How does ServiceMix compare to Mule

Conceptually, ServiceMix and Mule are quite similar in that they allow services to be integrated through different APIs and across different transport technologies. Service Mix and Mule provide similar features, and both solutions enable a simple, lightweight POJO-based deployment model that uses the Spring framework to integrate services.

Both ServiceMix and Mule are designed as integration solutions whose capabilities are independent of the transport. ServiceMix achieves this through its JBI-based architecture, while Mule is based on a services container and configuration of message endpoints.

Compared to Mule, the major difference for ServiceMix is its architectural design, which is fundamentally based on the Java Business Integration (JBI) standard. Mule provides a JBI binding so that Mule components can interact with JBI containers, including the ServiceMix JBI container. However, the internal Mule APIs are not based on the JBI standard.

While JBI support is a very good thing, ServiceMix provides not only external APIs to support JBI-based integration, but also internal APIs based on JBI, to give agility and flexibility not only with respect to the integration capabilities at the endpoints, but also to the functionality of applications executing across the fabric of the integration environment.

For example, because JBI supports a standard deployment unit format, you can hot deploy any JBI-compliant BPEL engine (or set of BPEL files into a BPEL engine), rule engine, transformation engine, scripting engine or other integration component (such as specific JBI binding components) directly into ServiceMix. Also ServiceMix contains a number of WS-* implementations built on JBI, such as WS Notification.

Even though the JBI standard and the Mule messaging platform look very different at the API level, conceptually they are trying to do similar things. Like Mule, JBI is based on the idea of both component and container based routing. However, JBI provides a standard for the integration of any commercial or open source component. Equally important, the JBI standard supports a very high level of integration functionality. A JBI component can perform smart routing to endpoints. A JBI component can just communicate with its default service, letting the container do the routing for it. Or a JBI component can give the JBI container a hint of which endpoint to use (such as providing a service QName to use or specifying an operation QName) and then the JBI container can, if there are many available services to choose from, use some kind of policy negotiation or load balancing algorithm to choose which physical endpoint to choose. Lastly, JBI is a cleaner abstraction for working with both logical and physical web service endpoints, WSDL and the standard WSDL defined message exchange patterns (MEPs).

While ServiceMix supports a lightweight deployment model similar to that of Mule, ServiceMix is also fully integrated into Apache Geronimo, bringing JBI functionality into Apache's J2EE container. This allows Apache Geronimo to hot-deploy any JBI component or service using its standard deployment tooling (JSR 77/88 compliant). This difference between the two products also originates in our focus on agility. Some ServiceMix implementations will be endpoint-centric, like Mule. Others may leverage the capabilities of an application server. ServiceMix is designed for either integration scenario, or some combination of the two.

Today's Open Source ESB solutions are increasingly focused on interoperability, and despite their differences in architecture and design philosophy, this principle of interoperability applies to ServiceMix and Mule. The Mule JBI binding enables Mule components to interact with the ServiceMix JBI container. Also, Mule transports, components and transformers can be used inside the ServiceMix JBI container. Likewise, ServiceMix has full support for Mule; so if you already have an investment in some Mule configuration or code, you can reuse it inside ServiceMix along with any other JBI components.