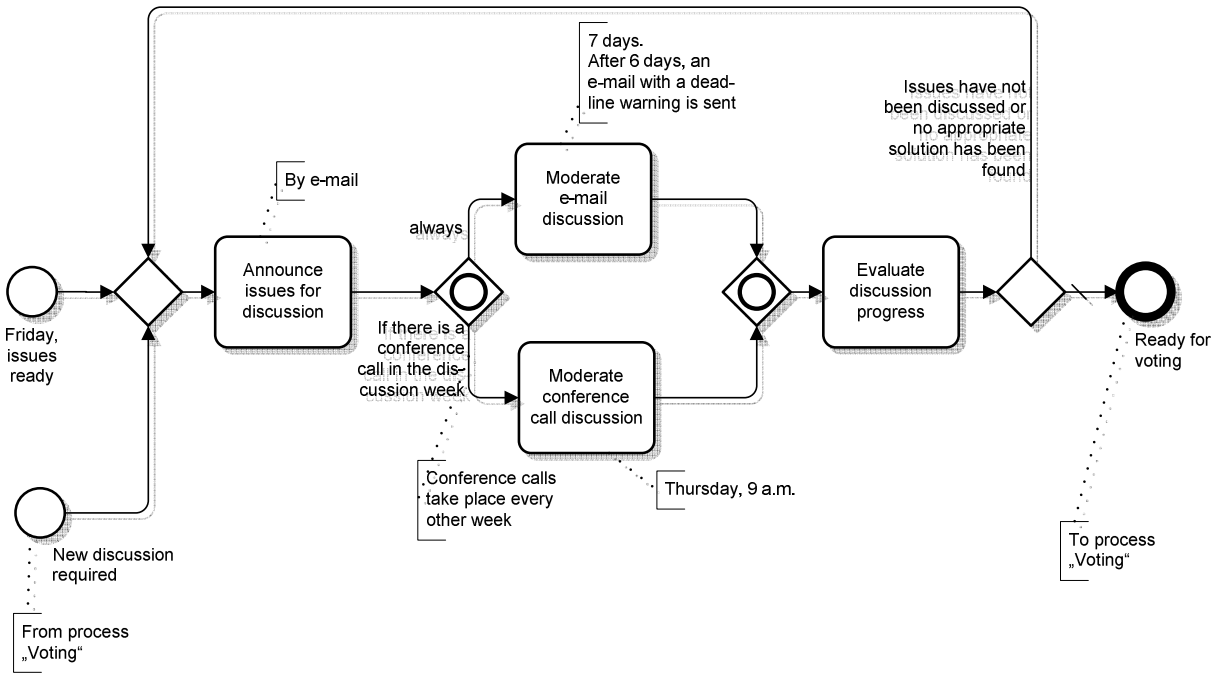

Figure 7: Discussion process, second version

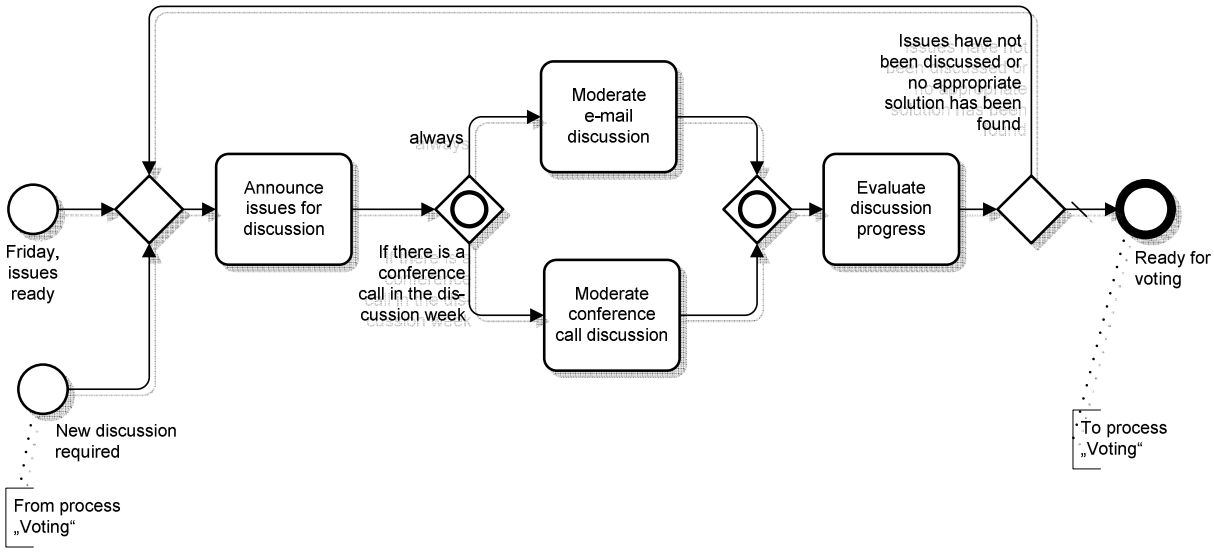

Figure 8: Discussion process with hidden task annotations

The basic structure would be even better visible if the tasks' annotations would be hidden (Figure 8). Although this diagram now provides an overview that can be grasped very quickly, the information of the annotations has been lost. Therefore, this information must be provided in a different way.

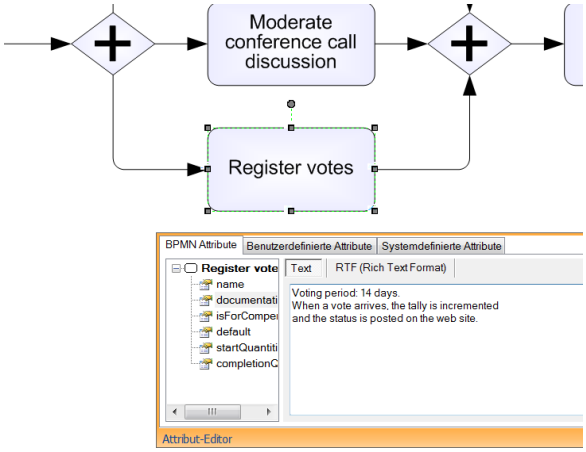

Figure 9: Displaying attributes for a selected element in a BPMN modeling tool (itp Process Modeler)

If the model is printed out in a document, it can be supplemented with a table that contains a short description of each task. If a modeling tool or an interactive web-interface is used for displaying the model, the description of each task can be displayed when it is selected. Such functionality is provided by many modeling tools. Figure 9 shows a screenshot from itp commerce's Process Modeler, as an example.

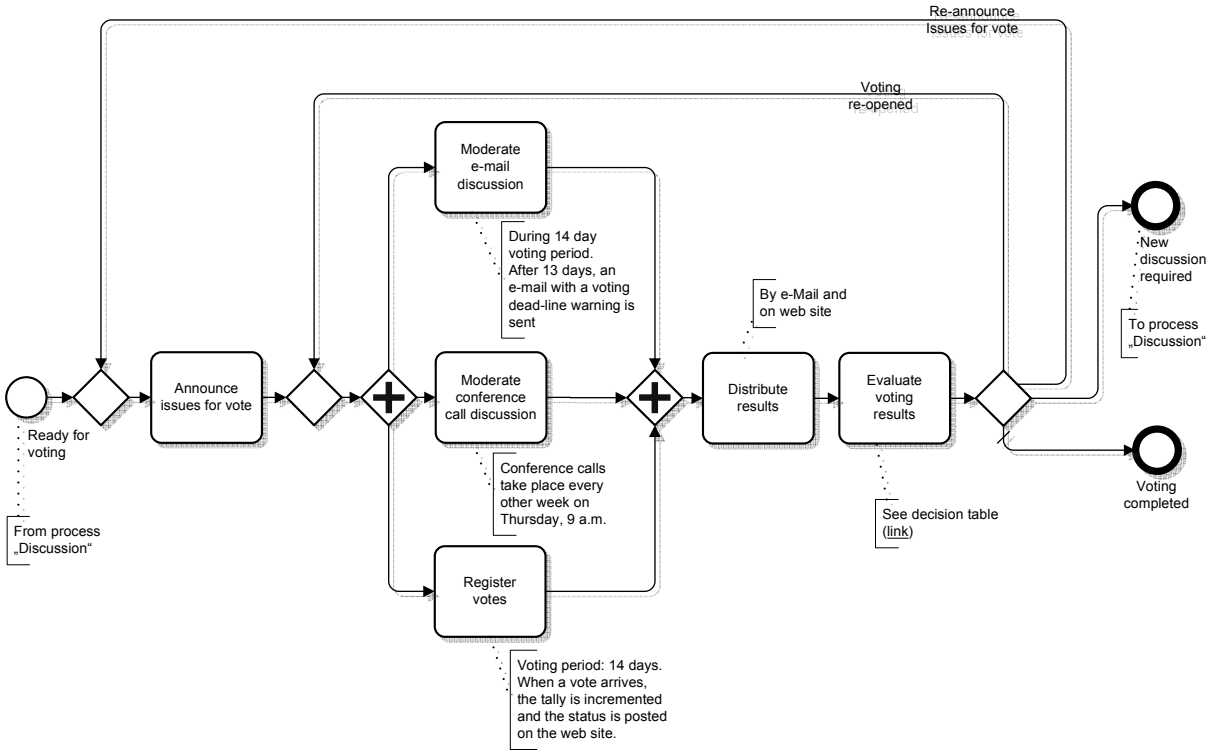

Figure 10: Voting process, second version

The model of the second process, the voting process, can be changed in a similar way to the first one. Although the number of gateways already has been reduced in Figure 6, the decision logic is still fairly complex. Since a process should only focus on the flow but not on complex decision rules, the entire chain of decisions can be hidden in one task “Evaluate voting results”. This task produces one out of four possible results which is used for selecting a path at the following exclusive gateway (Figure 10).

Enough members have voted	Members have been warned	Issues without majority	2nd time	Action	Decision
Yes	-	No	-		Voting completed
Yes	-	Yes	Yes		New discussion required
Yes	-	Yes	No	Reduce to 2 solutions and e-mail voters who have to change their votes	Re-announce issues for vote
No	Yes	Yes	Yes		New discussion required
No	Yes	Yes	No	Reduce to 2 solutions and e-mail voters who have to change their votes	Re-announce issues for vote
No	Yes	No	-		Voting completed
No	No	-	-	Re-announce with warning	Voting re-opened

Figure 11: Decision table for activity “Evaluate voting results”

The decision logic can be described in a separate text or model, such as a decision tree or a decision table. Figure 11 shows a decision table describing what needs to happen within “Evaluate voting results”. For each combination of conditions, the necessary actions within the task and the resulting decision are shown. This table could be connected to the task by a hyperlink.

This table is a little bit simplified, since for the case “Not enough members have voted” and “members have been warned”, a re-calculation of the vote for the reduced number of members is required, in order to determine whether there are issues without majority. If this needs to be reflected more thoroughly, there could be a sub-process for “Evaluate voting results” in order to define how the decision is made. In any way, the detailed decision logic is separated from the overall sequence flow.

Again, the process structure is easier to see when the task annotations are hidden and only displayed when required (Figure 12).

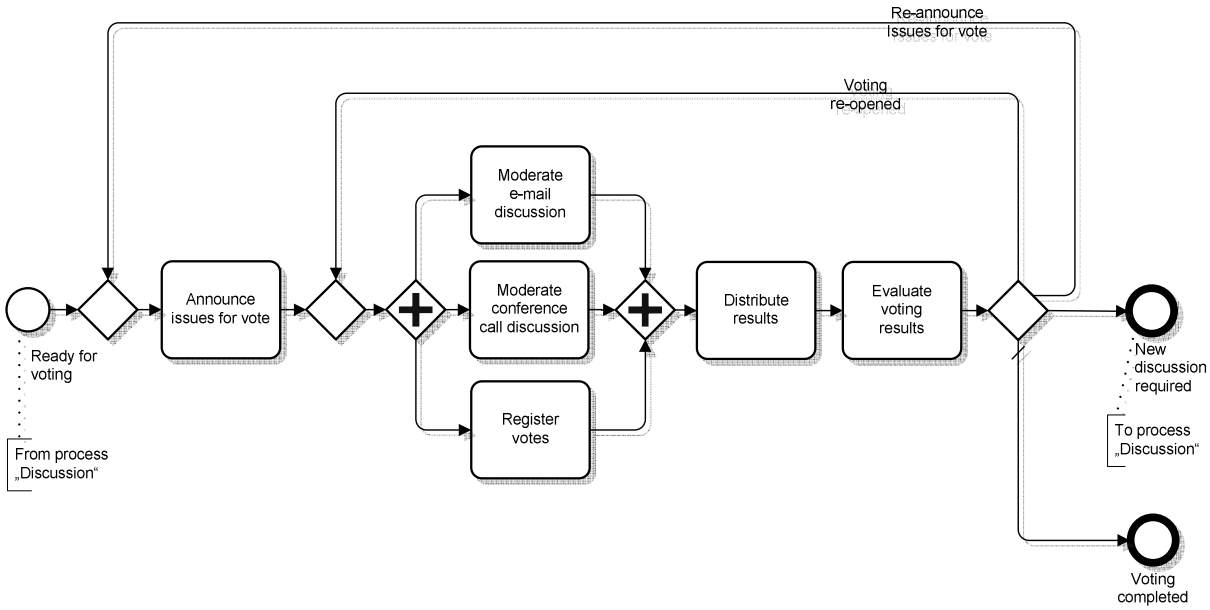

Figure 12: Voting process with hidden task annotations

7 Data Objects

In the previous diagrams, no data objects have been show in order to make the sequence flow more clearly visible. However, if the information about input and output data objects is required, data objects need to be included. The discussion process with data objects is shown in Figure 13.

In the original model, there are many long data association arrows crossing other lines. In order to avoid such a confusing structure, the data object symbols have been duplicated. They are always drawn above or below the connected tasks. This layout makes it easy to distinguish between sequence flow and data flow. Note that the data flows with arrows at both ends actually are two separate data flows which have been drawn on top of each other, since there are no bi-directional data flows in BPMN.

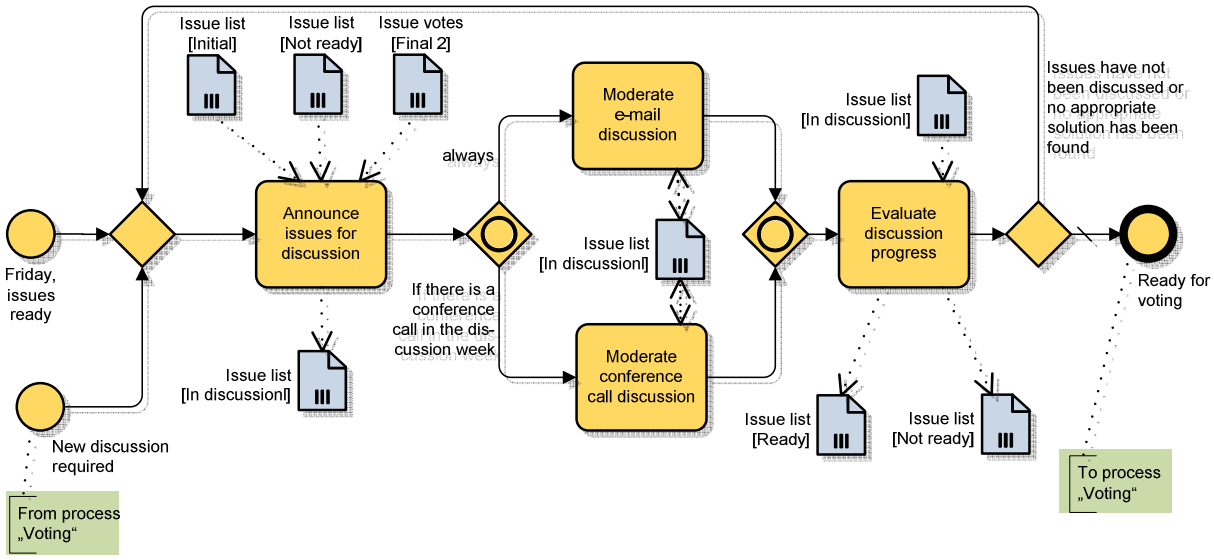

Figure 13: Discussion process with data-objects

Duplicating data object symbols is correct according to the BPMN specification. All data objects with the same name are references to the same data object. They may refer to different states of that object. This state is shown in square brackets.

Copying data objects to another process, however, is different. By definition, a data object only exists within the process in which it is defined. So if the two processes contain data object symbols with the same names, the two symbols represent two different data objects. Data objects are not automatically passed on between processes. If the processes were independent, a message flow would be required in order to exchange data. Another possibility would be to use a data store symbol, however this is more for permanent data stores which are independent from the processes.

If the two processes are regarded as sub-processes of a common parent process, the data objects can be defined in the parent process. The sub-processes can then refer to these common objects.

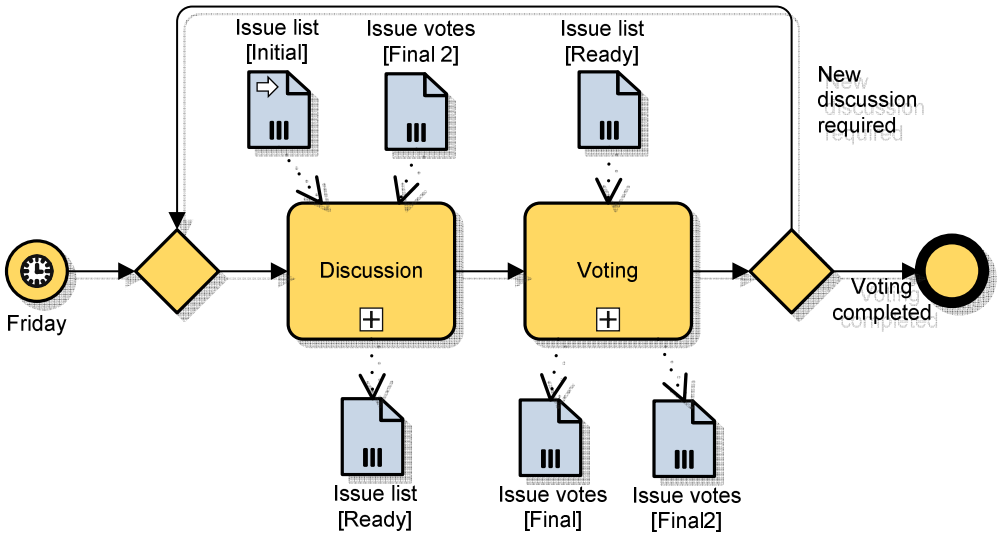

Figure 14: High-level process with data objects

Strictly following these rules, the parent process model from Figure 2 would need to be complemented with the common data objects. The resulting high-level diagram is shown in Figure 14. As in the original diagram, the object “Issue list [initial]” is marked as an input object to the entire process. In the sub-process, the input marker is not required any more, since the sub-process only refers to this object that already exists in the parent process.

Again, it is not recommended to present the model from Figure 14 to normal business users. However, this high-level model provides the context for the two sub-processes, so that the overall collection of models conforms to the BPMN specification (with the exception of the start events in the first sub-process). So even if the high-level model is not drawn, the existence of a “virtual” parent model could still be assumed, in order to conform to the specification.

The diagrams contain a few alterations compared to the original model. Some data objects have been omitted, because they are not important in the overall context of the process (such as “warning text” as input object for sending an e-mail with a deadline warning). The “Issue list [in discussion]” is probably changed by the discussions, so the data associations between the discussion tasks and the issue list are now pointing in both directions. All occurrences of the “Issue votes” object now have the multiple marker. This was missing in the original diagram at some places.

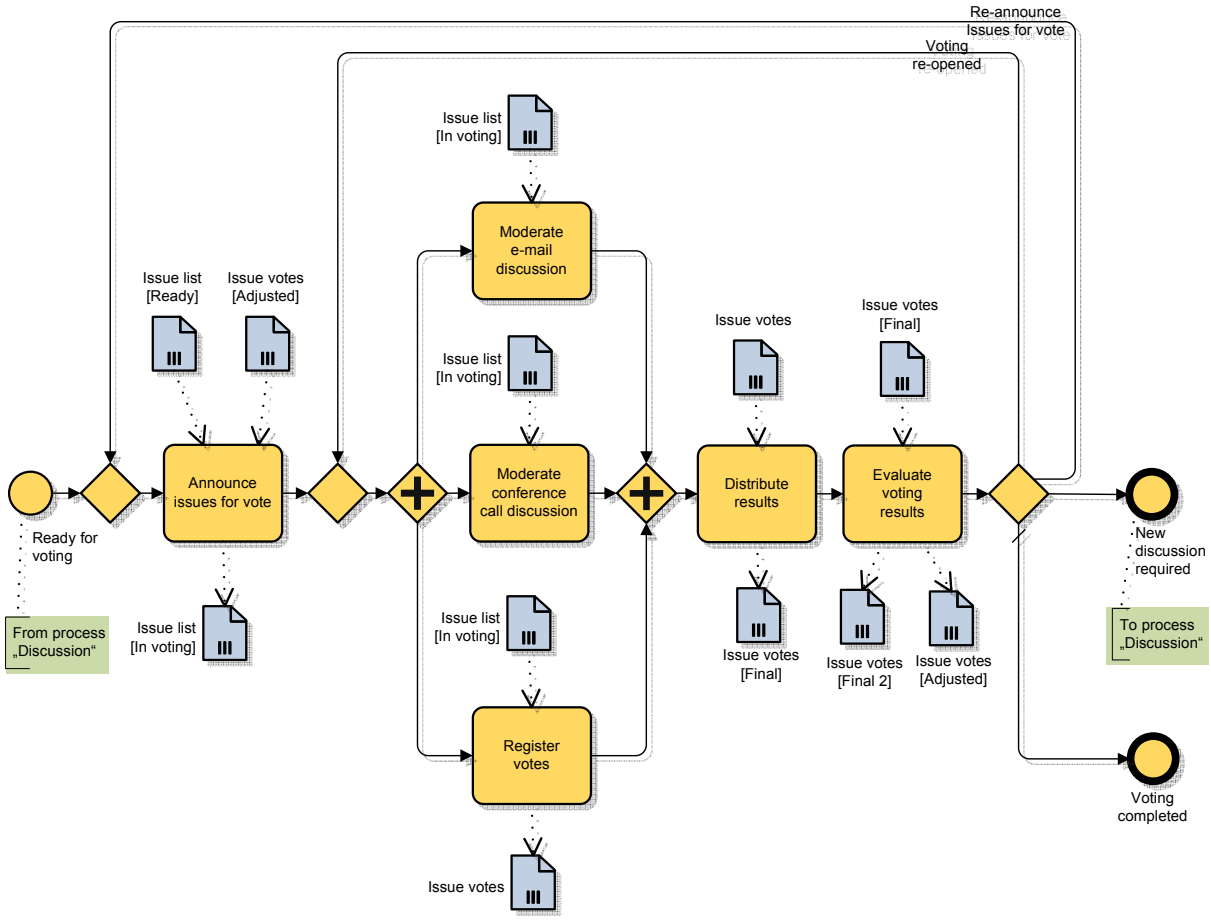

Figure 15: Voting process with data-objects

The diagrams here also show how color can be used for increasing the readability. All sequence flow objects have been colored orange. The data objects are blue, and the annotations concerning con-

nections to other processes are green. Although there are many elements in these diagrams, the process structure can still be grasped very quickly.

With some modeling tools the user can define which types of information should be displayed. There can be different views on one comprehensive model. In our example, the data objects could be hidden in order to provide the simple views of Figure 8 and Figure 12. When the user wants to inspect the data flow, he can have the data objects and associations displayed within the same model.

Another means to hide additional information is to attach detailed diagrams to each task. These detailed diagrams can include information like input and output, system transactions, resources etc. However, this requires the user to open a lot of diagrams if he is interested in several tasks.

8 Message Flow

It has been argued that for understanding the e-mail voting process, the modeling of message flows is not really necessary. Nevertheless, the message flows can also be integrated into the re-designed models.

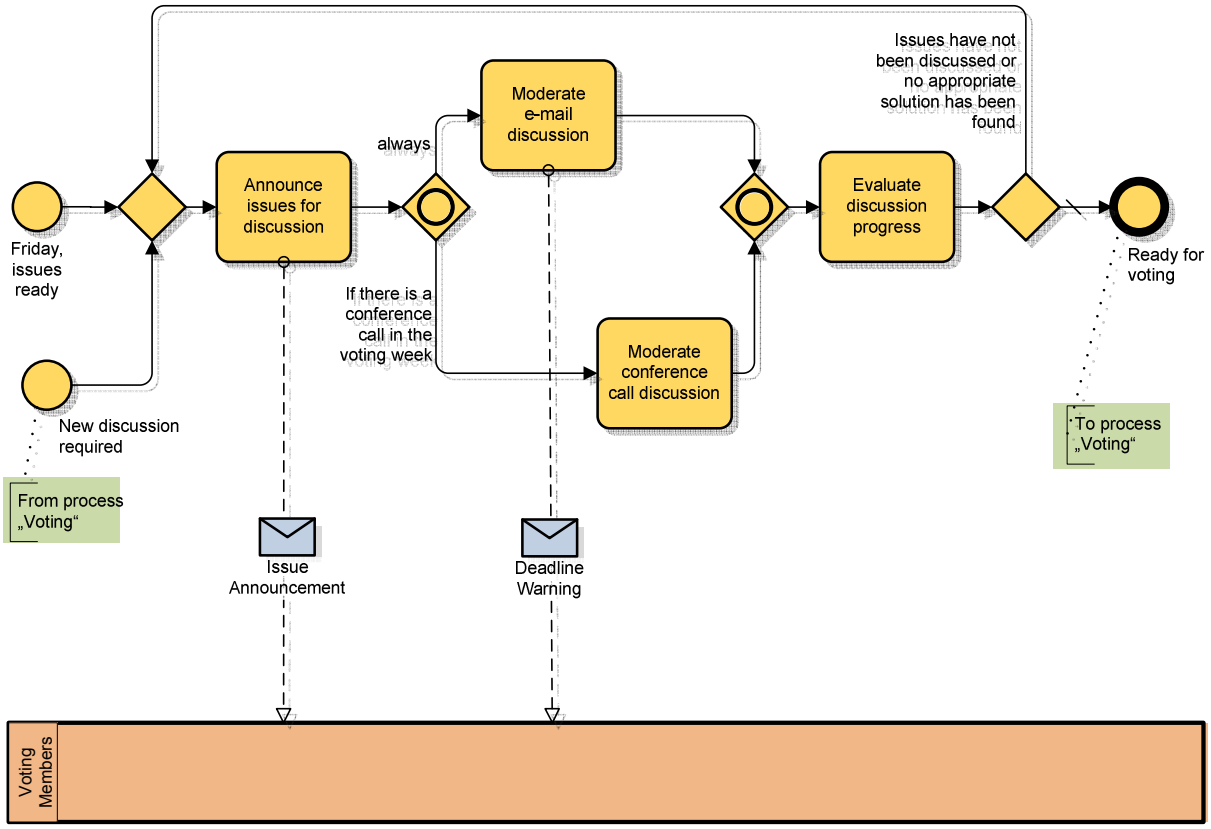

Figure 16: Discussion process with message-flows

In Figure 16 and Figure 17, the same principles as in the previous chapters have been applied:

- The simplified process structure allows for a clearer depiction of the message flows.
- The layout helps visualizing the flow structures. Some of the tasks have been moved horizontally so that it was possible to draw all message flows as straight vertical lines. There

are only few crossings of lines. This number could have been further reduced by exchanging the vertical positions of “Moderate e-mail discussion” and “Moderate conference call discussion”. However, it was decided not to change the basic structure of the previous diagrams, so that orientation is easier for the reader.

- Colors have been used for highlighting different aspects of the diagram.
- Other elements, such as task annotations and data objects have been hidden.

Again, a modeling tool can be helpful in order to hide details when they are not required. It could also provide a combined view with data objects and message flows. Without such a tool it is too tedious to draw and update several different diagrams of the same process.

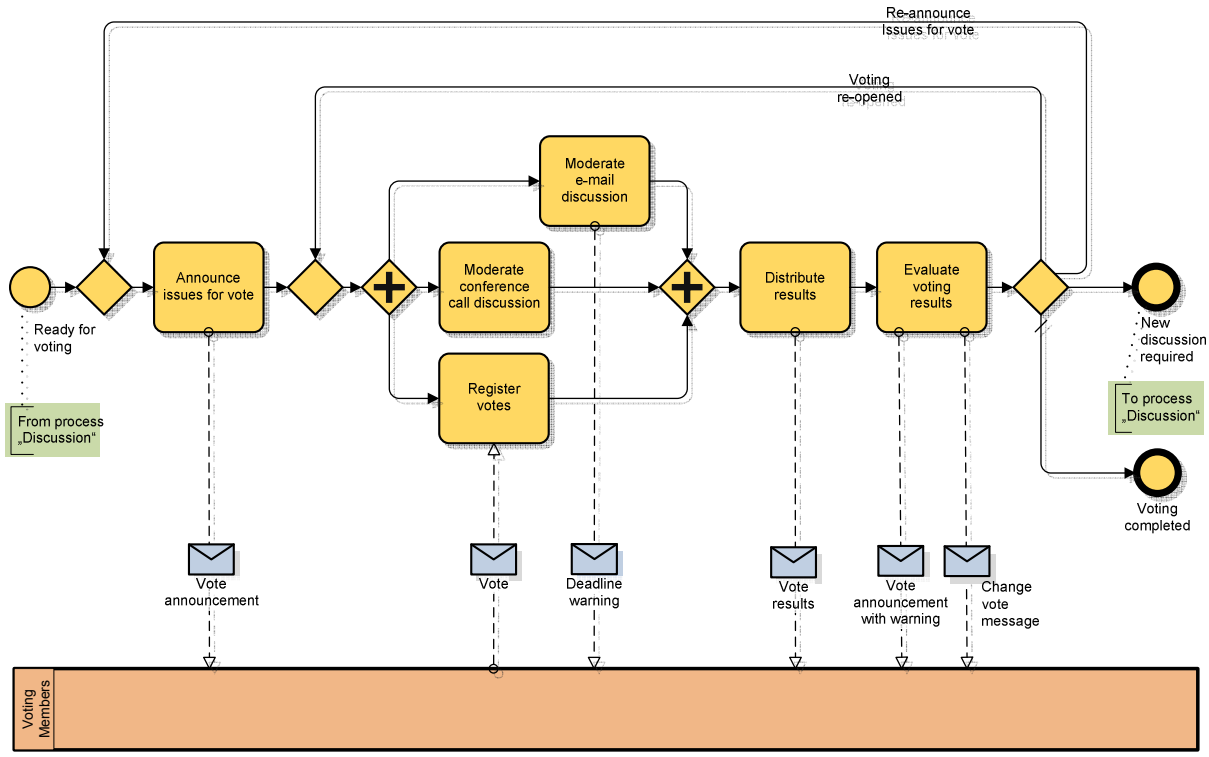

Figure 17: Voting process with message-flows

9 Conclusions

It has been demonstrated how a complex BPMN model can be re-designed in order to make it easier to read and comprehend. The re-designed models only serve the purpose of explaining the process to business users. They are neither a basis for workflow automation, nor detailed work instructions.

The re-design included the following changes to the original model:

- Dividing the model into two smaller processes, each fitting on one page
- Defining a simplified high-level model as a freeform diagram
- Creating fewer, but more comprehensive tasks
- Documenting details in annotations, when text is easier to understand than the graphical representation
- Replacing complex structures of parallel and exclusive gateways with inclusive gateways

- Replacing looping sub-processes by backwards-looping sequence flows
- Replacing event sub-processes by parallel sequence flow paths
- Applying uniform rules for modeling splits and merges (always using gateways, each gateway either used as a split *or* as a merge).
- Using annotations for defining connections between the two processes
- Moving complex decision logic from the BPMN model to a separate documentation, e.g. a decision table
- Using a consistent layout with horizontal sequence flows and vertical data and message flows
- Highlighting different aspects of the model with colors
- Hiding annotations, data objects and message flow, and displaying them only when required. Thus, different views onto the process are provided.

During the re-design it turned out as a problem that the BPMN 2.0 concepts still have a strong background in process automation. This creates many challenges for creating process models for business users. Especially creating hierarchical models is rather difficult, since correct BPMN models require high-level models to include many details (e.g. exception flows and data objects). Defining correct connections between different sub-processes also requires rather complex model structures which make it difficult to understand the models (e.g. when a sub-process can be triggered by different incoming sequence flows, each requiring a different start in the sub-process). Another example concerns the markers for different task types. Icons for different task types could also be useful for business modeling; but they are only defined in respect to process automation.

The author hopes that future BPMN versions will better reflect the business modeler's needs.

In order to create easily readable models for business users, modelers are recommended to relax the strict BPMN rules where they make the models difficult to understand. However, this should only be done when there is a good reason, since otherwise the resulting models may be too far away from the standard. Modeling conventions should be defined in which deviations to the standard are documented.

Such modeling conventions should also include rules concerning modeling size and layout, the sub-set of BPMN elements to be used, naming conventions, etc. This is a precondition for uniform and consistent models.

Whether a model is easy to understand is always subject to personal judgement and invididual preferences. The models in this paper are therefore only suggestions, and the readers are invited to discuss these models or provide alternative solutions. Although no systematic evaluations of the readability of these models have been made, the author is convinced that they are much easier to understand than the original model.

References

[BPMNSpec] Object Management Group, OMG (ed.): Business Model and Notation (BPMN) Version 2.0. Beta 2. June 2010. OMG document number: dtc/2010-06-05
URL: http://www.omg.org/spec/BPMN/2.0/Beta2/PDF/

[BPMNExamples] Object Management Group, OMG (ed.): BPMN 2.0 by Example. Version 1.0 (non-normative). June 2010. OMG document number: dtc/2010-06-02
URL: http://www.omg.org/spec/BPMN/2.0/examples/PDF/10-06-02.pdf

The Author

Thomas Allweyer is a professor for enterprise modeling and business process management at the University of Applied Sciences Kaiserslautern, Germany. He is a regular speaker at national and international conferences and has published many papers and several books, among them an introduction into BPMN 2.0. He holds seminars and consults international companies on process management and modeling. He is the author of the process management blog www.kurze-prozesse.de (in German).

He can be reached at thomas@allweyer.de.

The models in this paper have been created with Trisotech BPMN 2.0 Modeler, a Microsoft Visio-add-on which can be downloaded for free at http://www.businessprocessincubator.com/ (registration required).

Teach Yourself BPMN 2.0!

ISBN 978-3-8391-4985-0

Paperback, 156 Pages

www.bpmn-introduction.com