cprime

RETHINKING BPMN: HOW TO BUILD GOOD PROCESS MODELS USING BPMN

Myths, Misconceptions, Best Practices

BY RAZVAN RADULIAN

©2020. Cprime, Inc. All Rights Reserved

WWW.CPRIME.COM | LEARN@CPRIME.COM

TABLE OF CONTENTS

INTRODUCTION

In the last decade or so, Business Process Model and Notation (BPMN) (Object Management Group 2014) has become the 'de facto' standard for modeling business processes. Numerous books about BPMN have been published and most process modeling tools have already adopted the BPMN standard.

Still, most BPMN process models in the real world (and, unfortunately, even in books and tutorials/ training courses, etc.) are often bad models (Leopold, Mendling and Günther 2016). Furthermore (and, probably, because of that), quite a few people (surprisingly, many from the BPM community) complain/claim that BPMN diagrams are "too complex and confusing" and that business people "cannot understand BPMN models". Indirectly, suggesting that BPMN diagrams are good only for technical engineers and developers. Despite the fact that one of the main reasons BPMN had been created in the first place was to build models that can be easily understood by both business and technical people.

An often-used example, of how unfriendly BPMN is, comes from Object Management Group itself, in its BPMN By Examples documentation of an "E-Mail Voting" process (OMG 2010). someone committed to the technology, I will attempt the pull you towards the positive aspects and benefits of the platform.

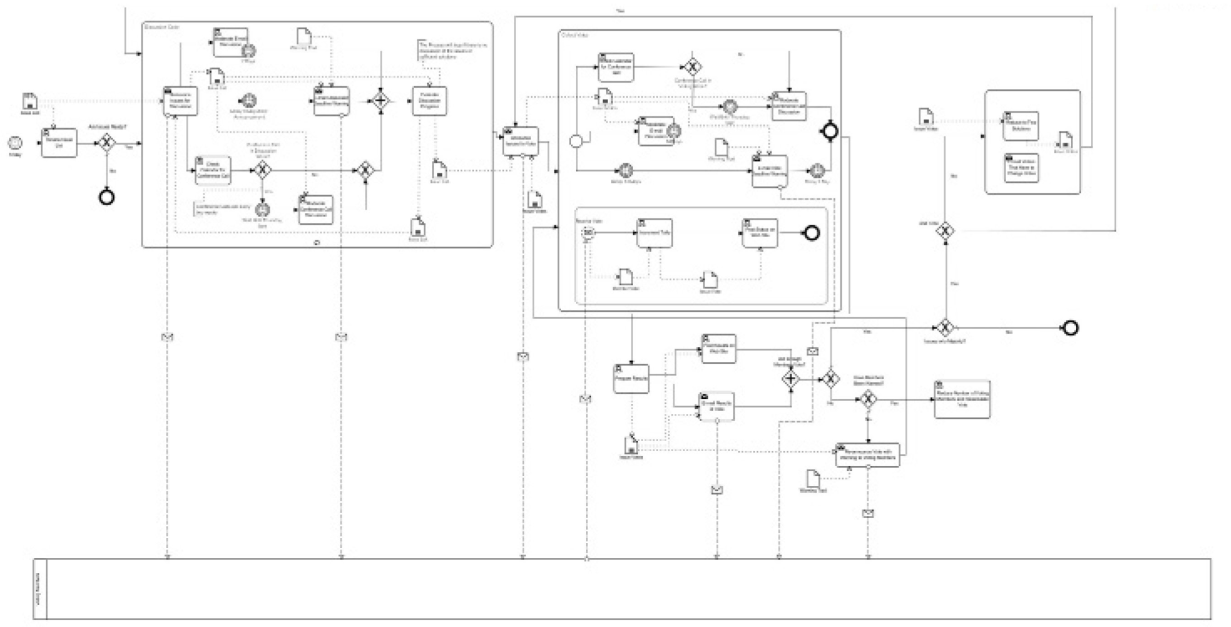

Figure 1: E-mail voting example... Not the best example of how to model processes

BPMN's main goal is to build process models that can bridge the communication gap between business and technical/IT stakeholders. No more copying and pasting, no more translating (and all the inherent translation errors that come with that), no more using different notations, for different audiences, and no more using separate tools, depending on who the users are or who their audience is. One process model for all! Whomever they are, whatever tool we choose to use.

HAS BPMN FAILED? WHAT WERE/ARE THE REQUIREMENTS?

Let's consider some of BPMN's main requirements/expectations. BPMN models should be able to:

- Be easily understood by both business and technical/IT stakeholders
- Capture both business and technical/IT's process requirements
- Allow zooming-in (to see lower level process details) and zooming-out (to summarize/"hide" lower level process details and to see just the "big picture")
- Allow progressive elaboration, adding more process details, without having to use a new language/notation or to create entirely new models (e.g. copy and modify)
- Separate process logic from non-process details (e.g. data definitions, user interfaces, decisions/business rules, organizational structure, complex events, etc.); however, they should also allow us to show how to integrate the process models with those other elements (i.e. relationships/traceability)

- Specify how a process model relates to other process-related architectural models/elements (e.g. value chains, process landscapes, decisions, etc.) and how it integrates with other recognized architectural frameworks (e.g. TOGAF, ArchiMate, etc.)
- Define processes that can integrate and coordinate/orchestrate external services (e.g. web-services, SOA), when executed in a BPMN capable process engine

WHY ALL THE COMPLAINTS?

BPMN meets all above requirements! Why then, are so many people complaining (often, passionately) about BPMN? Here are some possible explanations:

- Not really understanding BPMN and how it should be used
- Not knowing the difference between BAD and GOOD BPMN and, as a result, confusing the majority of bad BPMN examples with BPMN itself (kind of like saying "English is bad only because we see many bad essays during college applications")
- Not knowing how to build good BPMN models (and so, imagining that ALL built BPMN models must be bad)
- Not understanding when NOT to use BPMN, trying to stretch it beyond its indented use (no, BPMN is not a "Silver bullet"!)
- Not "liking" BPMN because it threatens the status-quo or some other alternative that some powerful stakeholders want to hold on to

CLEARLY, WE DON'T JUST NEED PROCESS MODELS, WE NEED GOOD PROCESS MODELS THAT ARE BUILT PROPERLY, USING THE BEST STANDARDS & TOOLS

To help us solve this challenge, we need to answer a few important questions:

- Why do we need good process models?
- What is a good process model?
- What makes BPMN a better choice for building good process models?
- How can we build good BPMN models?
- When is BPMN not a good choice?

I will assume, since you are reading this paper, that you already know the answer to the first question.

Today, I will try to address the second question.

WHAT MAKES A PROCESS MODEL "GOOD"?

We will adopt, and in some respects expand, Bruce Silver's recommendations in "BPMN Method and Style" (Silver 2012).

To be GOOD, a process model (BPMN or otherwise) must be:
a. Correct
b. Clear
c. Complete
d. Consistent

A. CORRECT

There are two level of correctness. First, the process model must respect the syntax/modeling rules specified by the process modeling language/standard being used (e.g. the BPMN standard). Second, the model should be semantically correct/accurate from the business point of view. It is important to note that a standard like BPMN can only ensure syntactical correctness.

Let's look at one example (using BPMN):

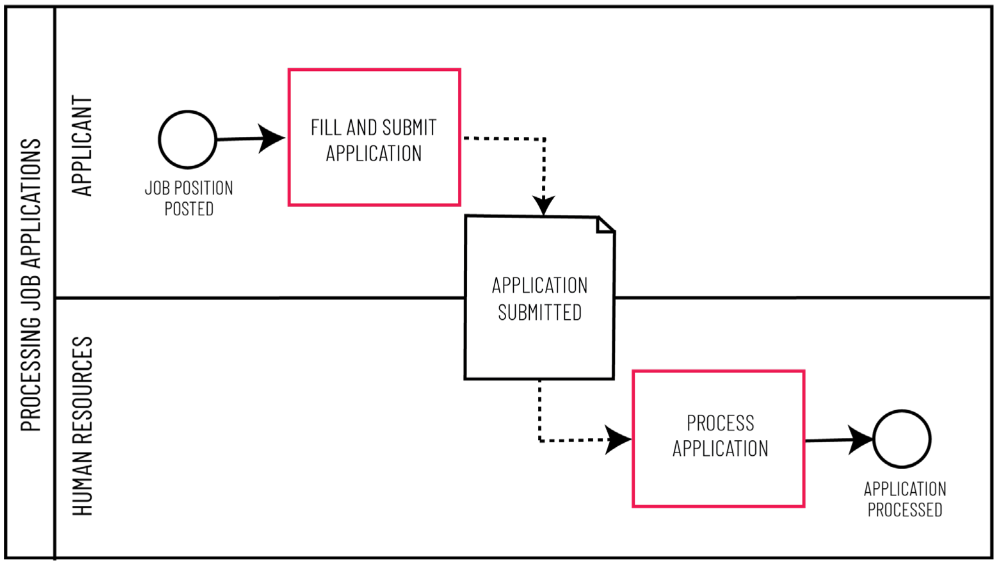

Figure 2: Does this process' work... flow?

At first glance, it seems simple and correct. But, while it might indeed be simple, it is not correct. Why?

Assuming the Applicant started the process (I'll explain shortly why that is an assumption), the application will never be processed. Once the application is submitted, yes, an Application (data object) will be generated, but the process will end there since there is no sequence flow indicating what should happen next (called "sequence flow" for an obvious reason). That is because the associations (dotted line) do not indicate that info.

So long story short, how could this process be interpreted to "work" (something a developer or a process engine would do)?

First...

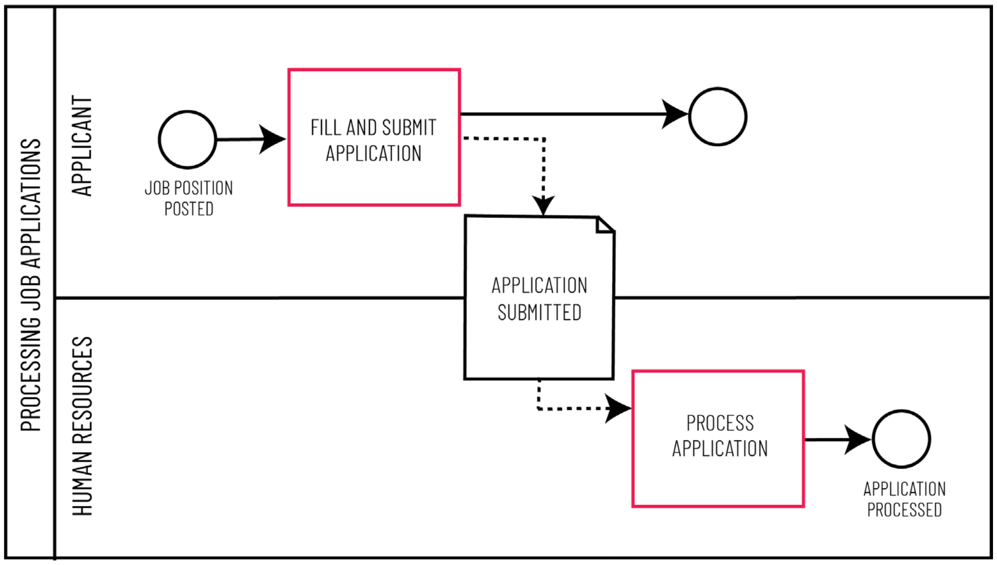

Figure 3: When an activity does not have any outgoing sequence flow, BPMN implies an End Event

... not quite what was intended, isn't it? What about the "Process application" task? Is it completely "idle"? Not really.

A process engine (if you could feed this model into an actual, or even a virtual, process engine) or a developer interpreting what to build, could see it this way:

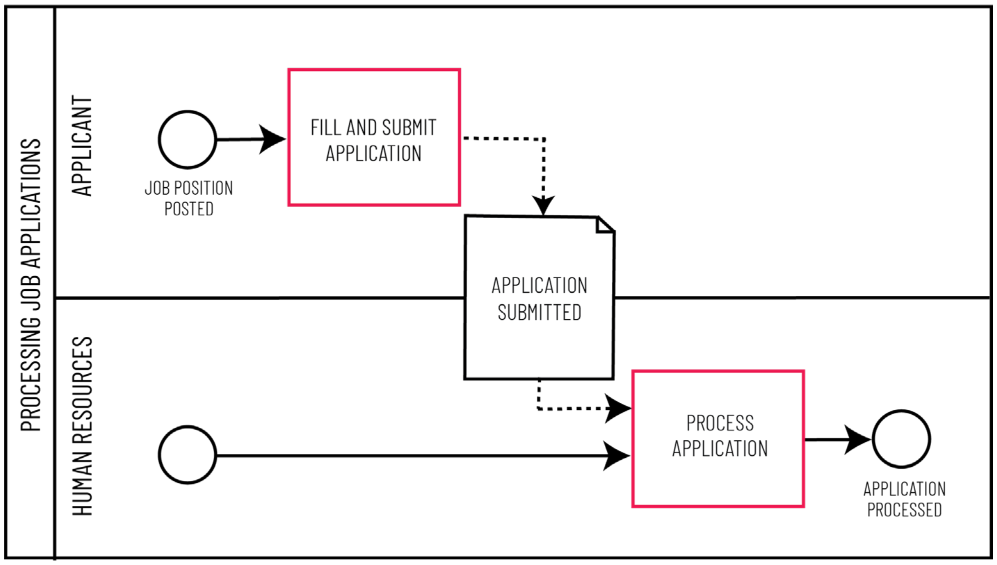

Figure 4: When an activity does not have any incoming sequence flow, BPMN implies a Start Event

Wow! What does that even mean? According to this model, the process can be started either by the Applicant ("Filling and submitting an application" that will never be processed) or by HR ("Processing an application" that came from... nobody? nowhere?). But never by both. It's just not how BPMN processes work. Now I hope you can see why I mentioned earlier "Assuming the Applicant started the process."
It could have also been, "Assuming HR started the process."

Now let's look at how we could have done this correctly.

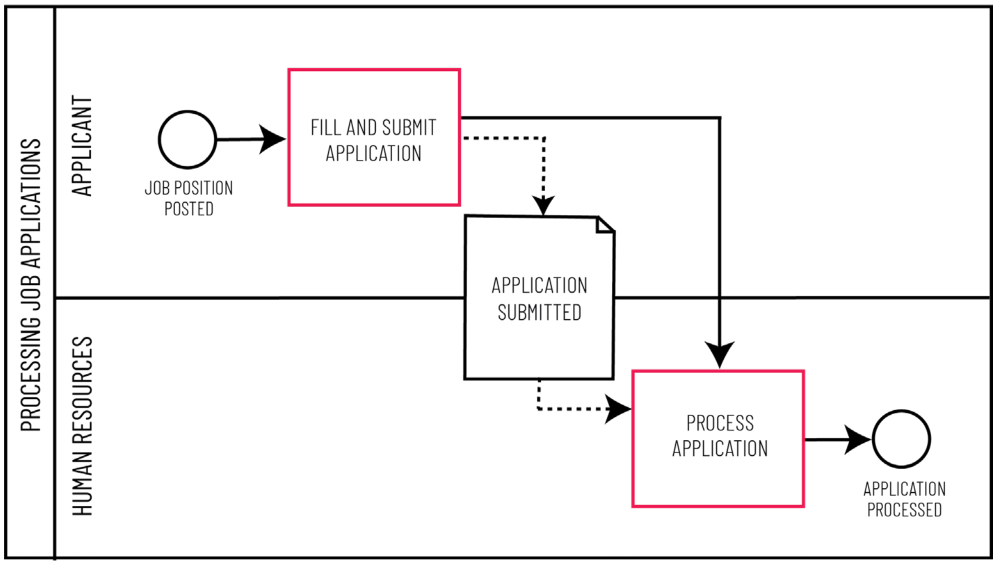

Figure 5: A process model must be continuous, from Start to End Event(s), through Flow elements and Sequence Flow

What's the big difference? One sequence flow connector, right? But that sequence flow makes all the difference.

Note: All these examples are valid from the BPMN point-of-view (syntax). However, only one is correct from the business point of view (semantic).

One last comment on this: some may argue that a better model would have separated the Applicant into a separate process/pool (known as a black-box or abstract pool). I will come back to that later.

Before we move on, one other example of a very common mistake:

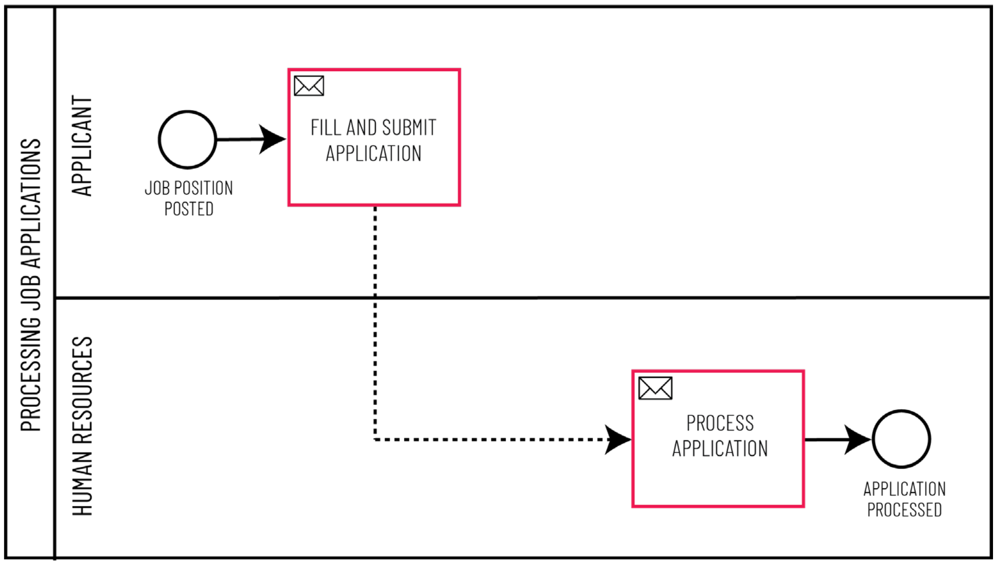

Figure 6: Message Flow is intended for sending/receiving messages between processes, not to pass "tokens" between flow elements within a process (pool). This process' work-flow is broken, too.

It's obvious, isn't it? The applicant sends (perhaps, by email) the application to HR. No, not really. Message Flow is used incorrectly here. At least, this one is "caught" by the BPMN Rules "police" (invalid, since you can't use Message Flow to connect elements in the same Pool).

B. CLEAR

A good process model should be clearly understood by all stakeholders, no matter which domain they represent (business or technical/IT). That, of course, implies that a model must also have the ability/flexibility to "adapt" its presentation depending on who the audience is.

Let's look at some examples (again, using BPMN).

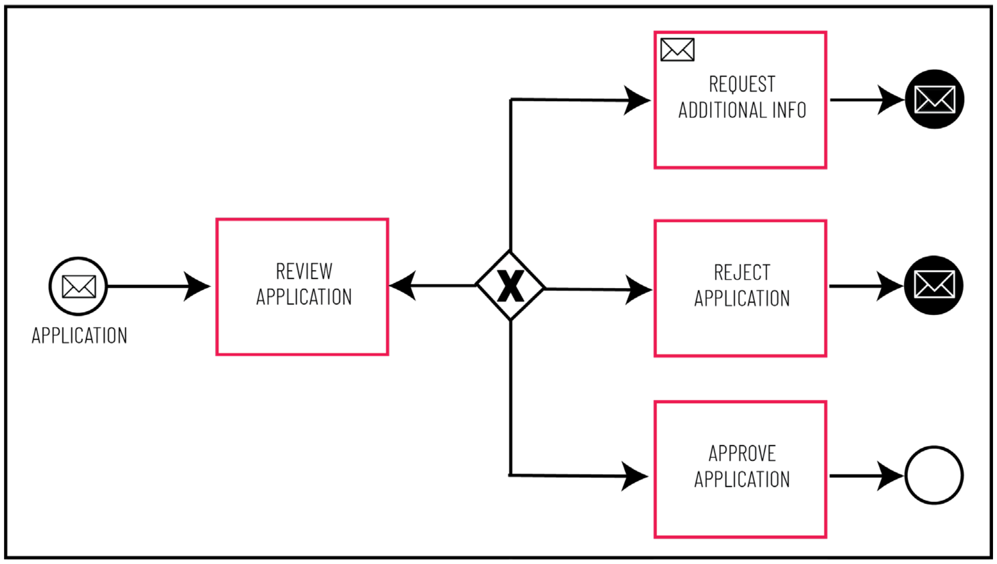

Figure 7: An example of process open for interpretation. Good processes do not rely on readers making "correct" assumptions.

Once again, it looks straight-forward, doesn't it? Well, no.

Here are some possibly confusing/ambiguous "facts" (i.e. assumptions):

- What does the "Application" (Start Event) mean? Does it mean the application was received from the applicant? Or, maybe, did the hiring manager request us to review an application? Or, still, was it because the application was re-submitted? Who knows? Can't tell.

- Under what conditions/criteria did the process route the work to "Request additional info", "Reject application", or "Approve application"? Was it because there was some missing info? Or, because we liked this candidate and, hence, wanted more info? Or was it because we wanted more info from the hiring manager? Who knows? Can't tell.

- If we "Request additional info", why do we have two messages being sent (one by the task, the other as a result at the end of that process branch)? Or is it the same message? Or, maybe, there are two separate messages: one sent to the applicant and the other to the hiring manager. Who knows? Can't tell.

- Finally, what are the end results of this process? Do the two message-sent end events represent the same result (i.e. same state, same message sent)? Or, are they separate end states, each resulting in a message (maybe the same message, maybe a different message) being sent? And if it's approved, is the application also closed? Who knows? Can't tell.

Here is one possible solution to these problems:

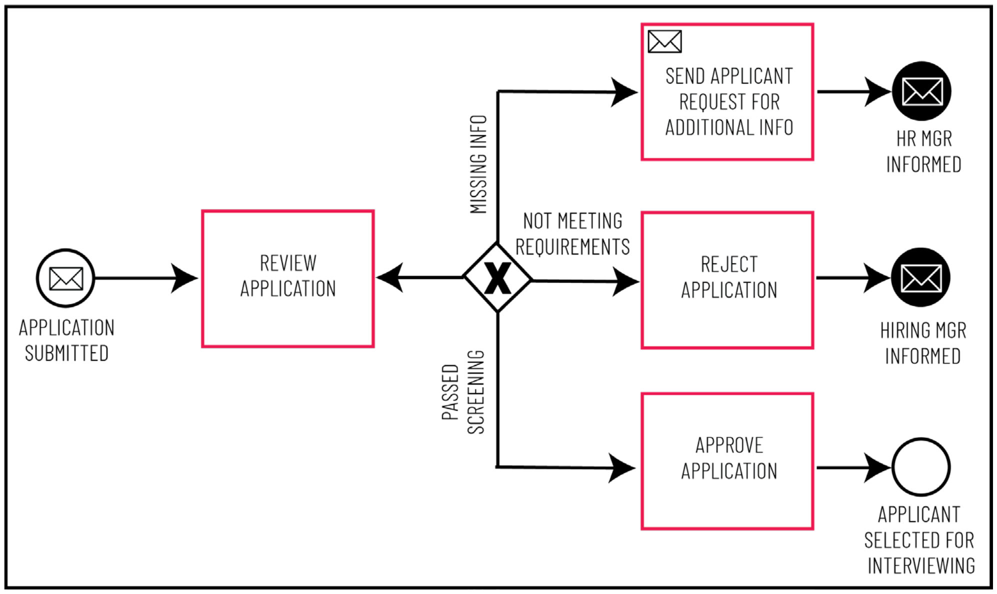

Figure 8: Labels and consistent naming conventions address the ambiguity in the previous version.

Some people may argue that the communication with the applicant (or the HR Manager and the Hiring Manager) is still not clear.

Here is an alternative solution to address the communication with the applicant (for simplicity, this version does not address the argument about the two managers):

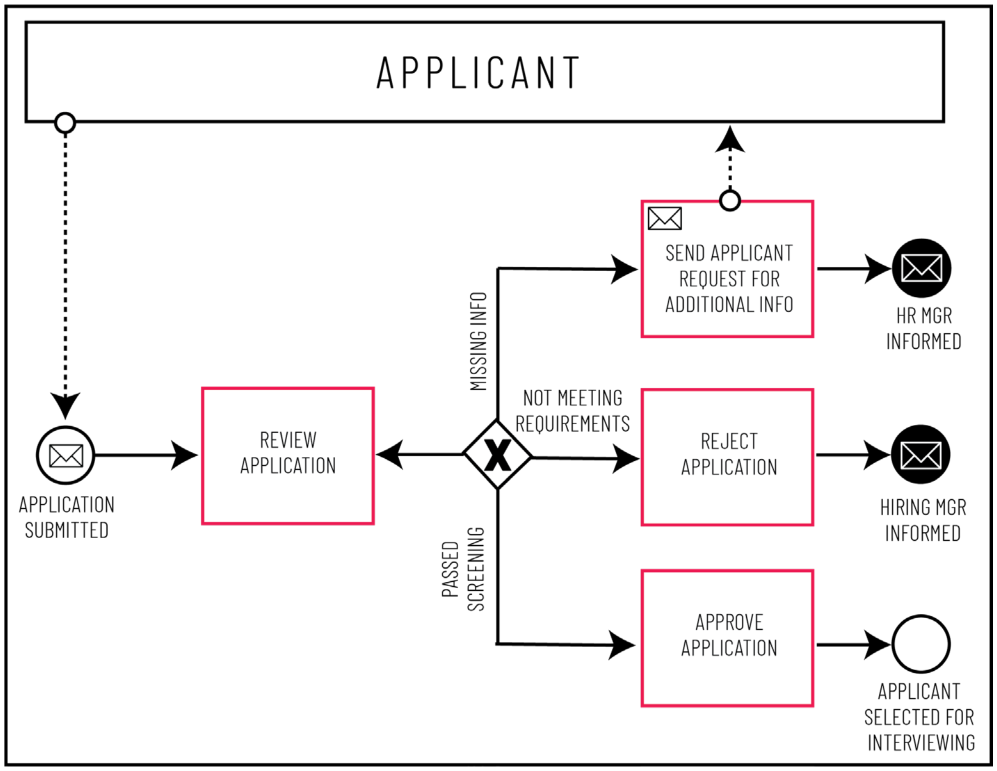

Figure 9: Sometimes it is better to explicitly show how the process collaborates/interacts with external actors (humans or external systems), represented as black-box pools.

Let's look at another example.

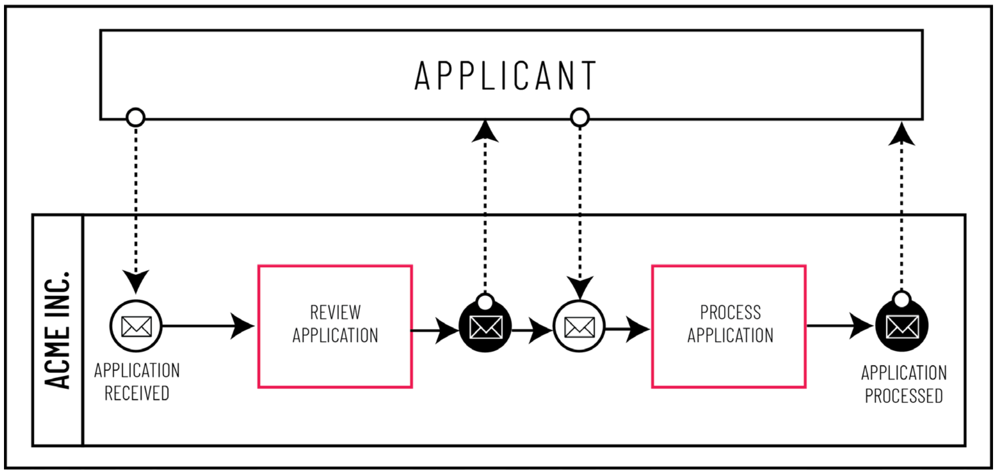

Figure 10: While it is clear who the process "collaborates" with, what messages are being sent to/received from the Applicant?

By now, I'm hoping you can already recognize some of the problems. While it is quite clear that ACME's process starts when an application is received, how can we tell what messages are going back and forth between ACME and the Applicant, after "Review application" and before "Process application"? We can't. Maybe it is a request for additional information. Maybe it is asking if the applicant is still interested in the job, waiting for a confirmation. Maybe it is asking for References. Can't tell.

But, wait, there's more here! What process are we representing here? ACME Inc. probably has hundreds of processes. So, which one is this one? Can't tell.

Let's see how we could fix that.

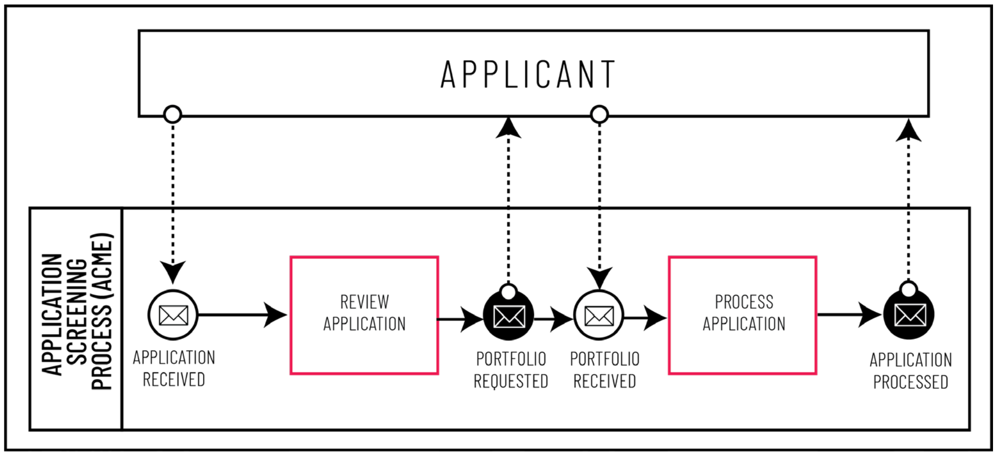

Figure 11: Like before, clear labels and consistent naming conventions make the collaboration clear.

That clarifies our previous questions. But, of course, you might ask "wait a second! what happens if the applicant doesn't send the portfolio?". OK, we will look at that next.

C. COMPLETE

A good process model describes not only the "happy path" (or some paths through the process), but all relevant paths through the process. Each one of these paths is just one of the possible scenarios that define the whole process.

The previous solution was clear. But was it complete? What should happen if the applicant chooses to ignore our request and not respond? What should happen if the applicant takes too long to respond? Let's see if we can clarify that:

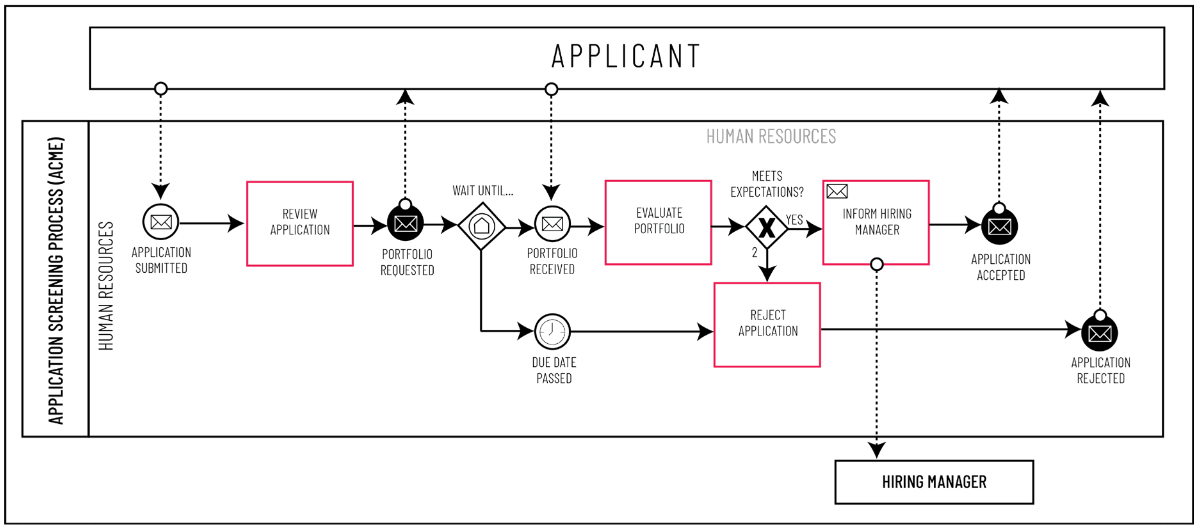

Figure 12: A complete process model must also show alternate and exception paths/scenarios.

Now we know if the applicant ignores our request and does not send us the portfolio before the due date, we will reject the application. Also, we will reject the application if the portfolio doesn't meet our expectations. You might be asking, "What happens if the applicant responds saying they're not interested anymore?", or, "what should the process do if we need some clarification about the portfolio?" I am quite sure you have probably already figured out how to refine the model to reflect these scenarios.

D. CONSISTENT

Good process models are always interpreted the same way (a consistent way) by all readers. Even when seen from different perspectives.

Finally, let's look at one more example (the last, I promise).

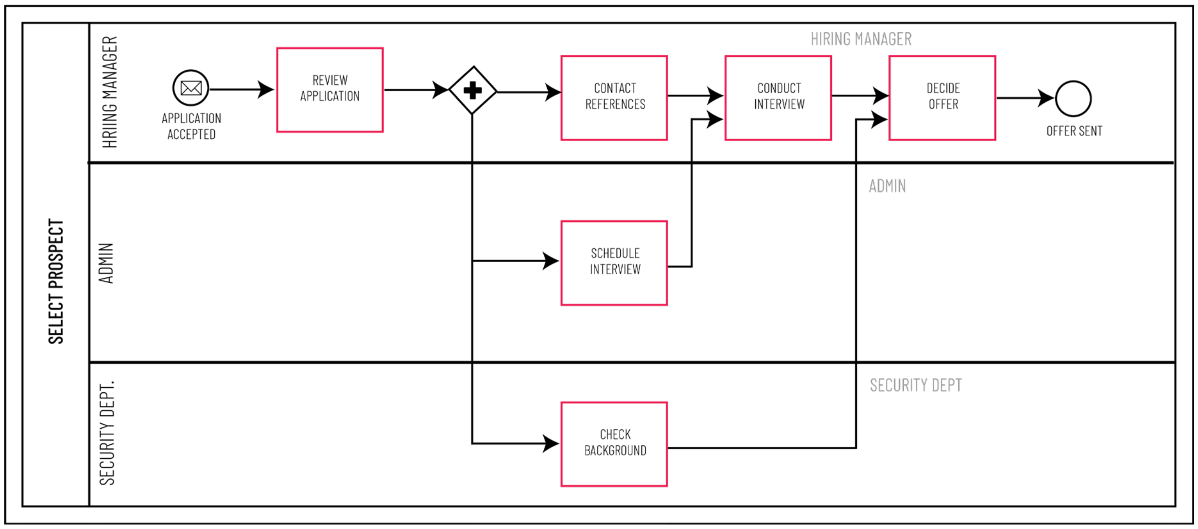

Figure 13: A process model should have only one meaning/semantic... would everybody interpret this model the same way?

First, before you start arguing that this model is incomplete, let's just assume that this is focused on only the main success scenario (aka "the happy path"). Obviously, to be complete we would also have to model the alternate and exception scenarios.

Back to my example, it looks pretty good… or, does it? Let us walk through the process. Once an application was accepted (say, from the Screening process), the Hiring Manager reviews it and, since they want to pursue this application, they contact the references and, in parallel (as indicated by the Parallel gateway), asks the Admin to schedule an interview. Once the interview is scheduled, they interview the candidate. Assuming again that this is the "happy path", they select the candidate (who, some people now call a prospect). However, before deciding what offer to make, the background check must be completed (done by the Security department). Once that's completed, they decide the offer which is then sent to the candidate/prospect.

Right? No, not really. At least not to everybody. Some people will interpret the model that way. Others, however, will interpret it the "BPMN way". According to this model, the Interview would take place twice: one time after "Contact References", and another time after "Schedule Interview". Why? There is nothing in the process model saying that it should wait until both tasks are completed. Similarly, "Decide offer" (and, indirectly, sending that offer) would happen three times: two times after each of the "interview" duplicates, and a third time after "Check background".

Not quite what we intended to say, isn't it?

Here is an alternative version that addresses those "mis-interpretations":

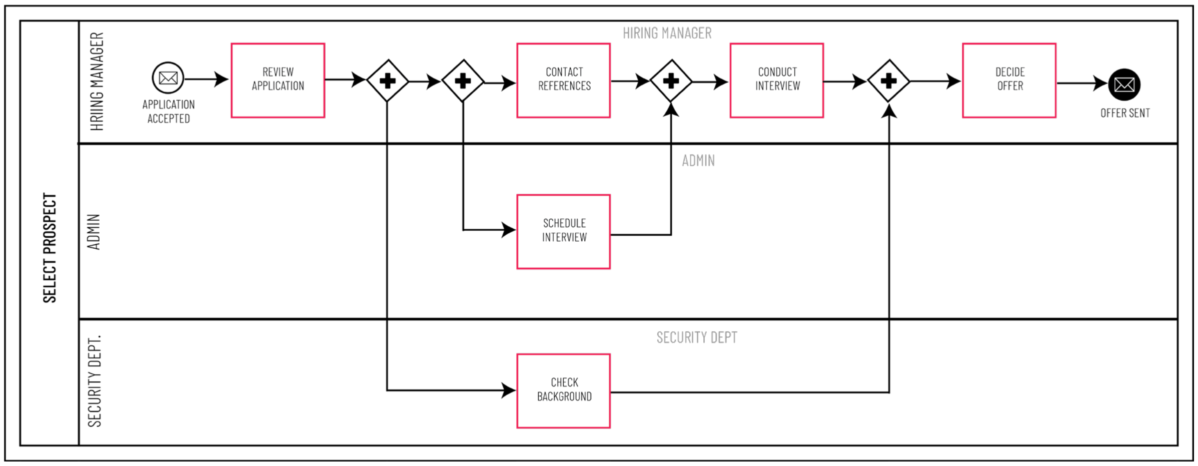

Figure 14: Pairing the gateways correctly prevents readers interpreting the model to work differently, depending on different assumptions each reader makes.

Some people may suggest a “simplified” version:

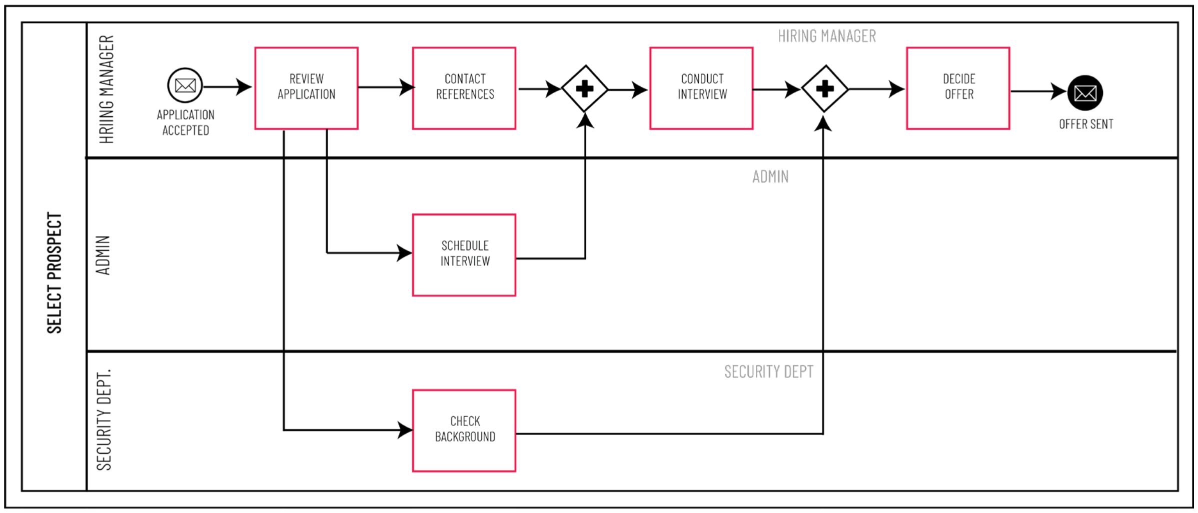

Figure 15: Good process models are concise, too, as long as making them concise does not change the way the process works (but, at the same time, be careful to not confuse your audience).

This version is correct, too (behind scenes, the BPMN model is “saved” in an XML format, which in this case should be semantically identical). However, removing any of the join-gateways would cause the process to behave incorrectly again.

Also, keep in mind that some people may be confused by this “short-format” version. I’d just encourage you to use common sense and good judgement.

CONCLUSIONS

So, what was all this about?

- When people complain that BPMN is too complex and/or not easy to understand, they probably do not understand how to use BPMN and, more problematic, how to build good, understandable BPMN model
- BPMN models that are not built correctly are unnecessarily complex and very confusing. Unfortunately, most of the BPMN models out there are bad models. But instead of blaming BPMN, we should learn how to do BPMN right.
- Building good BPMN does not require one to learn programming (or rocket science). It requires knowledge, expertise, and skills that everyone can learn. But it is not something we are born with.

I left out all sorts of interesting details (e.g. Data Objects/Data Stores, Inclusive Gateways, Event Subprocesses, Workflow patterns, etc.). And, yes, even BPMN weaknesses (yes, there are also inconsistencies and weaknesses in BPMN). But I will come back to almost all of them in future articles. I am hoping you will join me again on this interesting journey.

FUTURE TOPICS TO LOOK FOR

- What is the best method to build good process models?
- Have you seen Spaghetti processes? Can DMN (along with BPMN) help us untangle that mess?
- Can you analyze "Descriptive level models"? Can you describe "Analytical level models" The problem with OMG's classification into "Descriptive" and "Analytical" models?
- Business Processes vs. Capabilities. What's the big fuss?
- What is the difference between Boundary Events and Event Sub-Processes? When should you use them?
- What's best, flat or hierarchical processes? Does your notation support both? Should you even consider using both?
- Business Processes vs. Value-Chains. Why do we need both?
- Are business processes and process landscapes the same?
- BPMN and Process Improvement/Re-engineering. Pitfalls and best practices.
- BPSim Standard: Validating, analyzing, and exploring processes through BPMN and simulation
- Will S-BPM replace BPMN? Or will we use them both?

REFERENCES

- Haisjackl, Cornelia, Jakob Pinggera, Pnina Soffer, Stefan Zugal, Shao Yi Lim, and Barbara Weber. 2015. *"Identifying Quality Issues in BPMN Models: an Exploratory Study."* Lecture Notes in Business Information Processing book series (LNBIP, volume 214). Springer, Cham. 217-230.

- Kossak, Felix, Christa Illibauer, Verena Geist, Jan Kubovy, Christine Natschläger, Thomas Ziebermayr, Theodorich Kopetzky, Bernhard Freudenthaler, and Klaus-Dieter Schewe. 2014. *A Rigorous Semantics for BPMN 2.0 Process Diagrams.* Springer International Publishing.

- Leopold, Henrik, Jan Mendling, and Oliver Günther. 2016. *"Learning from Quality Issues of BPMN Models from Industry (Extended Abstract)."* Proceedings of the 7th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA 2016). Vienna, Austria: CEUS-WS.org. 36-39.

- Object Management Group. 2014. *"About the Business Process Model and Notation Specification Version 2.0.2."* Object management Group. January. https://www.omg.org/spec/BPMN.

- OMG. 2010. *"BPMN 2.0 Examples."* Object Management Group Portals. June. http://www.omgwiki.org/bpmn2.0-ftf/lib/exe/fetch.php?media=public:sub-teams:bpmn_2.0_by_example_version_rc.pdf.

- Silingas, Darius, and Edita Mileviciene. 2007. *"Refactoring BPMN Models: From 'Bad Smells' to Best Practices and Patterns."* ResearchGate. January. https://www.researchgate.net/publication/280547761_Refactoring_BPMN_Models_From_'Bad_Smells'_to_Best_Practices_and_Patterns.

- Silver, Bruce. 2012. *BPMN method and style.* Aptos, CA: Cody-Cassidy Press.

ABOUT THE AUTHOR

Razvan Radulian brings over 20 years of business and IT experience both as an Analyst and as Project Manager. He is an expert in Business and Enterprise Architecture, Business Analysis, Business Process Analysis, User Experience, OOAD and UML within the Pharmaceutical, Clinical Trials, Clinical Testing, Non-Profit and Training industries. Razvan has used a wide-range of methodologies from highly formal like Waterfall to highly adaptive like Agile and Scrum.

Razvan is an active speaker in local and global professional organizations including International Institute of Business Analysis (IIBA), Business Process Simulation Working Group and is a mentor for MicroMentor.

ABOUT CPRIME

An Alten Company, Cprime is a global consulting firm helping transforming businesses get in sync. Cprime is the partner of choice for Fortune 100 companies looking to achieve value and agility. We help visionary business leaders compose solutions, execute implementations, and exceed against business goals. With our key partnership recognitions, including Atlassian Platinum, AWS Advanced, and SAFe Gold SPCT partner, our industry-leading software and services work in synergy to deliver transformations.

Visit us at **cprime.com**