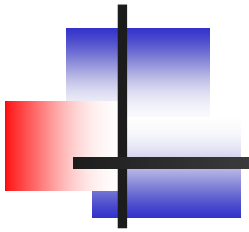
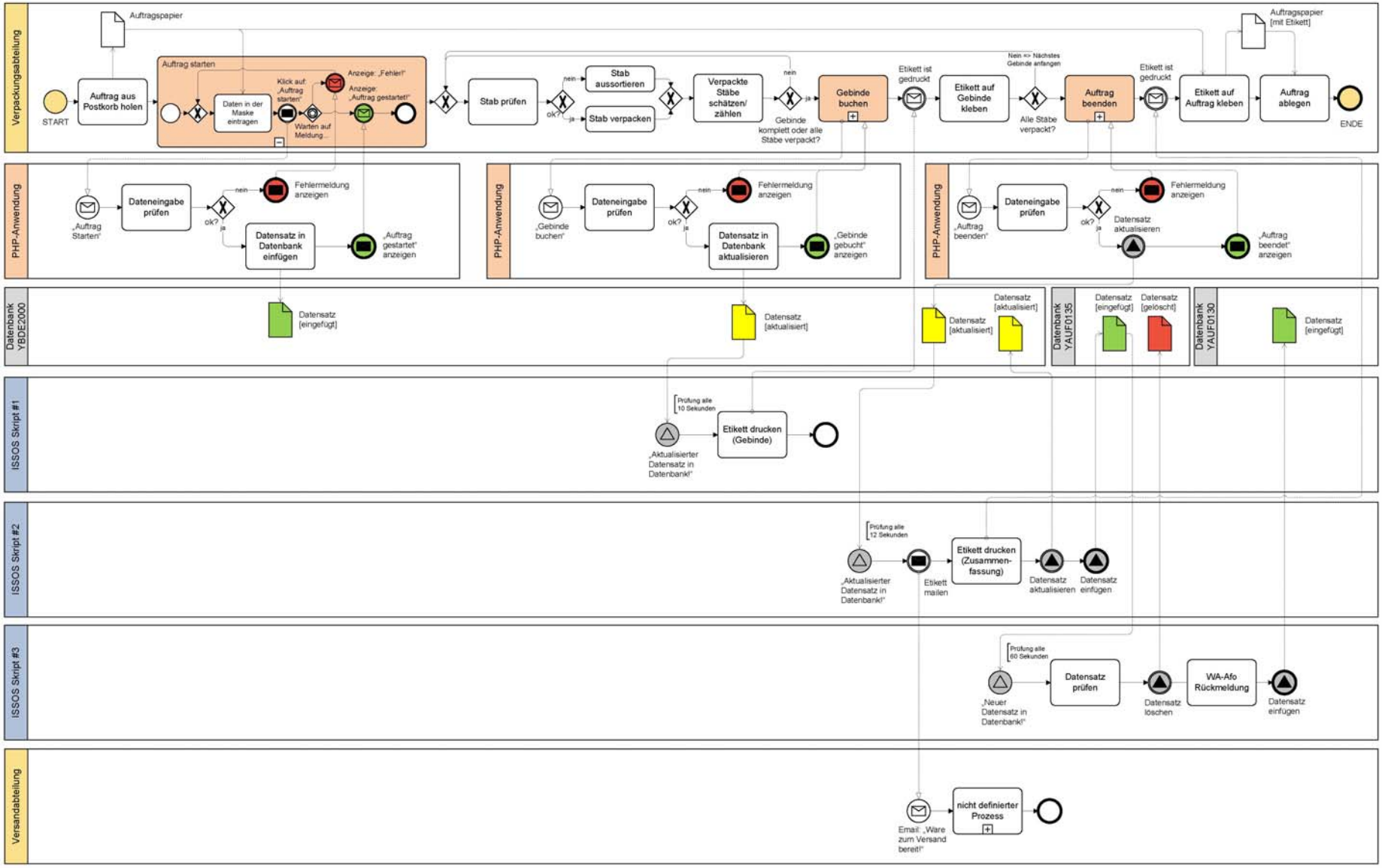


A: Ein ganz normaler Prozess
B: Best Practices in BPMN 1.x

ITAB / IT Architekturbüro Rüdiger Molle

März 2009

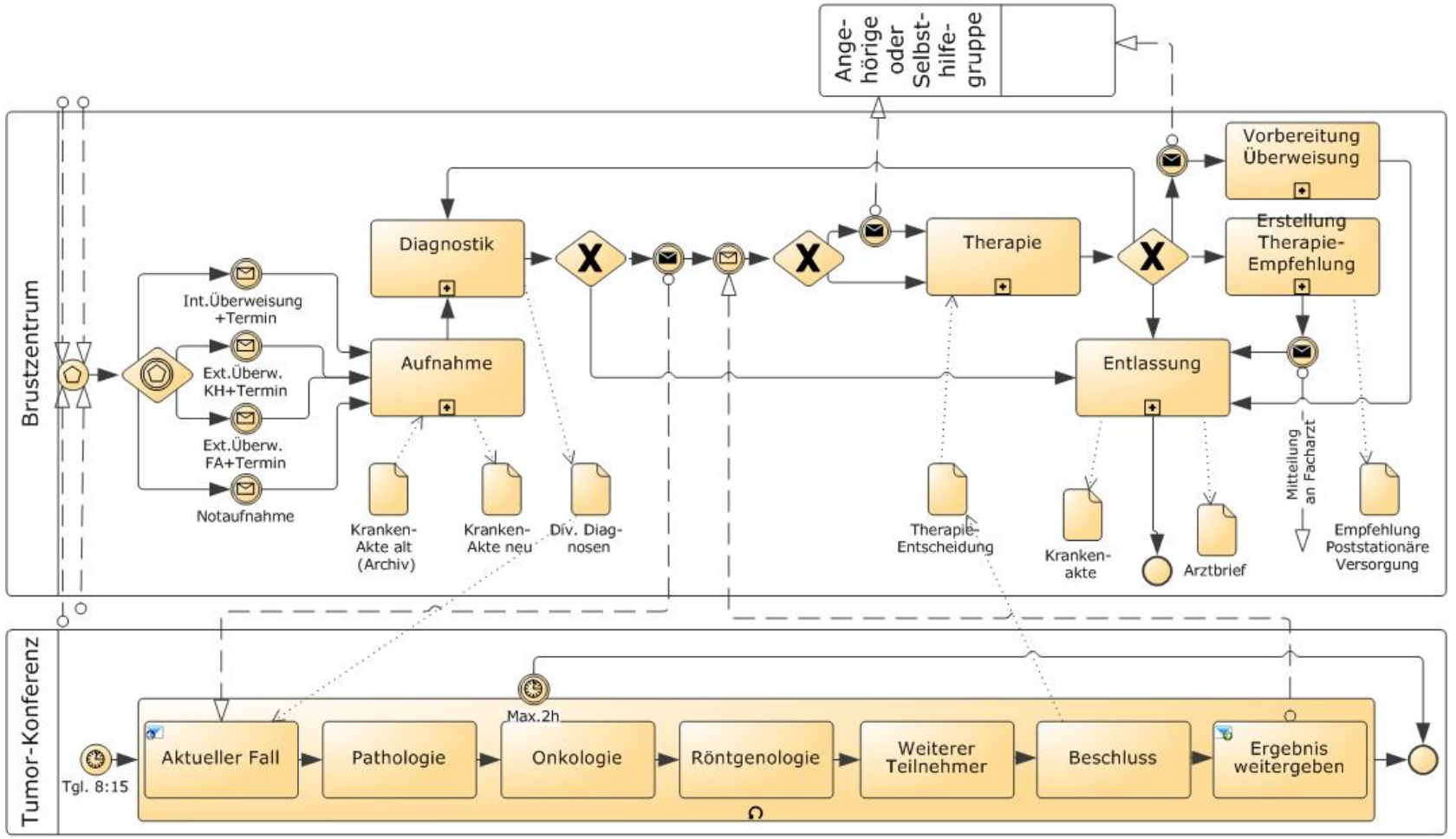
Geschäftsprozess „Stabverpackung“



Lessons learned

- Beschreibung eines GP durch das Business läßt Fragen der Prozess-Seite offen → Prozessentwurf im Team + Konventionen erforderlich
- Es gibt unterschiedliche Lösungen: elegante, schnelle, implementierungs-nahe Variante
- Die verwendeten Modellierungswerkzeuge (ORYX, BizAgi, PM) interpretieren BPMN 1.x unterschiedlich, es gibt ein „Loch“ in der Spezifikation.
  - Nutzung unterschiedlicher Task-Typen in obigem Geschäftsprozess?
  - „Gebinde buchen“ und „Auftrag beenden“ als User Task?
  - Eine Assoziation darf nicht ein Datenobjekt mit einem Ereignis oder einem Gateway verbinden (Syntaxfehler laut Process Modeler 5.2, für ORYX nicht, für BizAgi ebenfalls nicht). Problem: Ein Zwischenereignis läßt einerseits kein assoziiertes I/O-Set zu, andererseits steht in Kap 9.7.2., dass Datenobjekte (als Artefakte) mit Flussobjekten (wie Ereignissen und Gateways) assoziiert werden können
  - Ein Throw Signal sollte den gleichen Namen haben wie das zugehörige Catch Signal: entscheidend ist das Attribut „EventDetail“ mit dem „SignalRef“-Eintrag, der bei beiden Ereignissen übereinstimmen muss. Da diese Einträge in der Zeichnung nicht sichtbar sind, sollten sie im Namen wiederholt werden.

Der klinische Pfad als normaler Prozess



Mein Diagramm	Autor: Administrator	erstellt: 30.01.2009 12:43:12	
CP Mammakarzinom	Version: 1.0	geändert: 18.03.2009 17:32:19	
	Status: erstellt		
	Mammakarzinom.vsd		

Best Practices in BPMN (Bruce Silver 1)

- **Make the process logic visible in the diagram.** This is absolutely fundamental, but is routinely ignored by beginning modelers. The BPMN spec describes various shapes and connectors that print in the diagram, plus supplementary detail in attributes visible only through the modeling tool or in the detailed documentation that can be generated using the tool. But maximizing shared understanding doesn't come from individuals privately examining your model through the tool, nor from digging through 100 pages of documentation. It comes from a group sitting around the table looking at a printout of the diagram, discussing it, usually thinking about how it could be better done.
  - Effectively that means two things: First, label everything in your diagrams – not just activities, but sub-processes, intermediate events, gateways, sequence flows, end events and message flows. Some required attributes, like the duration of a timer event, may not be displayed in the diagram. If not, add a label to the event that replicates that information. If you can't see it in the diagram, it doesn't really count.
  - Second, show exception handling logic explicitly in the diagram. Unlike many traditional notations, BPMN gives you the tools to do that even if you're not a developer.
- **Make your models valid.** While the diagram is the key output, a process model is more than a drawing, and a modeling tool is more than a drawing tool. A real modeling tool has the semantics and rules of BPMN baked in, and gives you a Validate button that can display a list of errors when you violate the spec. A free BPMN stencil in Visio can't do that. If you want others to understand your models, you need to start by making them valid, so you should use that Validate button and learn to fix the errors.
- **Make your models hierarchical.** What makes BPM different from traditional management disciplines is its emphasis on viewing the business from a cross-functional end-to-end perspective. Capturing processes in flat models that take up thirty feet of wall space does not allow that end-to-end perspective to be consumed all at once. Instead, we teach a top-down methodology in which the top-level diagram shows the whole process on a single page, and uses sub-processes to expand process detail at nested diagram levels, so you can zoom in and out of your model to describe any level of detail. It may print out as multiple pages, but internally the integrity of a single model is maintained.

Best Practices in BPMN (Bruce Silver 2)

- **Label process activities VERB-NOUN.** BPMN describes processes in terms of activity flows, where activities are *actions*. They represent work done in the process. Activities are not states, not business functions, not use case interactions. To reinforce that, we ask students to consistently name their activities using the Verb-Noun construction, like Check Credit or Validate Order, not Credit Check (a function) or Valid Order (a state). A resource is going to be performing each activity, and the name of the activity should describe what action is performed.
- **Specify task types.** One of the BPMN attributes with no standardized representation in the diagram is the task type. The spec defines several task types, but there are really two that are important to distinguish: user (human task) and service (automated task). Fortunately, most BPMN tools distinguish task types with icons inside the activity shape... but you need to specify which type you mean.
- **Don't use a task to route work.** Another common beginner mistake is to insert tasks like Send to Manager, followed by a sequence flow to a task in the manager's swimlane. That sequence flow is already routing work to the manager, so the Send to Manager task is redundant. Just get rid of it.
- There's a second problem here as well. Best practice is to reserve the keywords "Send" and "Receive" in task names to send and receive task types, which are equivalent to message events. In BPMN a "message" means a signal between the process and some external entity. Here manager is not an external entity, but a participant in the process. So it's not a message and should not be labeled Send. If you want to communicate something to the manager without routing the work itself, you could use a task called Notify Manager. Practices like this may seem petty at first, but in the end they make it easier for everyone in your organization to understand immediately from the diagram what is going on.

Best Practices in BPMN (Bruce Silver 3)

- **Distinguish success and failure end states in a sub-process with separate end events.** Each path in a sub-process that is enabled must reach an end event before the sub-process is complete. You can draw a single end event for the sub-process and route all paths to it, or you can draw multiple end events and route specific paths to each one. Since there is an implied "join" of all the end events, technically it doesn't matter. But there is benefit in drawing and labeling separate end events for each distinct end state of the sub-process, particularly if some end states represent "success" and others represent "failure" or some type of exception.

  You can follow the sub-process with a gateway that tests the end state to see whether to continue the process or do something else, like end it or loop back to a previous step. Matching the label on the gateway to that of the end event makes that linkage clear in the diagram. Alternatively, an end event on an exception path inside the sub-process can re-throw the error, which is caught by an attached error event on the sub-process boundary. This also can end the process or follow some other exception flow. Again, matching the labels of the throwing and catching events makes the connection obvious from the diagram.
- **Use sub-processes to scope attached events.** Intermediate events attached to a process activity mean if the event occurs while that activity is running, abort the activity and proceed down the sequence flow out of the event, called the exception flow. A neat trick is to wrap a sequence of activities with a sub-process for the sole purpose of defining that scope for the event. For example, if you have an order handling process with steps A through Z and you want to allow the customer to cancel or change the order without penalty any time between steps B and G, you can wrap the sequence from B to G in a sub-process and attach a message event and particular handler exception flow to that.

Best Practices in BPMN (Bruce Silver 4)

- **Standardize on specific diagram patterns to distinguish types of exceptions.** BPMN provides a business-friendly notation for describing exception-handling behavior. Even though the BPMN spec gives the modeler great freedom, best practice is to learn specific diagram patterns to distinguish each type of exception, and use them consistently. In our course, for example, we teach distinct patterns for modeling internal business exceptions, system faults, timeouts, solicited response exceptions, unsolicited events, and others. Again, the key principle is understanding exactly what is meant just from the diagram itself.
- **Use message flows consistently to show business context.** In addition to the activity flow of your own process, BPMN lets you show the interactions between your process and external processes as dashed connectors called message flows. Message flows typically represent requests, responses, and unsolicited events exchanged with the external process. In your own process, message flows connect to specific activities and events that indicate exactly how your process responds to an incoming message flow or generates an outgoing message flow. Since you do not control or even know the internals of the external process, it is common to just attach the message flows to the boundary of the pool representing that process.

  Message flows can add valuable business context to your diagram, but it is important to use them consistently. For example, if you are going to show *any* message flows from and to a requester of your process, you really should show *all* of them, and show them consistently in each level of your model. That is, if a particular message flow is shown in a sub-process nested three levels down, it should also be shown in the top-level diagram, and labeled the same at every level.

Best Practices in BPMN (Stephen White & Derek Miers 1)

- **Sending and Receiving Messages**: The modeler could choose to use only Send and Receive Tasks, or to use only the throw and catch Message Intermediate Events. The Best Practice is to avoid mixing both approaches together in the same model. There are advantages and disadvantages to both approaches. Message Intermediate Events give the same result and have the advantage of being graphically distinguishable whereas the Tasks are not. On the other hand, using Tasks, rather than the Events can enable the modeler to assign resources and simulate costs.
- **Use of Start Events**: In general, we recommend that modelers use Start and End Events.
- **Setting Timers**: Avoid specific date and time conditions as they inhibit the re-usability of the process.
- **Use a Default Condition**: One way for the modeler to ensure that the Process does not get stuck at an Exclusive Gateway is to use a default condition for one of the outgoing Sequence Flow. This creates a Default Sequence Flow. The Default is chosen if all the other Sequence Flow conditions turn out to be false.
- **Use a Timer Intermediate Event with an Event Gateway**: One way for the modeler to ensure that the Process does not get stuck at an Event Based Exclusive Gateway is to use a Timer Intermediate Event as one of the options for the Gateway.
- **Ensure that the number of incoming Sequence Flow is correct for a Parallel Gateway**: The key point is to exercise care, ensuring that merging Parallel Gateways have the correct number of incoming Sequence Flow – especially when used in conjunction with other Gateways. As a guide, modelers should match merging and splitting Parallel Gateways if the desired behavior is to merge them again.

Best Practices in BPMN (Stephen White & Derek Miers 2)

- **Use a Default Condition on an Inclusive Gateway**: One way for the modeler to ensure that the Process does not get stuck at an Inclusive Gateway is to use a default condition for one of the outgoing Sequence Flow. This Default Sequence Flow will always evaluate to true if all the other Sequence Flow conditions turn out to be false.
- **Always use Inclusive Gateways in pairs**: A way to avoid unexpected behavior is to create models where a merging Inclusive Gateway follows a splitting Inclusive Gateway and that the number of Sequence Flow match between them.
- **Use a Text Annotation with the Complex Gateway**: Since the actual behavior of a Complex Gateway will vary for each usage of the Gateway, use a Text Annotation to tell the reader of the diagram what behavior the Gateway is set to perform.
- **Use a Standard or Default Sequence Flow when using Conditional Sequence Flow**: One way for the modeler to ensure that the Process does not become stuck after an Activity is to use a standard or Default Sequence Flow whenever Conditional Sequence Flow are used.
- **Do not Associate a Data Object with a Sequence Flow if the Sequence Flow is connected to a Gateway**: The application of inputs and outputs can be easily confused when one or more Gateways is used for Sequence Flow that are associated with Data Objects.
- **Modeling Inputsets**: If there is more than one inputset, pick a point on the boundary of an Activity and have all inputs that belong to a single inputset connect to that point. The inputs for the other inputsets should each connect to separate points on the boundary of the Activity. The same pattern should apply to modeling outputsets.