Is orchestration an ESB responsibility?

Asked 16 years ago Modified 7 years, 5 months ago Viewed 9k times

17

Is an Enterprise Service Bus (a tool that acts as a mediator, a message broker, a service enabler, schema transformation enhancer, transparent location provider, service aggregator, load balancer, monitor, and all that stuff) **responsible to orchestrate services**?

What about putting an automated business business process with more than thousand steps and dozens of service invocations inside your enterprise service bus?

Would you do it, or would you use a specialist in orchestration such as a BPEL engine?

Please gimme you opinion.

soa esb bpel orchestration alsb

Share

Improve this question

Follow

edited Jul 25, 2017 at 13:05

Arialdo Martini
4,567 ● 3 ● 34 ● 42

asked Dec 6, 2008 at 2:00

paulosuzart
1,092 ● 2 ● 13 ● 28

7 Answers

Sorted by: Highest score (default)

16

Yes and no. There's a thin, and sometimes indistinguishable line between orchestration and aggregation/service augmentation.

In general, if you've got any long-running or complex business process (process being the key word, although I'm going to avoid defining it) - that's best suited to BPEL.

Simple tasks, such as aggregating the results of three service calls, could and often should be done in an ESB layer.

It's not worth losing too much sleep over, though

Disclaimer: I am an IBM ESB consultant, although I'm not writing this in an official capacity.

Share Improve this answer Follow

answered Dec 7, 2008 at 9:25

Andrew Ferrier
17.7k ● 14 ● 53 ● 78

Another alternative to BPEL is BPMN -- although BPEL seems to be better suited to orchestration type activities and BPMN more to process like activities. – Marco Sep 26, 2011 at 18:12

10

No, an ESB's responsibility is not the orchestration of services (per se). The ESB provides a layer of abstraction at the "software infrastructure level".

This means that an ESB is a "single logical abstract port of call for connectivity" with any service that is published on the bus.

The ESB being abstract, means that consumers of services on the bus, don't "need to know" deployment details of the service, and it is possible to expose "internally facing services" with a single document model. The ESB provides low level services (such as protocol translation and message transformation), so that internally services can communicate in a simplified fashion.

This implies some orchestration: The ESB provides orchestration of the afore mentioned low level services (e.g. when service X is called via IIOP, translate this to SOAP with Attachments. Then transform the request from whatever serialized data to an XML payload).

The orchestration you would typically avoid in an ESB is: In order to process this (insurance) sale, we first need to validate the information provided by the buyer, then we need to underwrite the risk of insuring, and finally calculate the premium that needs to be paid for the insurance, after which we need to… etc.

The steps described above are clearly a business process (which could even be interrupted… e.g. if automatic underwriting is not possible, then a human underwriter needs to further assess the risk).

Business Services (e.g. Validation, Underwriting, Premium Calculation) that make up a Business Process (e.g. Insurance Sale), which is what is typically referred to as Orchestration, is best suited to happen in a Business Process Engine and defined using a formalized Business Process Modeling Language (such as BPEL).

Also making a guess about the many steps in your process: In the above example, Validation is a (course grained) service. The validation rules themselves are internal to that service. For complex business rules (i.e. not business process), the use of a Business Rules Engine may be required.

Share Improve this answer Follow

answered Dec 3, 2012 at 3:45

Brian

101 ● 1 ● 2

5

My short quick answer is NO, that not its responsability.

I would rather let that to the BPEL or a BPM suite.

Mhh I don't know what else to add :) ... Good luck?

Share Improve this answer Follow

answered Dec 6, 2008 at 2:05

OscarRyz

199k ●119 ●396 ●573

5

Now my own vision.

Regarding all the work an ESB has to do, putting service orchestration inside the main infrastructure element of your SOA is not a good idea.

Aggregate, ok! But keeping your communication channel busy with business logic will, for sure, cause a terrible impact in the ability to delivery other features.

After all, **most ESBs** such as as BEA Aqualogic Service have a **limited support for orchestration** including **lack of stateful** capabilities, and activities like wait (a timer) or pick (wait for some input to move on the process), split/join capabilities (already added on ALSB 3.0), and so on.

No way. Just use tools like a BPEL engine or a tool like Weblogic Integration.

Thanks.

Share Improve this answer

Follow

edited Dec 6, 2008 at 2:14

OscarRyz
199k ●119 ●396 ●573

answered Dec 6, 2008 at 2:09

paulosuzart
1,092 ●2 ●13 ●28

Or ALBPM (oops I meant Oracle BPM) – OscarRyz Dec 6, 2008 at 2:14

Jjajajaja! Ok man, i forgive you. But for me Oracle BPM or Oracle Service BUS are still the old and nice BEA.
– paulosuzart Dec 6, 2008 at 2:18

2

Whenever you have two or more services that interact use service orchestrator, i.e. for composition and process control services. If you have esb expose this composition service on esb. Now if you have to compose new service that includes this composition service use orchestrator and again expose on esb. Use esb as service delivery mechanism and web service broker and proxy. In composing a service orchestrator will use esb to reach interacting services. If these interacting services use incompatible xml schemas esb can transform/map them to common schema in runtime and route service requests based on the content, e.g. namespace.

Share Improve this answer Follow

answered Sep 30, 2011 at 16:25

Bran
21 ● 1

1

Yes orchestration is a responsibility, in most cases, of the ESB. Or, alternatively, if you draw a line between ESB infra and orchestration infra, then you are doing so on a physical level for performance reasons, not for logical attribution of responsibility.

You have 2 choices - when, for example, an HR system receives a new employee - where do you place the business logic that says "the compliance department will need to approve and check first, and then if that's ok, the HR department will need to finalise the hire, then the accounting department will need a new entry, and then the payroll system will need updating, and if that fails, then we'll need to send an email to HR"? If all business processes are considered 'owned' by the initiating dept/application, then the overall system that is the enterprise becomes complex, with disparate orchestration systems.

The second choice is centralise the orchestration, essentially making it a logical partner of the messaging platform. If you choose to see these as separate artifacts, that is up to you, but it is equally valid to described both as ESB.

Share Improve this answer Follow

answered Oct 25, 2011 at 6:46

Sentinel
3,677 ● 2 ● 34 ● 45

disagree, and others above seem to also disagree. even your example screams dont do orchestration in ESB. no the initiaiting platform should orchestrate the status of this new data, and the ESB will be called multiple times at different points in this process (the key word is process not service) other platforms have no use for this process instead they enjoy the fruits of the initiating platforms labour, and the result is then available via the ESB. so the answer is, "NO" – TheArchitecta May 22 at 8:16

If the initiating platform is not capable to process the data and do the orchestration, then that is an entirely different subject. – TheArchitecta May 22 at 8:18

0

An Enterprise Service Bus should never be responsible for orchestrating services.

Orchestration implies a minimum of "smarts", specifically the ability to compensate for failed transactions. Service bus tools will often say they offer "try-catch" or something like that but the ability to run scoped componsation is the mark of a proper orchestration tool. Additionally the ability to wait, know its own state, or keep things in suspense is another indicator that you're dealing with an orchestrator and not a bus.

Speaking to 1000+ steps plus dozens of services, consider the if-then's in the process. If all the if-then statements in your 1000 steps speak only to routing with no change to the payloads then you're still in "routing" and therefore still in ESB. But if there's even one *nested* if-then and I start to look for different tools. Aside, if-thens that look like routing can very quickly impact business logic. Once business logic starts showing up then a better language such as BPEL or BPMN is better.

The example of an orchestra conductor is often given to describe how orchestration works, a central individual directing the musicians according to a score. Often what's left off is the idea that the conductor is not only directing, but listening as well, and if something goes wrong can compensate in a reliable, repeatable way.

For instance imagine our first conductor goes to bring in the tuba player but said tuba player has decided to go do something else. A simple pinball-style "orchestrator" will bring in the tuba section, knowing full well it isn't there, and then wait for the audience to complain later. A really savvy conductor would see the tuba gone, and immediately bring up the deeper baritone horns to compensate.