15. Process Modeling for Information System Design

20 October 2008

Bob Glushko

Plan for ISSD Lecture #15

Levels of Abstraction in Process Modeling

ebXML Process Metamodel: Processes, Collaborations, and Transactions

Constructs in process models and BPMN

What Are You Doing Now?

If someone asks you "what are you doing now?" what might you say?

- "I'm living in Berkeley and taking courses at the University"
- "I'm studying Information Systems and Service Design"
- "I'm listening to a lecture on business processes and was just asked what I was doing now"

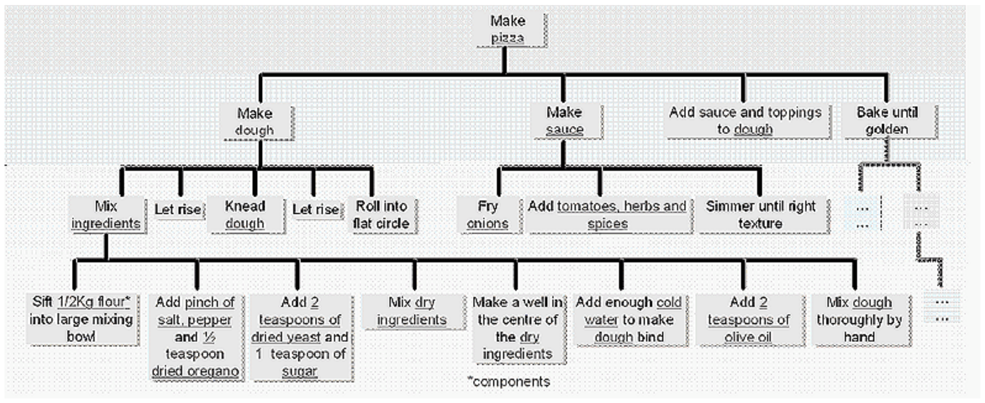
You can answer the question at many different levels of abstraction

What determines how you answer it?

Making a Pizza

1. Select recipe
2. Turn on as many lights as necessary to ensure adequate lighting in preparation and cooking area
3. Assemble required ingredients after washing and drying hands.
4. Sift 1/2 kg of flour into a large mixing bowl (preferably plastic), and then add a pinch of salt and pepper
5. Follow the rest of the preparation instructions in the recipe to make the pizza.
6. Pre-heat the oven to 350 degrees Fahrenheit.
7. Open the oven door, place the pizza in the oven, and close the door.
8. When the pizza is done remove it from the oven

Levels of Abstraction in Pizza Process Models


Make pizza
Make dough
Make sauce
Add sauce and toppings to dough
Bake until golden
Mix ingredients
Let rise
Knead dough
Let rise
Roll into flat circle
Fry onions
Add tomatoes, herbs and spices
Simmer until right texture
...
Sift 1/2Kg flour* into large mixing bowl
Add pinch of salt, pepper and ½ teaspoon dried oregano
Add 2 teaspoons of dried yeast and 1 teaspoon of sugar
Mix dry ingredients
Make a well in the centre of the dry ingredients
Add enough cold water to make dough bind
Add 2 teaspoons of olive oil
Mix dough thoroughly by hand
*components

The Abstraction Hierarchy in Process Models [1]

Unlike when we model documents, which by their very nature embody a relatively concrete perspective in an organizational model, we can model processes at many different levels of abstraction

We can model processes at an organizational level as the pattern of "value exchanges" between an enterprise and its suppliers, customers, and other entities in its "value chain"

These "value exchanges" connect activities to achieve some goal or solve some problem

- These models usually ignore exception cases and steps that don't have much business significance (they are "happy path" or "success" models)
- They are also called "Level 1" or "qualitative" process models

The Abstraction Hierarchy in Process Models [2]

When all of the steps and cases are included at a detail needed to explain an "as is" situation or simulate most aspects of a "to be" one, they are sometimes called a "Level 2" model

This level of description is what most people call a process model; transactions are visible, and documents and the processes that produce and consume them are at the same abstraction level

This level of description is technology-independent

The Abstraction Hierarchy in Process Models [3]

A "Level 3" model contains all the details needed to implement the process

Process models have little abstraction when we want to specify every detail of control and decision logic needed in an implementation

A model at this level of abstraction would specify both the success and failure cases and define any information components that are process inputs or outputs

The Abstraction Gap in Process Models

An abstract / coarse / strategic / goal-oriented view of processes (Level 1) is the top-down perspective of executive management

The granular / transactional / implementation view is bottom-up, less likely to be technology-independent, and naturally emerges from operational personnel as they describe the processes and documents they work with

When processes are implemented, they must be specified at a level that is compatible with the identities and capabilities of the actors that carry them out

But when processes have "executable specification" it may be difficult to relate them to the strategies and goals at the abstract level

Bridging the Abstraction Gap in Process Models

We need to bridge this gap to get a complete, balanced, and actionable view of the processes

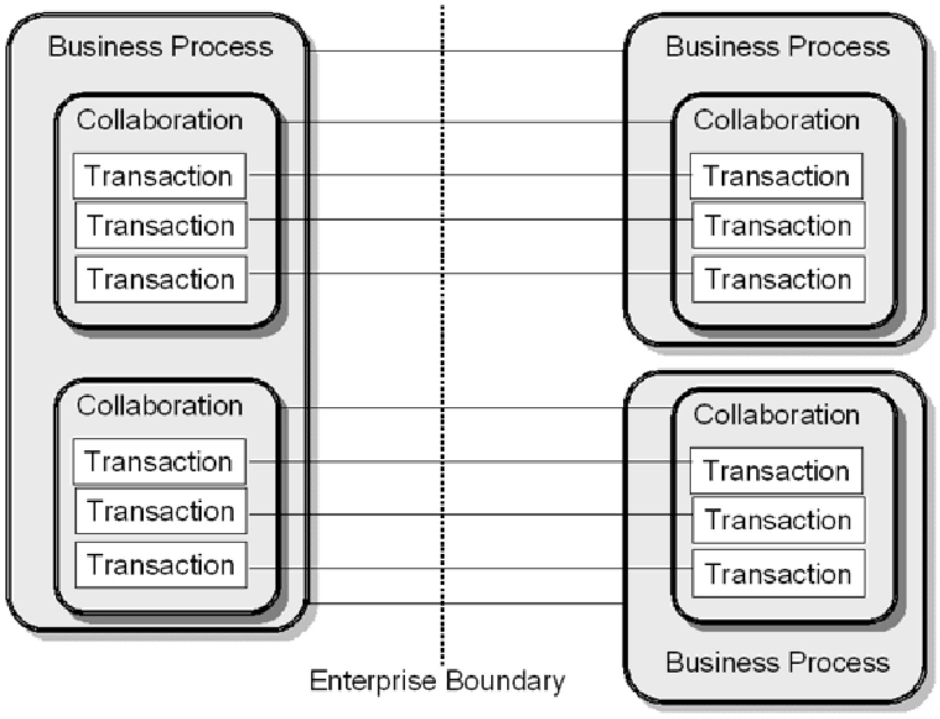
So there seems to be consensus that we need an intermediate level of process description that is more abstract than "transaction" and less abstract than "value chain"

Unfortunately, there isn't much consensus about what to call the different levels of abstraction

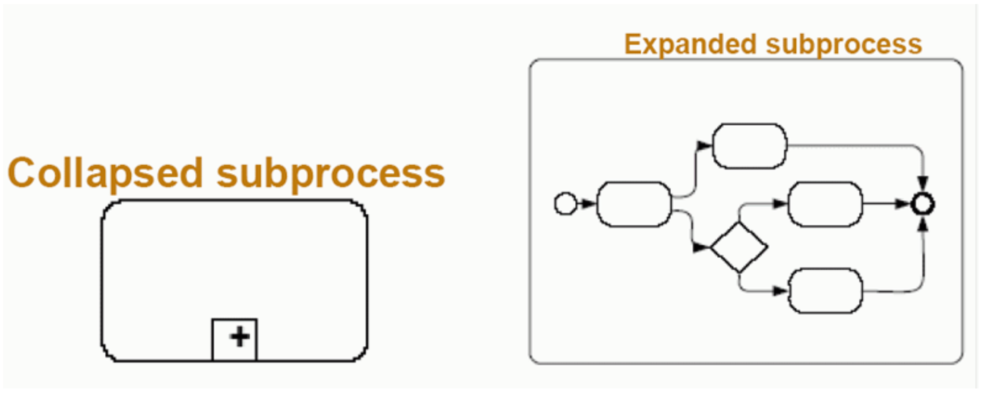
- in ebXML, these three levels are: process/collaboration/transaction; in BPMN, they are process/subprocess/task

Some but not all process metamodels and notations explicitly distinguish between human and computational actors

The ebXML Process Metamodel



Hiding Subprocesses in a "Collapsed" Model



A process is "atomic" when is has no interesting subpart structure (i.e., when it is a task or a transaction)

But it is often very useful to hide the subpart structure of a process and treat it as if it were atomic... because that's what a higher level of abstraction does

Business "Transaction"

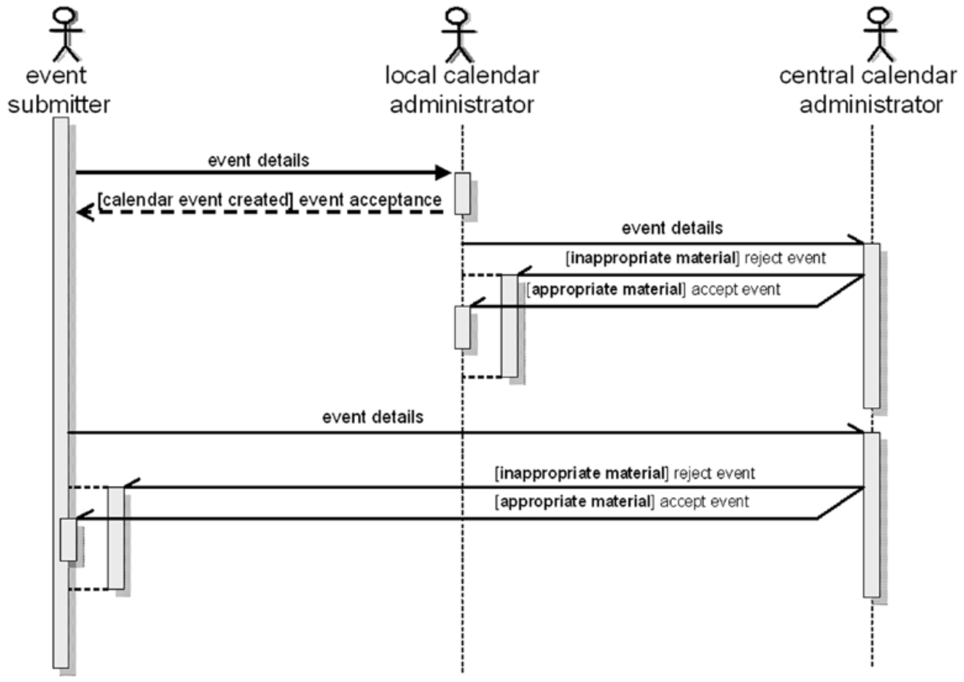
The most important perspective in process modeling for information system design is to describe processes at the level of the TRANSACTION

A business transaction is an atomic unit of work in a trading or commercial relationship between two parties or services

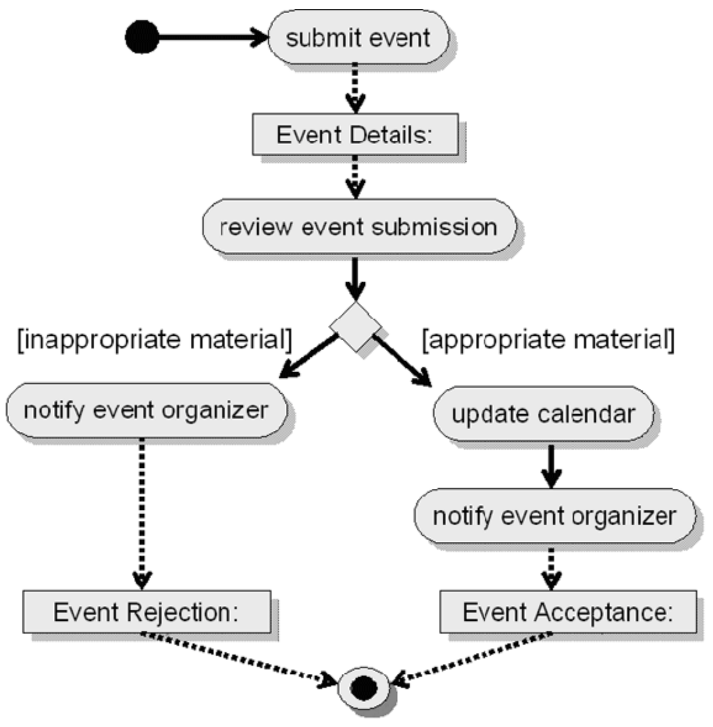
A business transaction is conducted between two parties or services playing REQUESTING and RESPONDING roles

There is always a requesting DOCUMENT (or MESSAGE) and optionally a responding one, depending on whether the transaction semantics are one-way or two-way

Sequence Diagram for "Submit Event" Transactions


event submitter
local calendar administrator
central calendar administrator
event details
[calendar event created] event acceptance
event details
[inappropriate material] reject event
[appropriate material] accept event
event details
[inappropriate material] reject event
[appropriate material] accept event

Activity Diagram for "Submit Event"



Business "Transaction" (2)

At the "business" level, a transaction either succeeds or fails.

- If it succeeds it may imply a legally binding contract or obligation between the parties participating in the transaction or otherwise govern their collaborating activities
- If the business transaction fails each partner must relinquish any mutual claim that would have been established by a successful transaction

Business transactions cannot be "rolled back" but the obligations established by a successful transaction can sometimes be undone by a COMPENSATING TRANSACTION

The time scale for a business transaction can range from seconds to days, weeks, or longer

(One Parenthetical Slide : Database "Transaction")

A DATABASE TRANSACTION is a group of statements or instructions to a database whose changes can be made permanent or undone only as a unit

Transactions provide a simple model of success or failure – a transaction either COMMITS (all its actions happen) or it ABORTS (all its pending actions are undone). This all-or-nothing quality makes for a simple programming model

A transaction can be ROLLED BACK in the same unit with which it was committed to undo all of its effects and return the database to a prior state

The time scale for a database transaction is measured in fractions of a second

Modeling Transactions with Worksheets

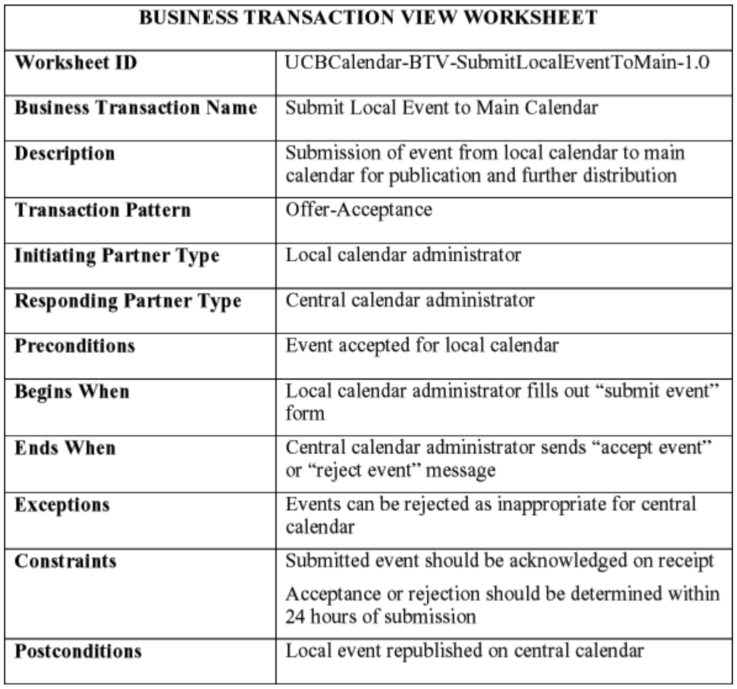
A model of a transaction can be recorded as an artifact as a diagram, an XML instance, or as a "worksheet" that simply lists the information components in the metamodel of the transaction

You can tailor the modeling approach by customizing the worksheet to fit your or your client's methodology

Using worksheets to capture information lets the analyst capture "fragments" of the metamodel whenever they arise or are discovered

This is a less prescriptive and more user-friendly approach than the other notations, which require more information to be "well-formed"

Transaction Modeling Worksheet for "Submit Event"



BUSINESS TRANSACTION VIEW WORKSHEET	
Worksheet ID	UCBCalendar-BTV-SubmitLocalEventToMain-1.0
Business Transaction Name	Submit Local Event to Main Calendar
Description	Submission of event from local calendar to main calendar for publication and further distribution
Transaction Pattern	Offer-Acceptance
Initiating Partner Type	Local calendar administrator
Responding Partner Type	Central calendar administrator
Preconditions	Event accepted for local calendar
Begins When	Local calendar administrator fills out "submit event" form
Ends When	Central calendar administrator sends "accept event" or "reject event" message
Exceptions	Events can be rejected as inappropriate for central calendar
Constraints	Submitted event should be acknowledged on receipt Acceptance or rejection should be determined within 24 hours of submission
Postconditions	Local event republished on central calendar

Business Process Concept of "Collaboration"

Two or more business transactions can be sequenced to carry out a *business collaboration*

A business collaboration is a series of activities carried out by two or more partners for the purpose of achieving a business goal; these activities are business transactions

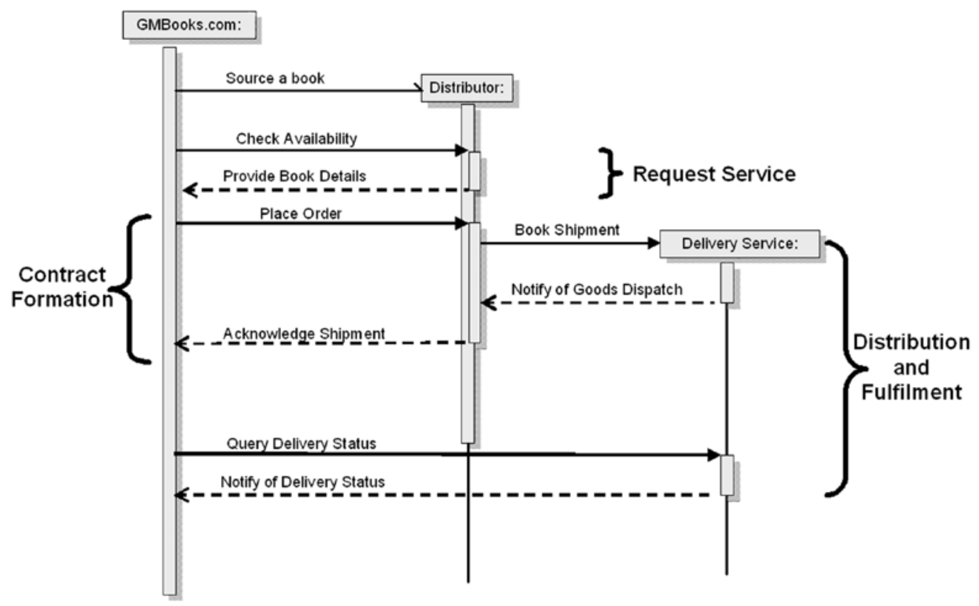
These transactions have more context in common with each other than with transactions that perform other parts of the business process

Meaningful and Necessary Semantic Overlap

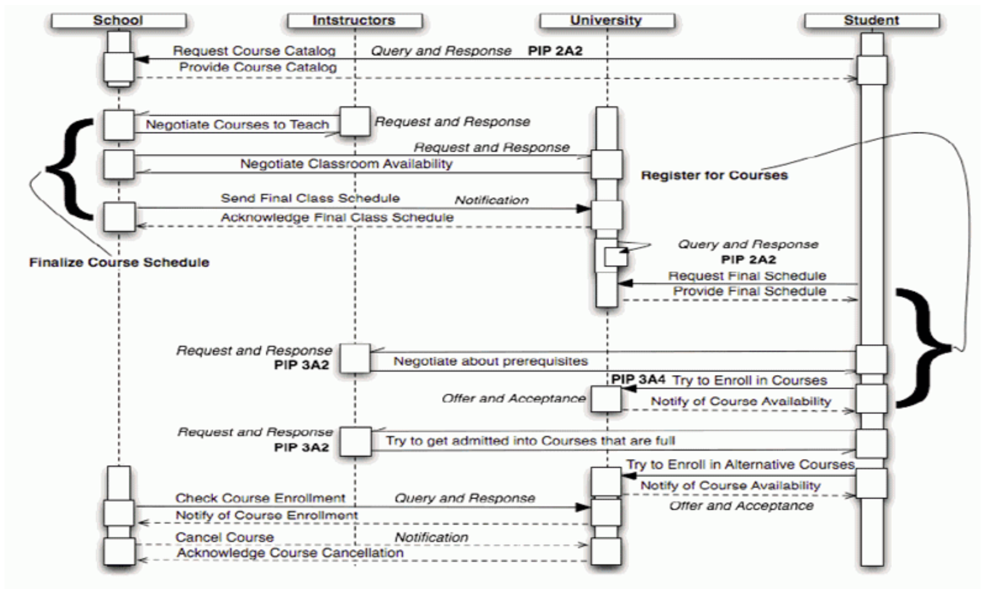
The set of transactions in a collaboration have meaningful and necessary semantic overlap with each other

- They have parties/actors in common
- The overlap must be necessary -- two parties must need to know about each of their transactions with a third party for the set of transactions to be collaborative

Example: Collaborations in Drop Shipment



Example: Registering for Classes



Business Process Modeling Notations

Process models, especially Level 1 and Level 2 ones, are conceptual: their meaning isn't tied to how they are represented or described or depicted

Notations for representing process models can be informal -- box and arrow diagrams on napkins as an end case -- or formal

More formal notations have the advantage that their explicit and consistent semantics can be supported by tools

And formal notations, once learned, can facilitate the shared understanding that is usually the point of process modeling

BPMN seems to be an emerging standard notation... but not the only one you probably need to have some familiarity with

BPMN Notation - 3 "First-class" Elements

Activity -- work performed in the process (rounded rectangle)

Gateway -flow control logic (diamond, with icons)

Event - a "signal" that something has happened; can be internal or external to a process (circle)

Activity Semantics

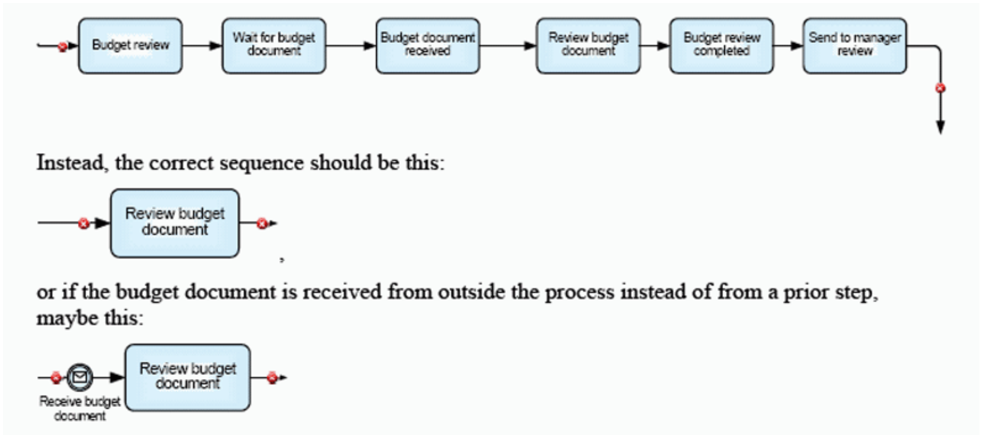
Activities are actions, not functions, states, or handoffs

Sequential flow is implicit in the (arrow) notation

Likewise, you don't need "waiting for" activities; an activity starts when its incoming flow reaches it

You can reinforce these ideas by labeling activities VERB-NOUN

Activity Semantics - Examples



Instead, the correct sequence should be this:

or if the budget document is received from outside the process instead of from a prior step, maybe this:

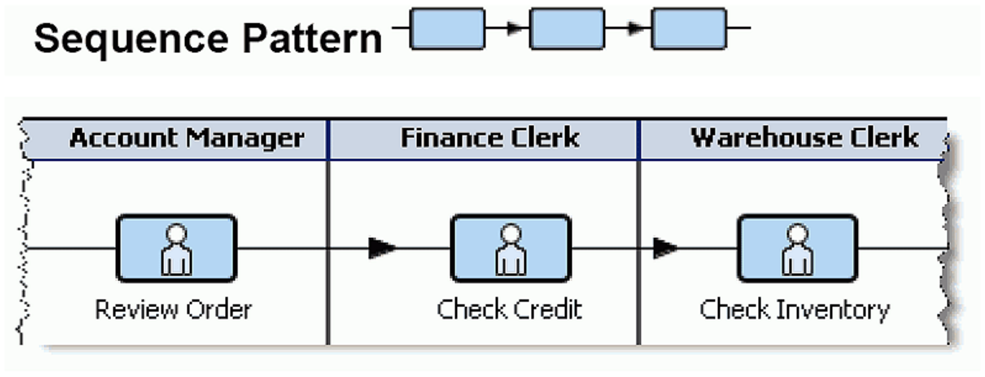
Control and Logic Patterns

Sequence

Choice

Merge

The Sequence Pattern



Sequence -- activities following each other -- is the default pattern for describing processes. Processes often look sequential when you take a coarse or distant view of them

But is it Really a Sequence?

A sequence model like this assumes that processes are dependent and the transitions between them are unconditional

A sequence model assumes that all inputs are treated the same way, and there are usually instance-level (content-based) rules that be applied to "thin" the queue or re-route items from it

A GATEWAY can be used to split a sequence into multiple flows

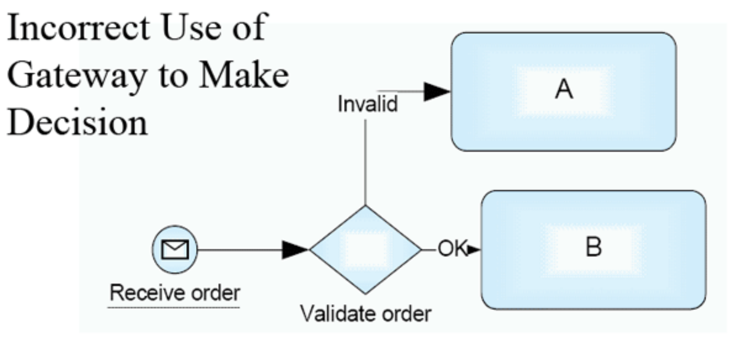
Gateway Semantics

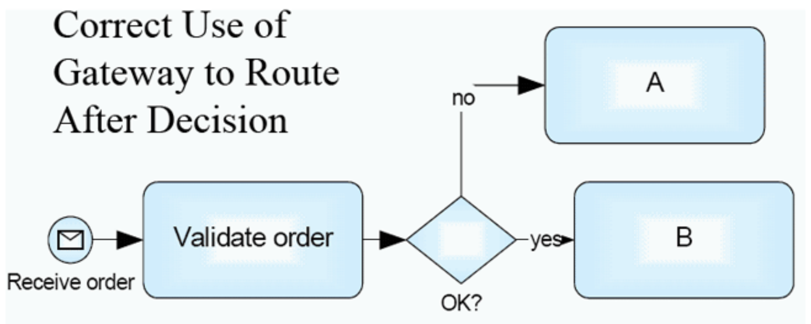
Unlike activities, gateways do NOT perform work or make decisions

They control the flow of a process AFTER a decision has been made

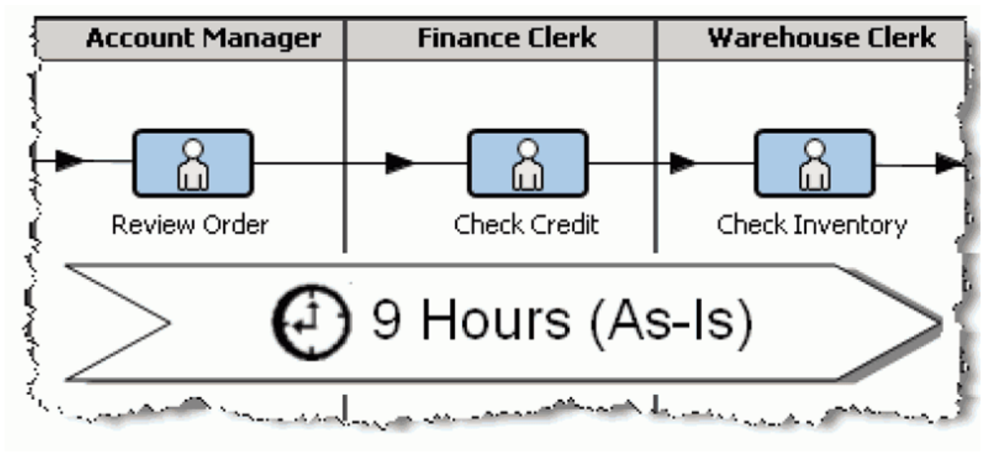
So don't use gateways to represent decision logic

Gateway Semantics - Examples

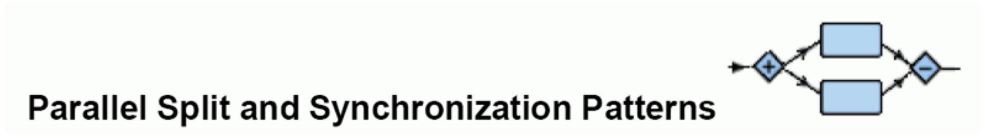


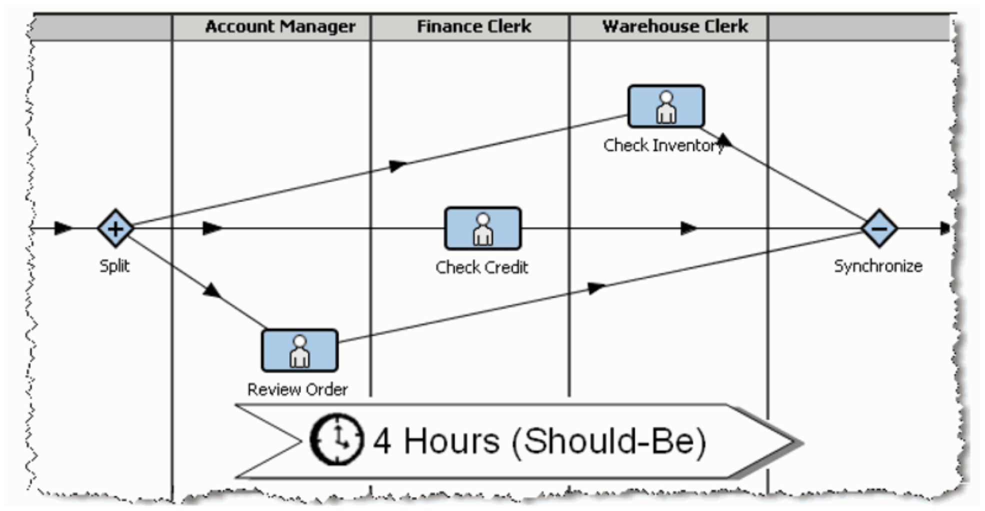


Inefficiency in Sequence Pattern

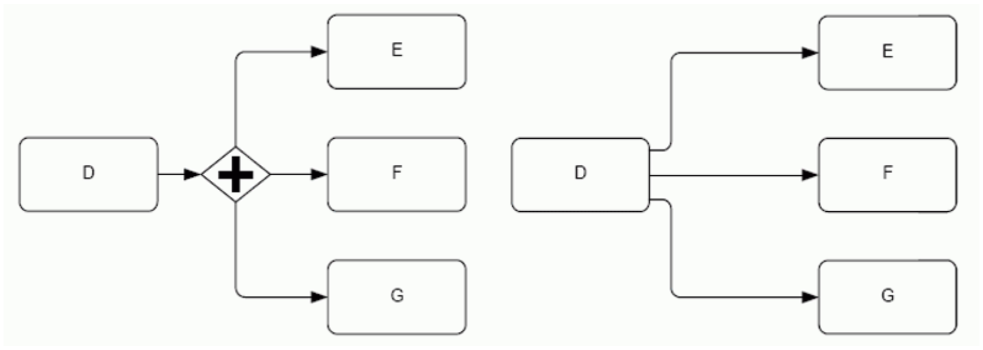


Removing Inefficiency with Parallelism





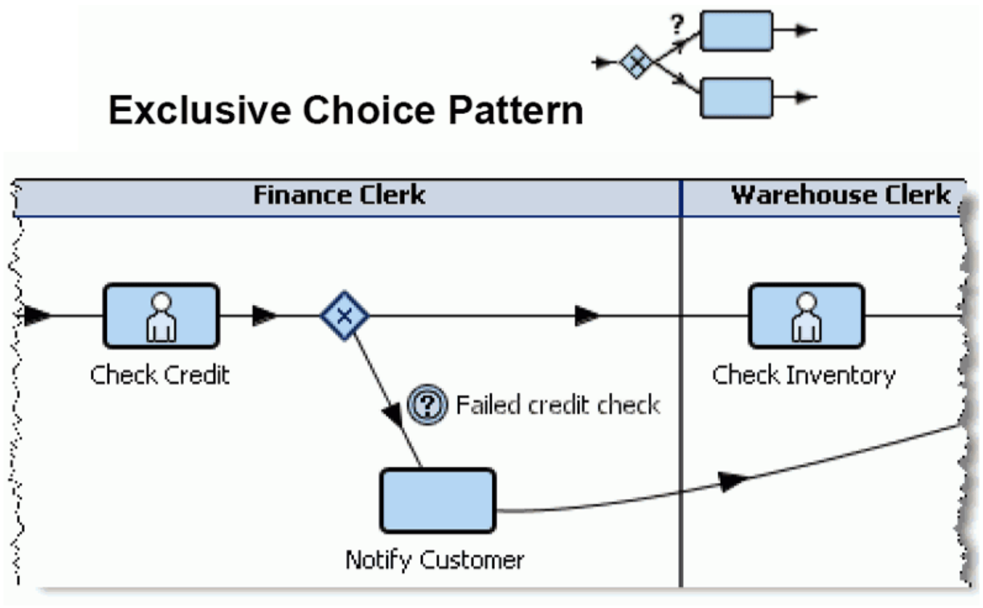
(Parallel Split | AND | Unconditional) Gateway



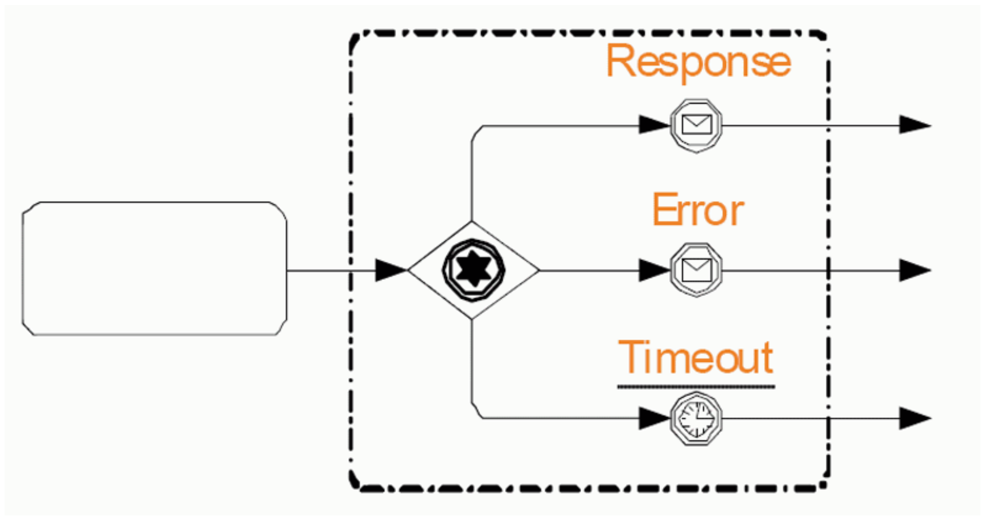
All of the outgoing paths are unconditionally enabled, which makes the gateway symbol redundant

But we'll have to merge the parallel paths at some point...

Conditional Activities



(Exclusive | First) Event-based Gateway



Emerging "Event" Buzzwords in Process Modeling

CEP (Complex Event Processing)

EDA (Event-Driven Architecture)

Event-Driven BPM (Business Process Management)

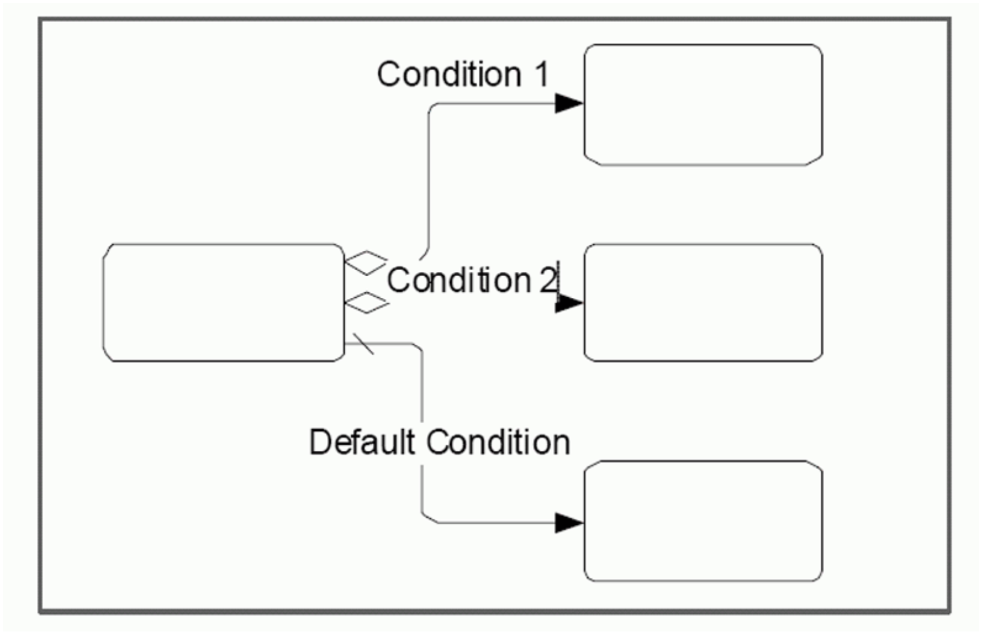
Event-Driven SOA (Service Oriented Architecture)

ESP (Event Stream Processing)

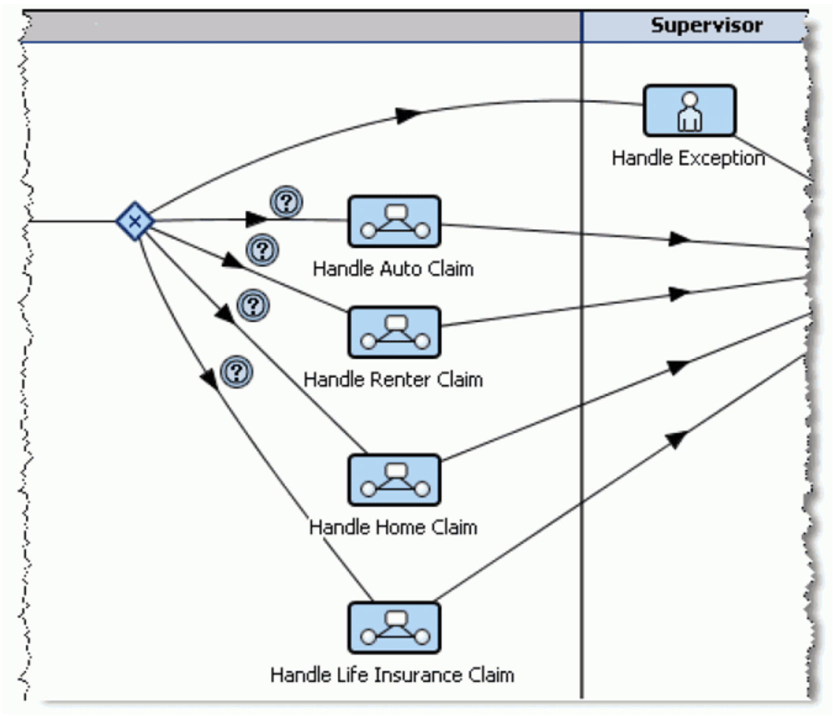
BAM (Business Activity Monitoring)

The "Event-Driven" Enterprise

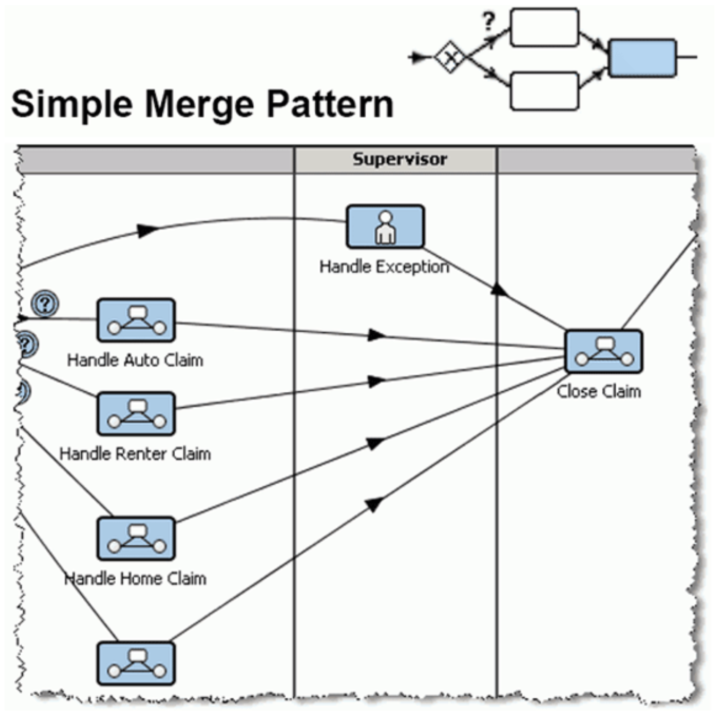
(Inclusive | OR) Conditional Parallelism



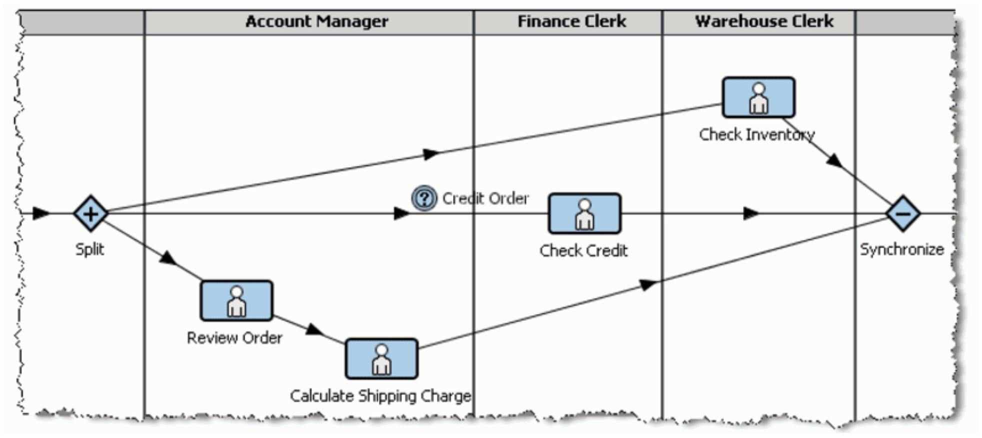
Allowing for an Unconditional Activity



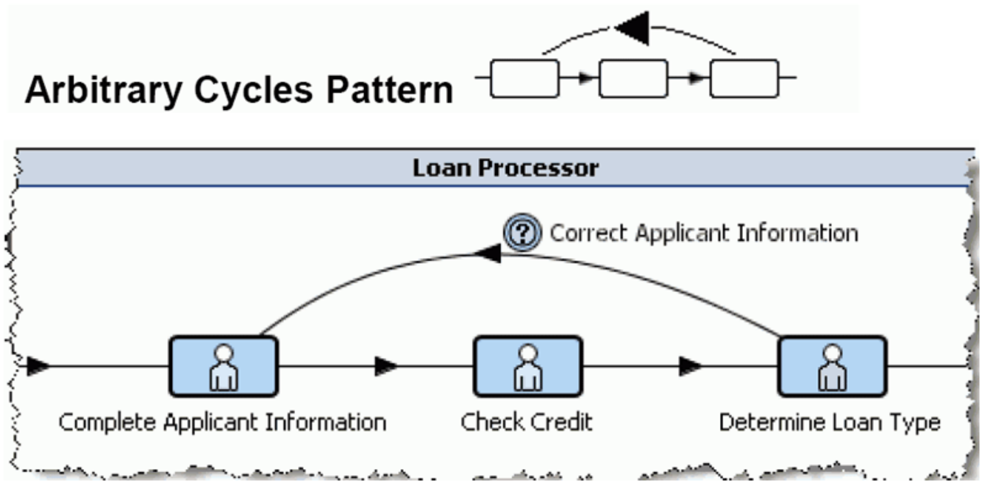
Merging Conditional Process Branches



Synchronized Merge (for Conditional Parallelism)



Poor Process Design Allows "Upstream" Cycles



Readings for 22 October

Robert J. Glushko & Tim McGrath, Document Engineering, Chapter 10 (through 10.6), "Designing business processes with patterns", 2006

Uday M. Apte & Richard O. Mason, “Global Disaggregation of Information-Intensive Services.” Management Science, July 1995.