**Understanding Orchestration Using Mule**

Orchestration is how Mule facilitates the process of moving data through a Mule implementation based on flow. Orchestration can be seen as combining service calls to create higher-level, more useful composite services, and implies implementing business-level processes combining business-specific services across applications and information systems.

**Service Orchestration and SOA**

Application or service orchestration is the process of integrating two or more applications and/or services together to automate a process, or synchronize data in real-time. Often, point-to-point integration may be used as the path of least resistance. However, point-to-point integration always leads to a complex tangle of application dependencies that is very hard to manage, monitor and maintain. Application orchestration provides a) an approach to integration that decouples applications from each other, b) capabilities for message routing, security, transformation and reliability, and c) most importantly, a way to manage and [monitor your integrations centrally](https://www.mulesoft.com/platform/anypoint-management-center).

SOA, or[Service Oriented Architecture](https://www.mulesoft.com/resources/esb/bottom-up-soa), is an approach to developing enterprise systems by loosely coupling interoperable services - small units of software that perform discrete tasks when called upon - from separate systems across different business domains. SOA emerged in the early 2000s, offering IT departments a way to develop new business services by reusing components from existing programs within the enterprise rather than writing functionally redundant code from scratch and developing new infrastructures to support them. With SOA, functionalities are expressed as a collection of services rather than a single application, marking a fundamental shift in how developers approach enterprise architecture design. A crucial aspect of SOA is service orchestration. Finding a platform with enhanced service orchestration capabilities, then, is a high priority for enterprises looking to build their systems according to SOA.

## **Service Orchestration: Making SOA Work**

Similar to an organizational workflow, service orchestration is the coordination and arrangement of multiple services exposed as a single aggregate service. Developers utilize service orchestration to support the automation of business processes by loosely coupling services across different applications and enterprises and creating “second-generation,” composite applications. In other words, service orchestration is the combination of service interactions to create higher-level business services. Service orchestration works through the exchange of messages in the domain layer of enterprise applications. Since individual services are not programmed to communicate with other services, messages must be exchanged according to a predetermined business logic and execution order so that the composite service or application can run as it is demanded by the end-user. This is usually accomplished through [enterprise application integration](https://www.mulesoft.com/resources/esb/enterprise-application-integration-eai-and-esb) (EAI), which enables data integration, and the use of a central messaging engine such as an [enterprise service bus](https://www.mulesoft.com/resources/esb/eliminating-point-point-integration-pain-mule-esb-use-cases) (ESB), which routes, transforms and enriches messages.

Service choreography is also related to service orchestration, as both are employed to create composite services and applications in service oriented architectures; however, it is still worth pointing out the differences. A service choreography model works without a central messaging engine or orchestrator while a service orchestration model relies on a central controller to couple services. In the former, the participating services each know the business logic and sequence and timing of message exchanges. In the latter, the participating services don’t know that they are being orchestrated as part of a higher-level service; only the central controller knows the business logic and messaging sequence.

To get a better sense of service orchestration, let’s take a look at the following example. A loan broker wants to make a loan request on behalf of a customer and uses an automated Loan Request Service. The broker accesses the Loan Request Service in the enterprise system to make the initial loan request, which is sent to an orchestrator that then calls and invokes other services in the enterprise, partner systems and/or the cloud to process that request. The individual sub-services involved in the loan request include a service to obtain credit scores from a credit agency, a service to retrieve a list of lenders, a service to request quotes from a bank service, and a service to process quotes with the data from the other services. Together, the orchestrated services comprise the Loan Request Service, which then returns a list of quotes from potential lenders to the broker who made the original request.As the above example illustrates, service orchestration is a fundamental aspect of successfully [implementing SOA](https://www.mulesoft.com/integration-solutions/soa/service-oriented-architecture). In a truly service oriented architecture, new applications are created by new orchestrations of existing services - not by writing new code.

## **The Challenges of Service Orchestration and SOA**

From the surface, service orchestration and SOA are relatively simple concepts. For enterprises faced with [integration challenges](https://www.mulesoft.com/resources/cloudhub/cloud-cause-and-solution-new-integration-challenges), skyrocketing IT budgets and increasingly complex infrastructures, building new applications with granular and reusable software components is an understandably attractive approach to creating more agile and competitive systems and reducing time to market.

Service orchestration and SOA, however, can be difficult to achieve without the right set of tools. In its early days, CTOs of large companies eagerly adopted SOA and went about implementing it with a rip and replace model. Such an approach resulted in high financial costs as well as major time investments since it often required developers to orchestrate services programmatically (i.e. write new code), defeating the ultimate purpose of adopting SOA.What was needed was a simpler and more flexible way to perform service orchestrations and implement SOA. The enterprise service bus (ESB) emerged as the go-to mechanism for service orchestration and SOA.

## **Using the Right Platform for Service Orchestration and SOA**

ESBs and other integration platforms make the service orchestration process much simpler and eliminate the need for custom coding. ESBs enable enterprise application integration (EAI) and act as orchestrators by allowing services to communicate with each other.

Mule as an ESB offers unparalleled interoperability and flexibility by making it possible to reuse service components of any type and exchange messages of any format both within the enterprise and without. With the right platform, SOA delivers on its promises to align business processes with IT systems and reduce costs while remaining agile and robust enough to handle changes in customer demands and the integration of new applications. When the composition of new business services is made simple and intuitive, SOA can be successfully adopted - one service orchestration at a time.