Rethinking BPMN, Part 2:
A GOOD Process to Model Processes

Benefits, Approach, Lessons

Table of Contents

Why do we need a process to…
…model processes?

1. Purpose

- Process' Purpose
- Modeling's Purpose

2. Head 'n Tail

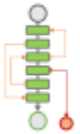
Process Boundaries:

- Start (Pre-Conditions, Trigger)
- Ends (Successful outcome, Alternates, Exceptions)

3. Backbone (Head to Tail)

Main Success Scenario/Path:

- Key activities linking Start to Successful End
- Identifying branching points & conditions (without branching!)
- Identifying "Unsuccessful" End(s)



4. Skeleton (Backbone + Rest of it)

Branching into:

- Alternate scenarios/paths: linking back to MSS
- Exception scenarios/paths: ending the process

5. Body Structure/Hierarchy: Arm to Hand to Fingers

"Breaking" down Processes into Sub-Processes:

- Hierarchical process structure
- Processes to Sub-Processes

6. Whole Body: Skeleton + Muscles + Nervous Sys

Linking Processes to other Business components:

- Organizational Roles/Structure
- Business Data
- Business Policies, Decisions, Rules
- User Interfaces, etc.

7. Communities

- Process to Process Collaboration

Conclusions

- Should have a process (this or yours)
- Should be consistent
- Should be effective, efficient, and flexible

Why do we need a process to…
…model processes?

What happens if we don’t have an approach?

- Miscommunication
- Scope Creep
- Incorrect or inappropriate levels of details
- Wrong audience and/or collaborating with wrong stakeholders
- Misunderstanding
- Missing or incorrect activities, events, branches, etc.
- Analysis Paralysis & lack of progress
- Lots of unverified Assumptions

… need more?

So, what's this approach?

Note: using an analogy from how human bodies "work"…

1. **Understanding "the purpose":**
   - Process's purpose
   - Modeling's purpose
2. **Head 'n Tail(s):**
   - Start & End Events
3. **Backbone (Head TO Tail):**
   - Activities linking Start to main success End states (Main Success Scenario)
   - Branching points & conditions (without branching!)
4. **Skeleton (Backbone + rest of skeleton):**
   - Alternate & Exception branches/scenarios
5. **Body Structure/Hierarchy: Systems to Sub-Systems (Arm to Hand to Fingers)**
   - Zooming-in/out from Processes to Sub-Processes
6. **Whole body (Skeleton + Muscles + Nervous systems):**
   - Processes + Organizational Roles/Structure, Business Data, Policies/Decisions/Rules, and/or more
7. **Communities:**
   - Process to Process Collaboration

Example: Buy Books (online)

To demonstrate the concepts in this presentation, we will use a process example that probably everybody can relate to:

- Buying books online (e.g. like on Amazon.com)

1. Purpose

- Process’ Purpose
- Modeling’s Purpose

The “Purpose”

Actually, two (2) purposes:

- **Process’s purpose:**
  the expected outcome of the purpose
  - “Buy books”
- **Our modeling’s purpose:**
  the reason for which we model the purpose
  - Understanding/communication?
  - Analysis/improvement?
  - Design/implementation?
  - Monitoring/management?
  - Other reason?

Why does that (i.e. purpose) matter?

Depending on that reason, or combination of reasons, our modeling may be:

- **More/less detailed**
  - Overall/everywhere in the process?
  - Only certain areas of the process?
- **More/less formal**
  (e.g. using templates, organization standards/guidelines, etc.)
- **Focused on:**
  - either current (As-is),
  - or future/desired (To-Be) process,
  - or on balancing the current and future states of the process

Example: Buy Books (online)

Let’s assume the modeling purposes for our example are:

- Understand how the process works (or how it should work)
- Communicate our understanding with other stakeholders
- Getting stakeholders (business or technical) to review the model and provide feedback (e.g. validation, questions, suggestions, feasibility assessments, etc.)

Note: if our purpose were to analyze and/or simulate the process, we would need more details (i.e. more BPMN details and/or other details, like Time, Cost, Resource availability, and other details; for more info on this, you may want to check the BPSim standard).

2. Head ‘n Tail

Process Boundaries:

- Start (Pre-Conditions, Trigger)
- Ends (Successful outcome, Alternates, Exceptions)

Process Boundaries: Start & End Events/States

To avoid scope creep, it is important to describe the boundaries/scope of the process we want to model:

- **Start Event:**
  - pre-conditions/state
  - what triggers the process
- **End Event(s):**
  - successful results/outcome
  - exception results/outcomes (optional at this stage)

If these boundaries are not explicitly identified/defined, people make different assumptions about when/how the process starts and ends, which causes lots of miscommunication, frustration, and/or waste of resources.

Example: When/how does “Buy Books” Start?

In our example, “Buy books” can start when:

- Customer attended a webinar mentioning a book/some books
- Customer thought of buying a book (on a particular topic)
- Customer searched and found a book (on a particular or related topic)
- Customer read a review or recommendation of a book
- Customer added a book to the shopping cart
- Customer chose ‘Check out’
- Customer payed for the order
- The order was shipped
- Etc.

Example: When/how does “Buy Books” End?

Similarly, the life of a “Buy Books” order can end when:

- Order payment was authorized
- Book(s) was/were retrieved from the warehouse
- Book(s) was/were packaged
- Package is ready to be picked up (by Shipping company)
- Package was picked-up (by Shipping company)
- Package was delivered
- Delivery was confirmed (by Shipping company)
- Book(s) was/were read
- Book(s) was/were given away (hopefully, to somebody else or to be recycled)
- Later?

Example: “Buy Books” chosen Start & End(s)

Start

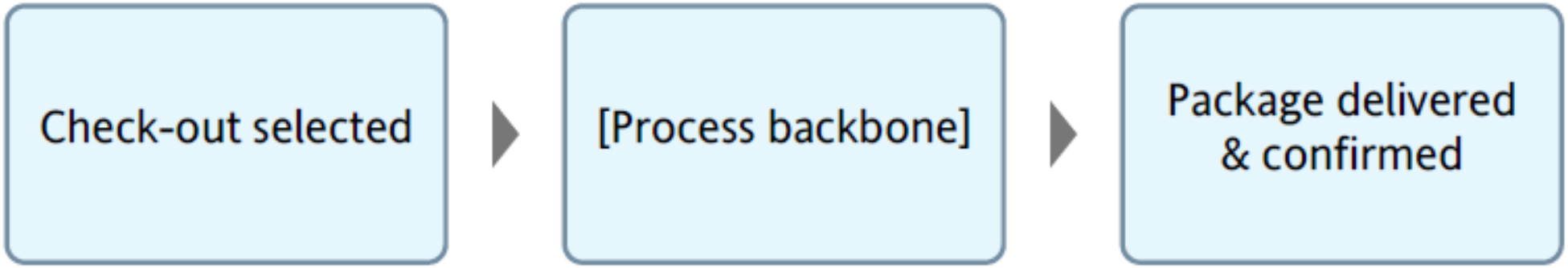
Customer selected “Check-out”

End

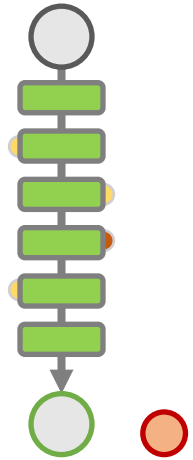
“Shipping Company has been delivered the package and confirmed delivery”

“Buy Books” - Start & End Diagram?!?

In MS Office



In IBM Blueworks Live



3. Backbone (Head to Tail)

Main Success Scenario/Path:

- Key activities linking Start to Successful End
- Identifying branching points & conditions (without branching!)
- Identifying “Unsuccessful” End(s)

The Process Backbone: Head TO Tail

When/where/who:

- Identify **key activities** along the main success scenario path (backbone)
  - Typically, in interviews and workshop with stakeholders
- Optionally, group them by **phases/milestones**
  - Often done by the analyst after the interviews and/or workshops

Typically, there are:

- 3 to 10 stages/milestones per backbone
- 3 to 10 activities per milestone
- 2 to 3 hierarchical level (i.e. child sub-groups/sub-processes within top/parent groups/processes)

For each activity, identify:

- **Normal/successful next step:**
  our focus right now, since this sequence defines the “backbone”
- **Alternate next/end state(s)**:
  we will elaborate these later

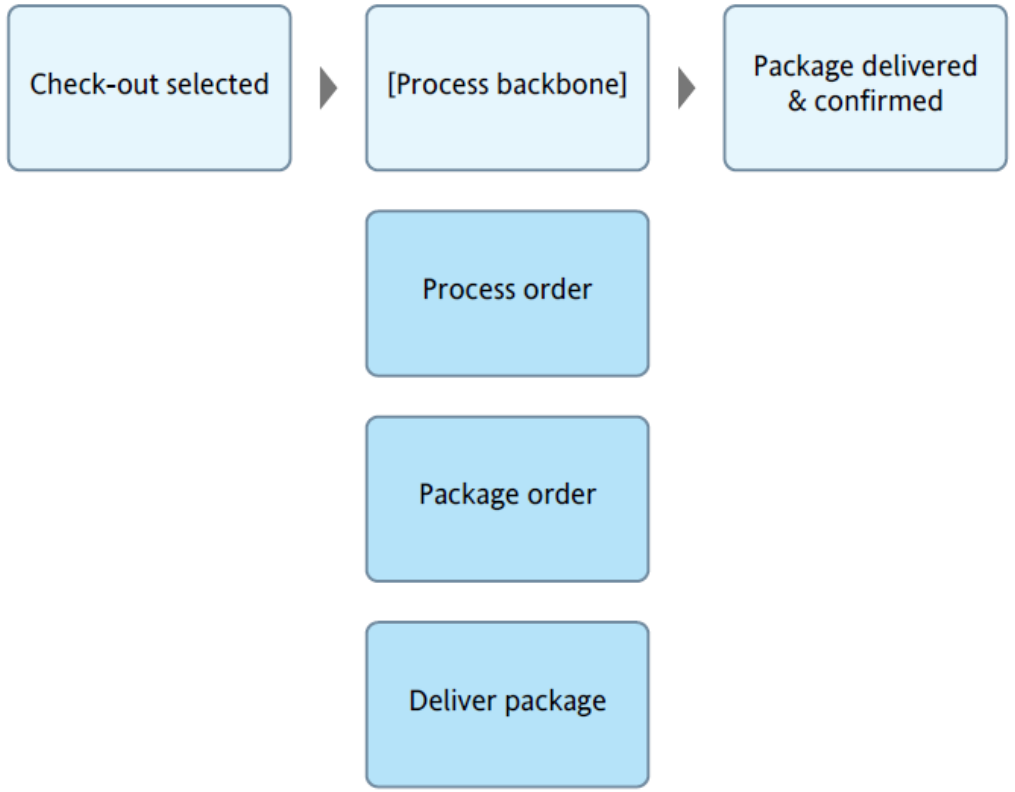
Example: “Buy Books” – Backbone…

We can easily identify 3 stages/milestones for the “Buy books (online)” process:

1. Process order
2. Package order
3. Deliver and confirm package

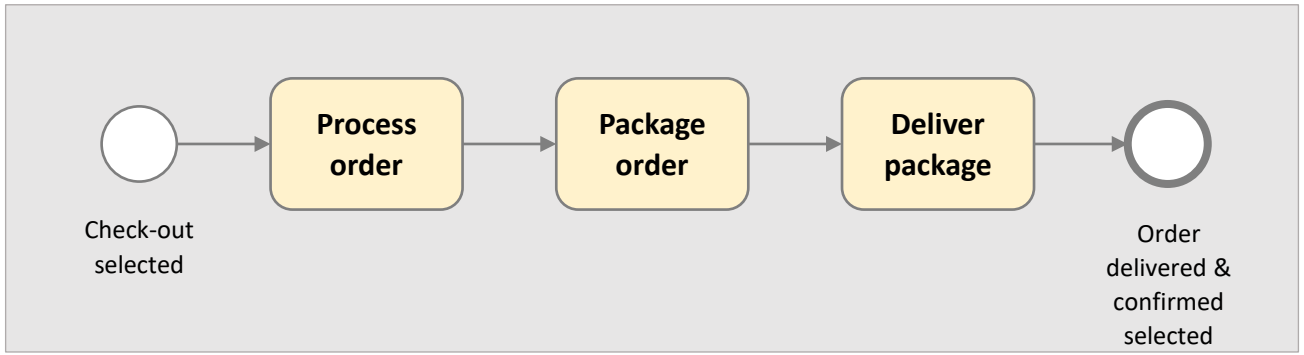
Step	End states (desired Success, Alternate, eXception)
Process order	**Order processed (S)**
	Item out of stock (A or X)
	Item out of print (A or X)
	Payment declined (A or X)
	…
Package order	**Order packaged/shipping co. notified (S)**
	Item missing or damaged (A or X)
	Missing/wrong packaging material (A)
	Shipping co. not available/busy (A or X)
	…
Deliver package	**Package delivered and confirmed (S)**
	Package lost/damaged (A or X)
	Package delayed (A or X)
	Can’t deliver/wrong address (A or X)
	Package refused (X)
	…

Example: “Buy Books” – Backbone as Diagrams

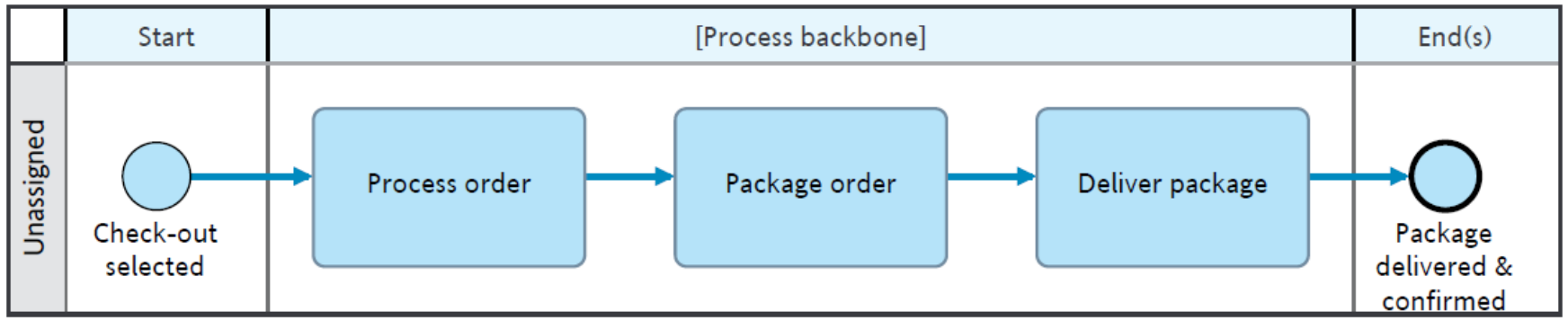


In IBM Blueworks Live
(as Discovery Map)

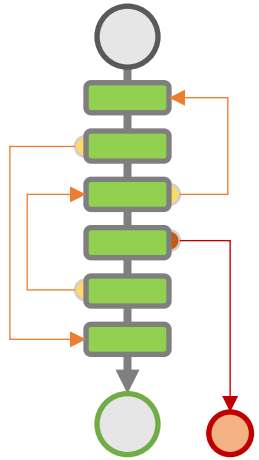
Example: “Buy Books” – Backbone as Diagrams



In MS Office



In IBM Blueworks Live
(as Process Diagram)



4. Skeleton (Backbone + Rest of it)

Branching into:

- Alternate scenarios/paths: linking back to MSS
- Exception scenarios/paths: ending the process

Skeleton: Backbone + Rest of it

Two kinds of branches (along the backbone):

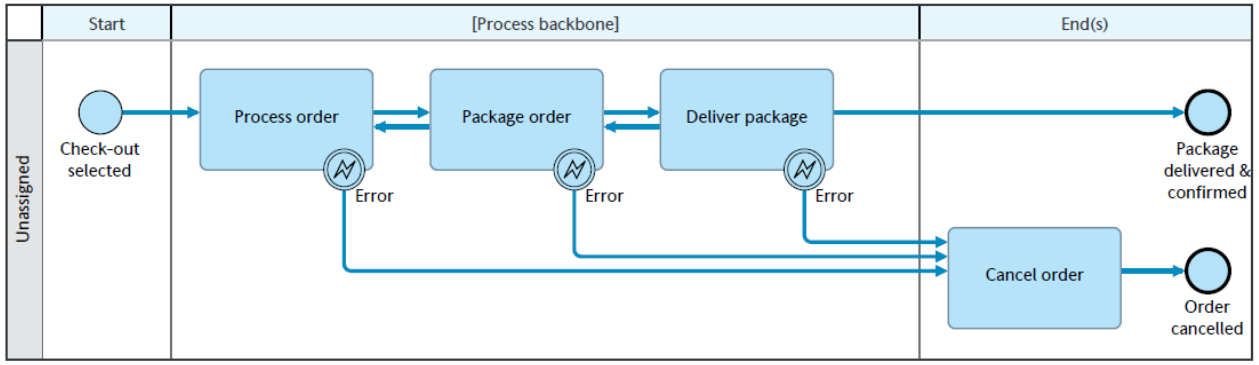
- **Alternate paths/scenarios**: branches that join back into the backbone
- **Exception paths/scenarios**: branches which will end the process

Example: “Buy Books” – Skeleton (in a table)

Now, that we know “all” the steps along the backbone (we might still get some “surprises”, as we elaborate and refine our model), we can describe when the process branches off the main path, indicating the branching condition and how it will be handled (as next step in the process)…

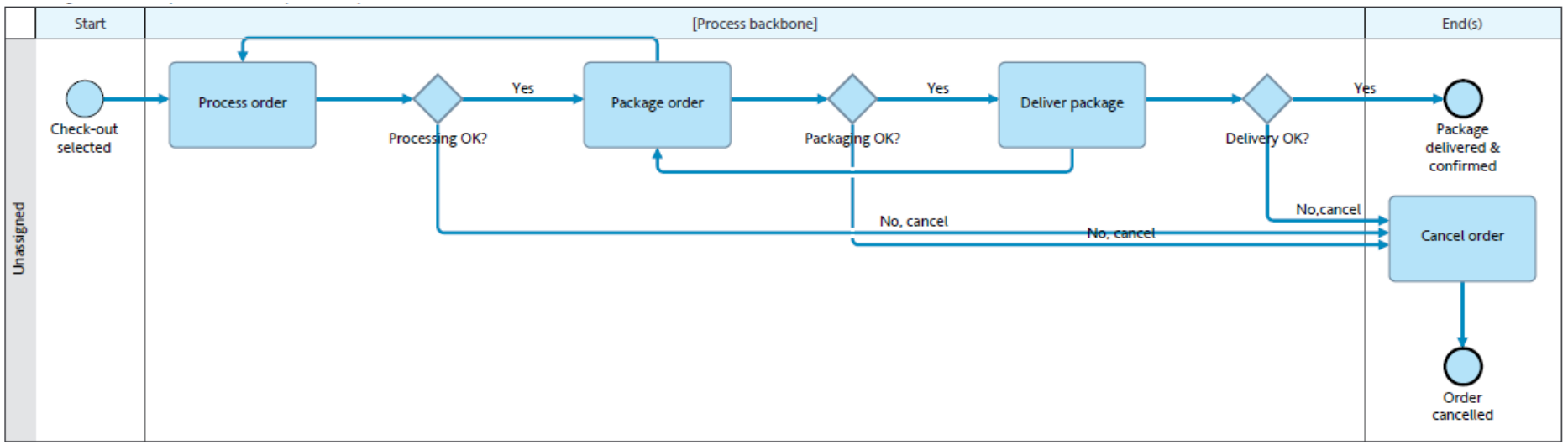
Step	End states (desired Success, Alternate, eXception)	Next step (n:main success/normal path; h:handle inside the activity; e:escalate to activity or end event in parent process)
Process Order	**Order Processed (S)**	**Package order (n)**
	Item out of stock, Backorder (A)	Restock item (h)
	Item out of stock, Substitute (A)	Substitute item (h)
	Item out of stock, Cancel (X)	Cancel Order (e)
	Item out of print, Buy used (A)	Replace with used item (h)
	Item out of print, Substitute (A)	Substitute item (h)
	Item out of print, Cancel (X)	Cancel order (e)
	Payment declined, Retry (A)	Process alternate payment (h)
	Payment declined, Cancel (X)	Cancel order (e)
	…	
Package order	**Order packaged/shipping co. notified (S)**	**Deliver & confirm package (n)**
	Item missing or damaged, Get another one (A)	Replace item (h)
	Item missing or damaged, Buyer wants substitute (A)	Substitute item (h)
	Item missing or damaged, Cancel (X)	Cancel order (e)
	Missing/wrong packaging material, Get missing material (A)	Substitute packaging (h)
	Shipping co. not available, Notify alternate carrier (A)	Swap shipping carrier (h)
	…	
Deliver package	**Package delivered and confirmed (S)**	**Package delivered & confirmed (n)**
	Package lost/damaged, Reorder (A)	Reorder item(s) (h)
	Package lost/damaged, Cancel (X)	Cancel order (e)
	Package delayed, Late delivery (A)	Expediate delivery (h)
	Package delayed, Cancel (X)	Cancel order (e)
	Can’t deliver/wrong address, New address (A)	Redirect delivery (h)
	Can’t deliver/wrong address, Cancel (X)	Cancel order (e)
	Package refused, Cancel (X)	Cancel order (e)
	…	

Example: “Buy Books” – Skeleton (as Diagrams)



Using Boundary Events

In IBM Blueworks Live
(as Process Diagrams)



Using Exclusive Gateways

5. Body Structure/Hierarchy: Arm to Hand to Fingers

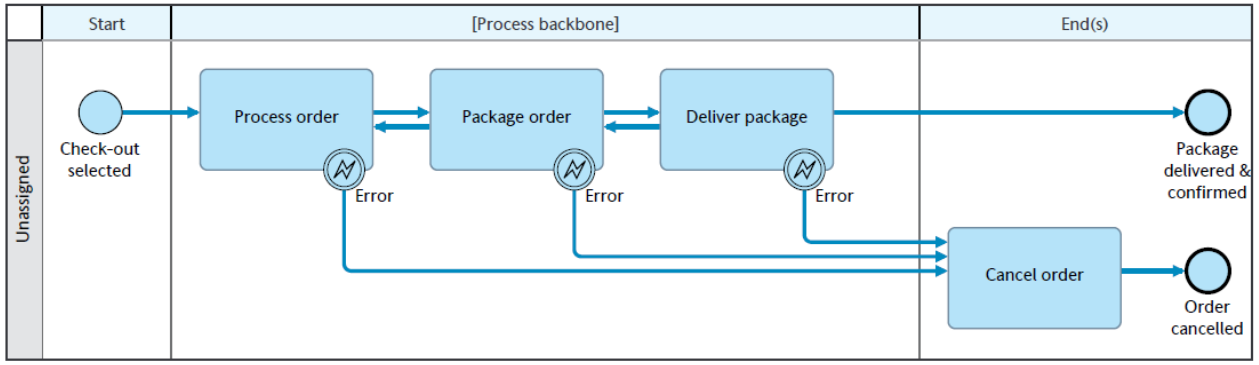
“Breaking” down Processes into Sub-Processes:

- Hierarchical process structure
- Processes to Sub-Processes

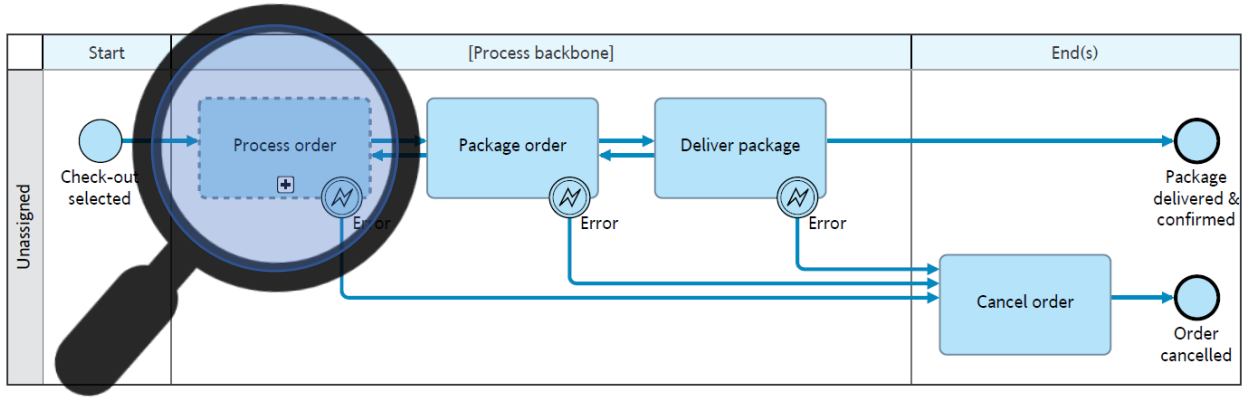
Body Structure/Hierarchy

Based on what we have captured up to this point, it is quite likely that most, if not all, the key activities identified so far are themselves sub-processes and we can/should zoom-in and “see” (elaborate) them in more details.

Example: Zooming-in “Process order”…

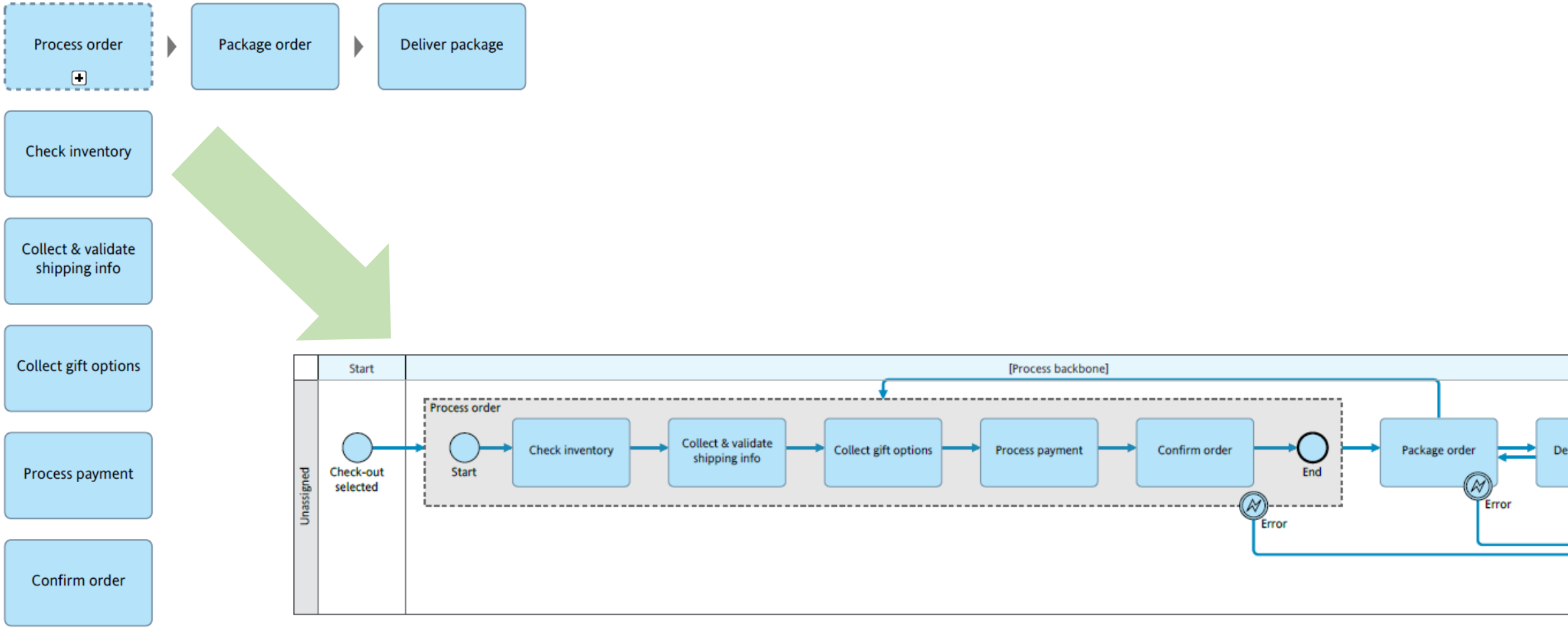


Previous view (high-level)…



Zooming-in…

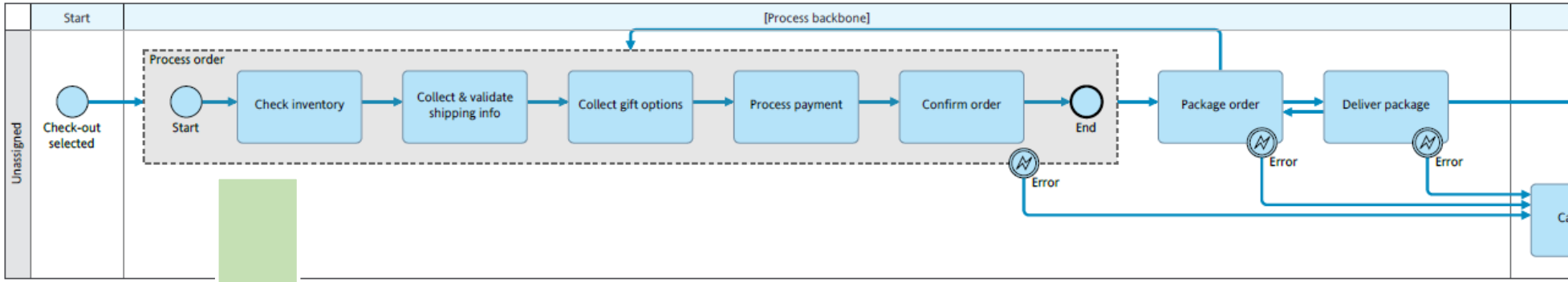
Example: Zooming-in “Process order”, 1st “draft”…

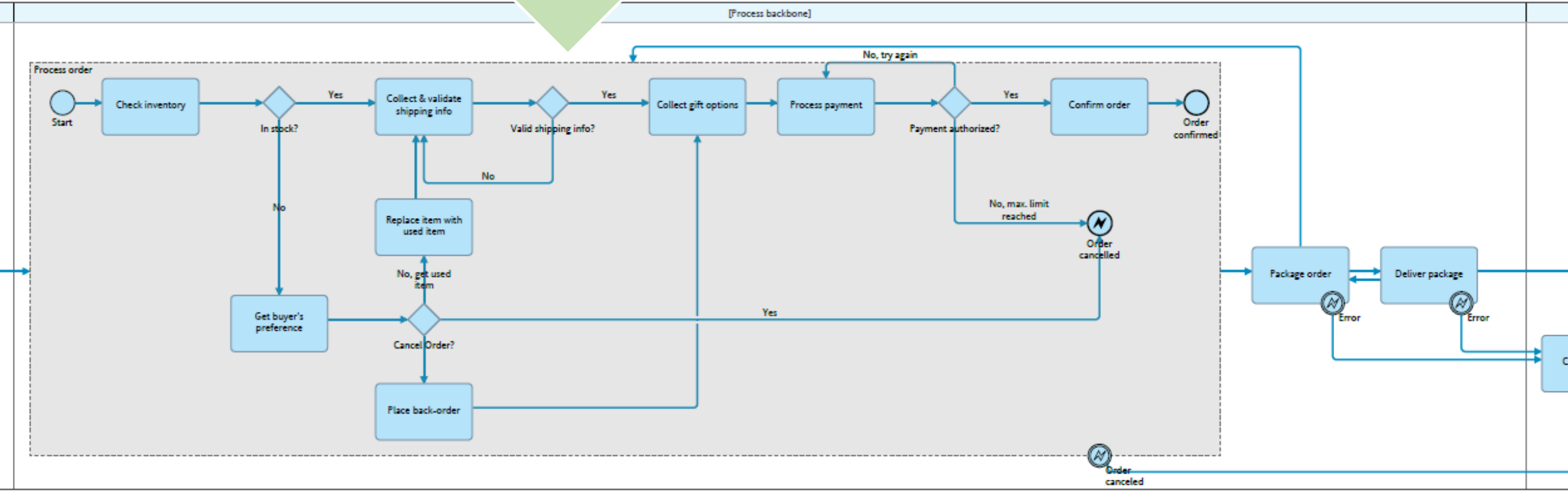


In IBM Blueworks Live
(in the Discovery Map)

In IBM Blueworks Live
(in the Process Diagram)

… zooming-in “Process order” (after elaboration)





Zooming even more: “Process Payments”…

In our example, “Process payment” could be further broken-down, on the “normal path”, into:

- Collect payment info
- Validate payment info
- Request authorization (by Bank [External])
- Confirm payment

and some possible alternate and/or exception paths:

- Invalid payment info (handle internally: re-collect payment info)
- Payment declined, by the bank (handle internally: re-collect payment info)
- No other payment options (escalate to parent process, “Process order”)

6. Whole Body:
Skeleton + Muscles + Nervous Sys

Linking Processes to other Business components:

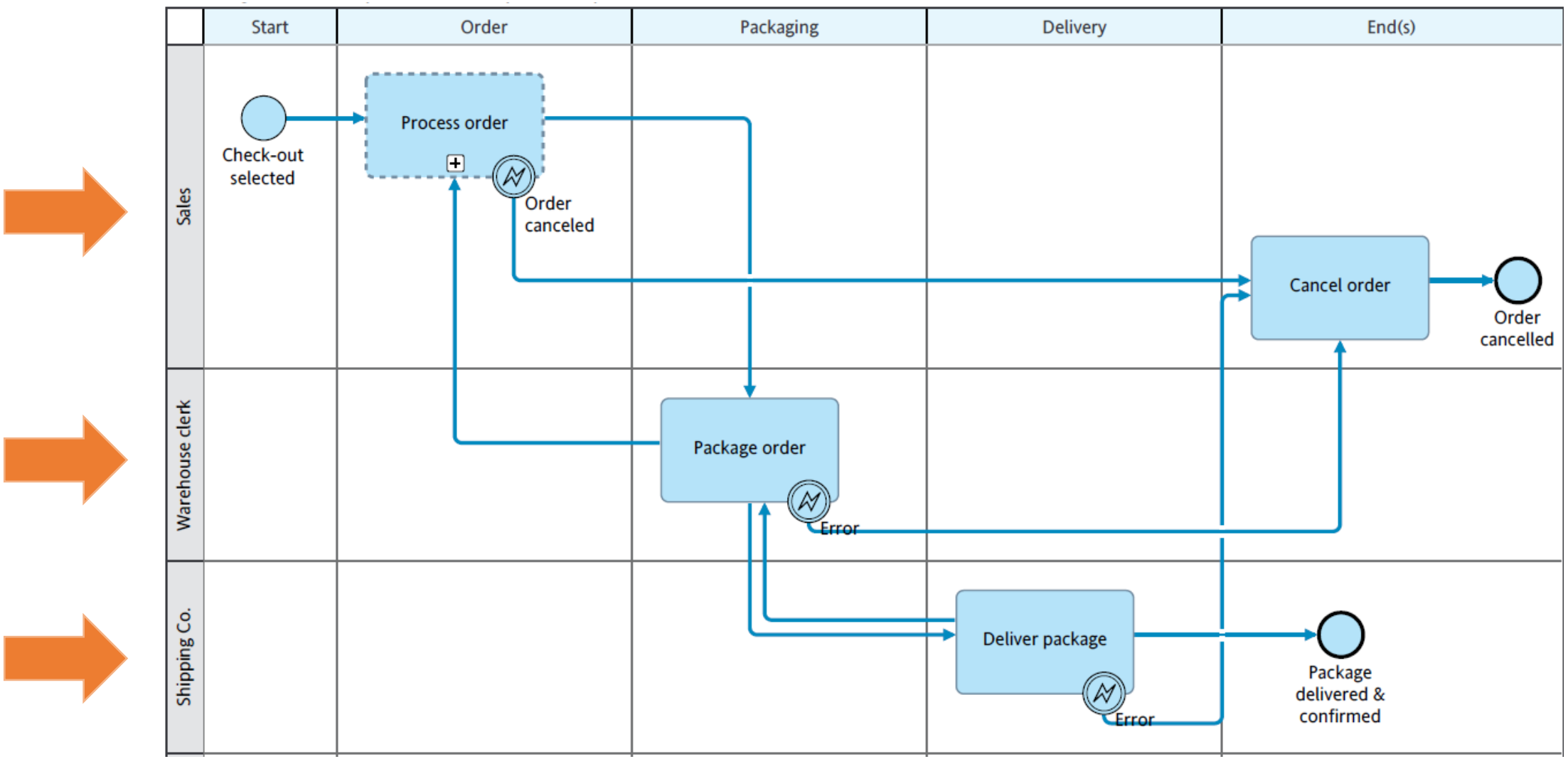
- Organizational Roles/Structure
- Business Data
- Business Policies, Decisions, Rules
- User Interfaces, etc.

Whole Body: Skeleton+Muscles+Nervous systems

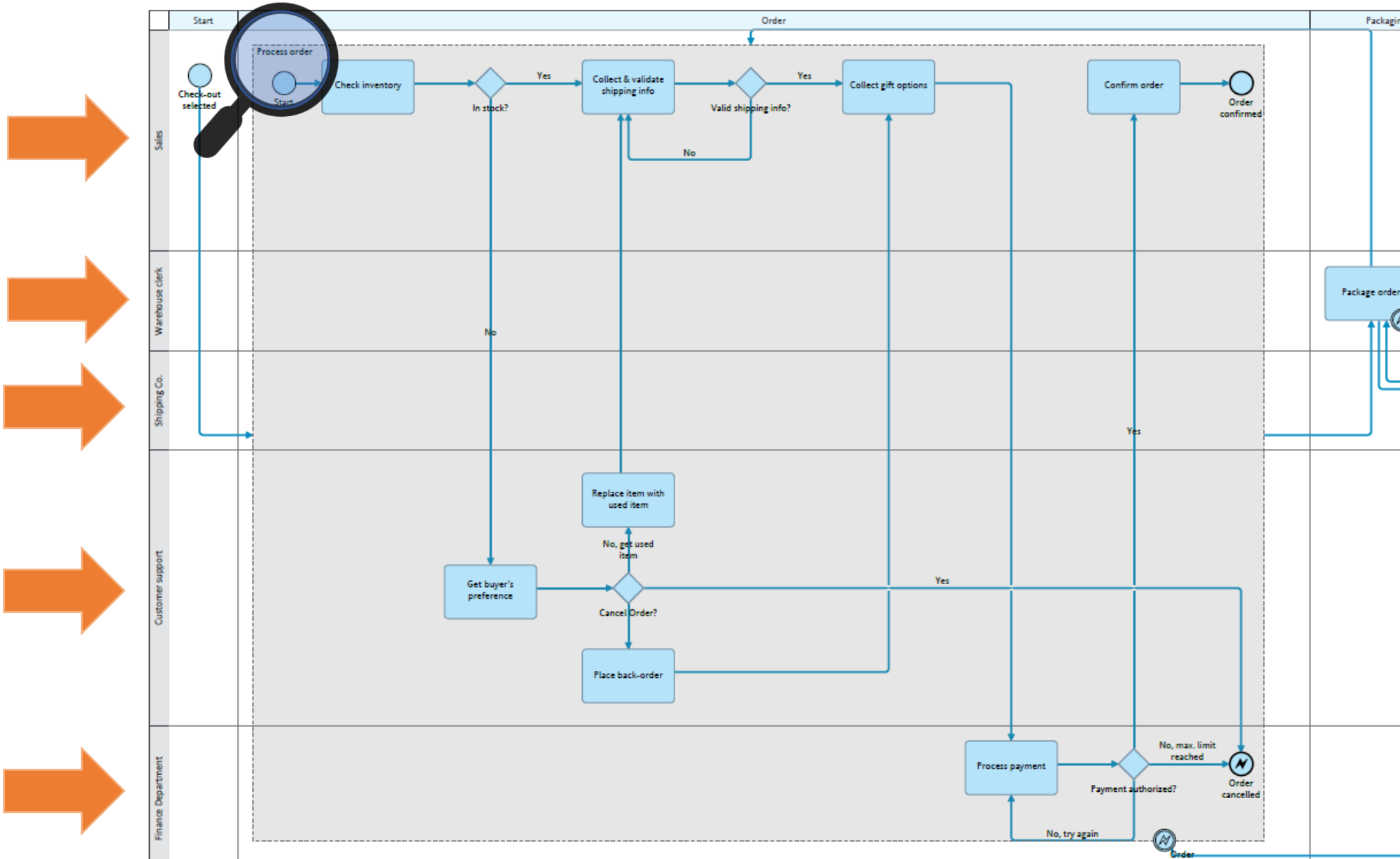
Complete the picture by attaching other relevant elements to the process:

- Actors/organizational structure
- Business data
- Business rules/decisions
- User interfaces
- etc.

Example: Adding Actors/Roles (as Lanes)…



Example: Actors/Roles (at sub-process level)



7. Communities

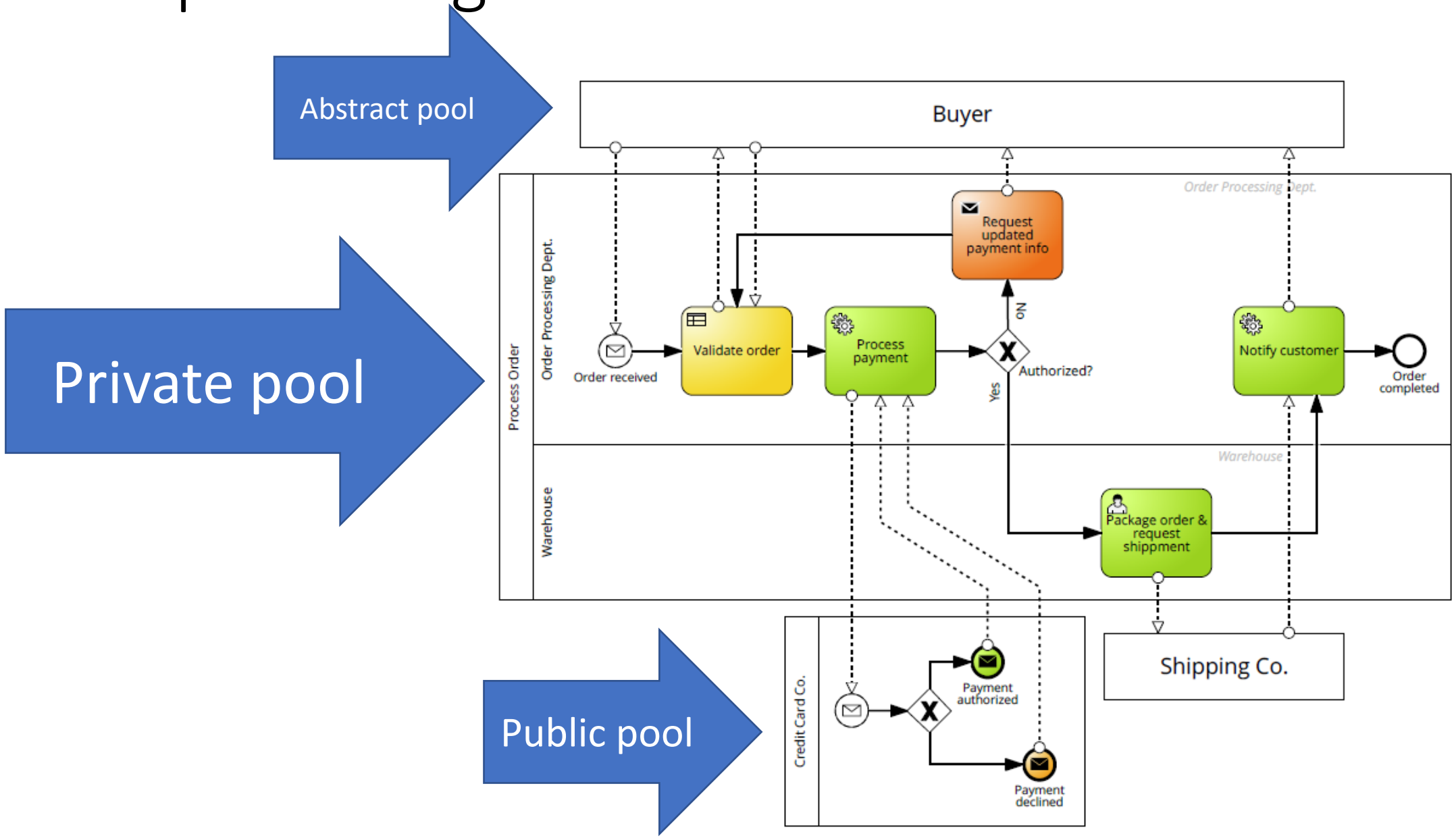
- Process to Process Collaboration

Communities: Process to Process Collaboration

When multiple processes need to collaborate (to accomplish a more complex purpose/goal), describe how processes (Pools) collaborate/coordinate work using messages sent/received between pools (Message Flow):

- Private pools (orchestrated)
- Abstract (“black”) Pools
- Public pools

Example: Using variant of “Process Order”



Conclusions

- Should have a process (this or yours)
- Should be consistent
- Should be effective, efficient, and flexible

WHO? WHEN? WHERE?

WHO:

- Analysts collaborating with Business & Solution Stakeholders

WHEN/WHERE:

- Iterative & incremental face-to-face sessions (WHEN), followed by analysts capturing (often done at their own desk) and sharing/getting feedback (from stakeholders) using process modeling tools (WHERE)

Common challenges:

Too many variations within one process

“Too many branches/variations, do we model them all?”

When teams get stuck in analysis paralysis trying to model all variations in a process (at one time):

- Pick the most common variation (usually, the main success scenario) and model that one first
- Then, handle the other variations, one at a time

Common challenges:

Too many variations across different actors

“Many actors, each doing it differently”

When teams get stuck in analysis paralysis trying to represent variations among different actors (performing the same process):

- Pick one of the actors, even if it’s an arbitrary choice, and first model his/her process
- Once that is completed, go to the other actors’ processes and determine if they can be “merged” into “main” process model. If not possible, it might be better to model them separately (e.g. “Handling refunds in USA”, “Handling refunds in South America”, etc.)

Final thoughts: CONCLUSIONS…

While previous presentation focused on **“What a GOOD process model is”**, this paper proposed **“A GOOD process to model processes”**.

The most important thing is not that you adopt this approach, but that you adapt, experiment, and improve a process that allows you to model processes in a way that is:

- **Effective** (create good process models)
- **Efficient** (use minimal/optimal resources to build good process models)
- **Flexible** (adapt the process, depending on why you do it, who cares about it, what do they care about, how much time you have available, etc.)

Thanks & Q&A

Hoping you found this paper and proposed approach useful.

Comments, suggestions, and/or questions?

Please contact me at razvan.radulian@why-what-how.com

Thanks:-)