To BPEL or not to BPEL?

BPEL disadvantage: more time to product...
BPEL advantage: better reusability, analysis...
A tale of two strategies...

The older strategy: salesforce.com:
Write business logic in a custom language.
Provide a large variety of services.
No BPEL.

The newer strategy: intalioWorks.com:
Write overall strategy in BPMN;
Refine into BPEL;
Plumb BPEL to services graphically without writing code.
Define your business processes via BPMN, and (using design patterns in WS-CDL), convert them to executable BPEL, which you execute on a BPEL engine, e.g., Apache ODE or IBM Websphere, which contacts web services (e.g., generated by apache AXIS or salesforce APEX), which utilize cloud services via java APIs.
So, why are we this high up?

BPMN is the Cobol of the new millenium!

    Easily readable by a manager, and
    Clearly expresses observable intent, but
    Darn near impossible for a manager to write!

BPEL is not understandable to managers, but need not be, because it only implements the BPMN observables.

It might be argued that BPEL caused clouds as we know them, by justifying their use.
The real effect of BPMN

Allows one to judge just how important reliability is, by answering "what-if" questions about service performance, and analyzing the cost of service delays, which justifies high-reliability services, which led to clouds!
The hidden use for BPMN models

Can use the model, without code, to predict business performance and value.
Can model the effects of performance and reliability.
Can predict bottom-line behaviors.
How to analyze BPMN

BPMN analysis is a simple generalization of PERT (Program Evaluation and Review Technique), which is a generalization of Critical Path Method analysis (CPM).
Critical Path Method

A basic tool for project management

Express task precedences as a **task graph**.
Task A -> B means A should precede B.

Has a **Start node**: when project starts.
Has an **End node**: when project is finished.
Relationship between CPM and BPMN:

BPMN processes take time.
Time delays are measurable and predictable.
Effects of delays are predictable.
Can use statistics on delays, as well as traces of business load, to predict overall performance, and even profit.
Can determine which processes are critical, which determines which cloud functions are critical.
Can ask "what-if" questions about what will happen if critical processes change. E.g., via a different SLA.
Two BPMN processes (request and response)
One CPM task (get data).
Trace-driven simulation

A common business decision support technique

**Input:**

statistics on subprocess time requirements, and an actual trace of customer requests for a given day.

**Output:**

predictions of how that day would have gone if processes had been more (or less) efficient.

**Key is cost/value analysis:**

- **Cost:** what you pay for service, or what it costs you to provide it yourself.
- **Value:** what customers do, and how much money you make.
The big deal:

BPEL can be tuned based upon observation.
It is not static, but its performance changes when you reassign parts of it to different services.
CPM Definitions

**Duration:** how long it takes to do a task.

**Earliest start time:** earliest time a task can be started
= max of earliest completion times for prerequisites.

**Latest start time:** latest time a task can be started
and still finish with minimum completion time for project.

**Earliest completion time:** earliest start time + task
duration.

**Latest completion time:** latest start time + task
duration.

**Slack time:** latest completion time - earliest
completion time = latest start time - earliest start
time.

A task is critical if its earliest start time is its latest start
time, i.e., slack time = 0!

Changing the time of a critical task changes overall
behavior.

Key idea: if a service is critical, improving it means
something.
The critical path theorem

Critical tasks do not occur in isolation, but instead lie on a critical path from start to finish.
There may be multiple critical paths.
If any task on a critical path changes in duration, the whole project duration can change.
The critical path algorithm

Objective: compute slack time for every task.
Two phases: forward scan and backward scan.
Forward scan: compute earliest completion time.
Backward scan: compute latest completion time.
A task graph has a start, end, and intermediate tasks.
Forward scan:

Label each task with its time.
Mark start's earliest start time and earliest completion time as 0.
From left to right (start to finish),

- If a task's predecessors are marked with their earliest completion times,
  - its earliest start time is the maximum of predecessor earliest completion times and
  - its earliest completion time is its earliest start time plus its duration.

At the end, every node is labeled with its earliest start and completion times.
Start at beginning

Compute **earliest start time** and **earliest completion time**.
Backward scan:
Mark end's latest start time and latest completion time as its earliest start time and completion time. From right to left (finish to start),
- If a task's successors are marked with latest start time,
  - its latest completion time is the minimum of successor latest start times and
  - its latest start time is its latest completion time minus its duration.
At the end, every node is labeled with its latest start and completion times.
Start at end
Compute latest completion and start times from end to beginning
A critical path has earliest start time = latest start time.
CPM and PERT

CPM: Critical Path Method
   input: task times and precedences
   output: minimum completion time, slack time, etc.

PERT: Program Evaluation and Review Technique
   input: task time probability distributions and precedences
   output: probability distribution of minimum completion time, slack time, etc.
How (simple) PERT works

Assign a normally-distributed probability distribution \( n(\mu, \sigma) \) to each task:

- \( \mu \) = mean completion time
- \( \sigma \) = standard deviation of completion time

\[
n(\mu, \sigma)(t) = \exp\left(-\frac{(t-\mu)^2}{\sigma^2}\right)
\]

Compute the PDF of the earliest start time and latest start time from the PDF of the task completion times!

If we have two tasks in sequence, and they're independent, then the probability that the two take time \( t \) is the integral of the product of PDF\((t_1)\)*PDF\((t_2)\), for all \( t=t_1+t_2 \) (convolution).
Suppose we have an empirical distribution as follows:

<table>
<thead>
<tr>
<th>Interval</th>
<th>Instances</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-10</td>
<td>50</td>
<td>50/100</td>
</tr>
<tr>
<td>10-20</td>
<td>40</td>
<td>40/100</td>
</tr>
<tr>
<td>20-30</td>
<td>10</td>
<td>10/100</td>
</tr>
</tbody>
</table>

Caveat:
- Distributions change over time.
- Sensitive to time of day and external load.
- Sensitive to external events.

Call this \( d \)

If \( 0 \leq d < 10 \), \( d \) is not critical, delay is 35 with \( p = 1/2 \)
If \( 10 \leq d < 20 \), \( d \) is not critical, delay is 35, \( p = 4/10 \)
If \( 20 \leq d < 30 \), \( d \) is critical, delay is 35-45, \( p = 1/10 \)

So,

We can compute the expected value, which is the sum of

\[
\text{value} \times \text{Prob(value)}, \text{where the value is an interval}
\]

\[
= (35, 35) \times 9/10 + (35,45) \times 1/10
\]
Suppose that d is

<table>
<thead>
<tr>
<th>10</th>
<th>p = 0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>p = 0.5</td>
</tr>
<tr>
<td>30</td>
<td>p = 0.3</td>
</tr>
</tbody>
</table>

and e is

<table>
<thead>
<tr>
<th>10</th>
<th>p = 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>p = 0.3</td>
</tr>
<tr>
<td>30</td>
<td>p = 0.6</td>
</tr>
</tbody>
</table>

10+10 at probability 0.2*0.1
10+20 at probability 0.2*0.3
10+30 at probability 0.2*0.6

....
Practical modeling

Completion times expressed in terms of frequency tables for observations:

Completion time range on X
Frequency on Y
Empirical probability: frequency/total trials.

\[ \begin{array}{cccccc}
& 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
6 & & & & & & & \\
5 & & & & & & & \\
4 & & & & & & & \\
3 & & & & & & & \\
2 & & & & & & & \\
1 & & & & & & & \\
5 & & & & & & & \\
\end{array} \]

\[ \begin{array}{cccccc}
& 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
5/9 & & & & & & & \\
2/9 & & & & & & & \\
2/9 & & & & & & & \\
1/9 & & & & & & & \\
5/9 & & & & & & & \\
\end{array} \]

\[ \text{Practical modeling} \]

\[ \text{Practical modeling} \]

\[ \text{Empirical probability: frequency/total trials.} \]
Limitations of modeling

Response times for external services vary with (unpredictable) external load (and time of day).

**Cannot model unforeseen situations.**

Amusement: IBM has tried to beat this, by doing modeling that injects unforeseen situations!

E.g., gnomes running around causing hardware failures!
Alas,...

I haven't been able to locate a freeware BPMN analyzer, that:
Assigns probabilities to failures,
Simulates resulting operational behavior, and predicts effects upon profits.
But these do exist; they just cost money!