Fundamentals of Service-Oriented Architectures

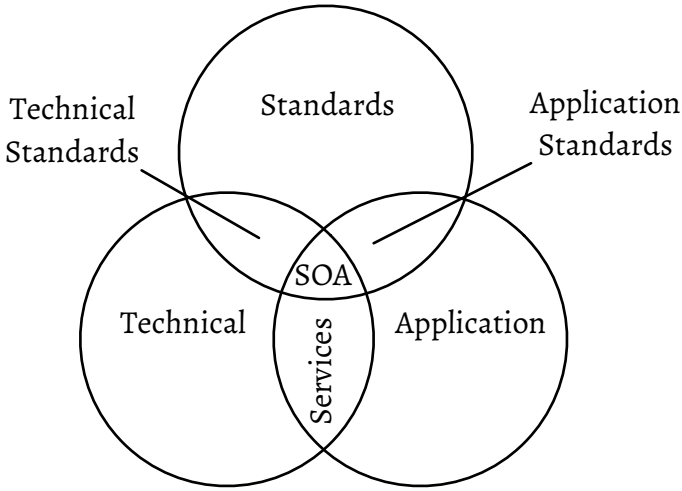
The concept of Service-Oriented Architectures (SOAs) came from the business world as a means of building large-scale distributed applications (businesses). The main idea is that by exposing every function or service that the business performs as a Service Interface, that the business will become more agile and flexible.

As SOA comes from the business world, it is largely based on practice and does not have a strong theoretic foundation. Some aspects of SOA have received theoretic foundations. For example, the Business Process Execution Language (BPEL), which is used to implement business processes in a SOA, can be transformed into Petri-Nets[1].

One area missing these fundamental theoretical foundations is, surprisingly, the core concepts of SOA itself. This is the area which I am exploring and hope to better understand.

The first core concept that seems to exist is that SOAs intentionally mix the technological, standards, and application aspects of building distributed applications around the idea of service orientation. By mixing these three aspects, SOA recognizes that successful distributed applications require more than just technology.

[1] N. Lohmann, H.M. W. Verbeek, C. Ouyang, C. Stahl, and W.M. P. van der Aalst. Comparing and Evaluating Petri Net Semantics for BPEL. Computer Science Report 07/23, Eindhoven University of Technology, 2007.



The second core concept that seems to exists is that a deployed SOA is a set of independent, executing processes (i.e., executing computation process, not a business process) which only interact through their service interfaces. The best abstraction for this interaction seems to be message passing. The message passing model has been used as a basis for several process calculus based models for SOA interactions[2].

The third core concept is that of Service Orientation, which allows the processes from the second concept to act in two roles: provider and consumer. There is no restriction that a process may only fulfill one one.

---

[2] L. Caires, R. De Nicola, Rosario Pugliese, V. T. Vasconcelos, and G. Zavattaro. Core Calculi for Service-Oriented Computing. Lecture Notes in Computer Science, 2011, Volume 6582/2011, 153-188, DOI: 10.1007/978-3-642-20401-2_8

These core concepts, when fully developed, are meant to be a solid foundation to build easy to use and deploy systems on. An example improvement, which builds on the core ideas but does not change them, is the set of Service Interaction Patterns[3]

[3] G. Decker and F. Puhlmann. Formalizing Service Interactions. Lecture Notes in Computer Science, 2006, Volume 4102/2006, 414-419, DOI: 10.1007/11841760_32