Bekanntlich stieg in den vergangenen Jahrzehnten sowohl Rechenleistung, als auch Speicherkapazität von Computersystemen. Ebenso wuchs mit der Nachfrage nach geeigneter Informationstechnologie die Abhängigkeit der Gesellschaft davon. Man stelle sich vor diesem Hintergrund vor, Software müsse analog zu Automobilen vor Erfindung des Taylorismus in manueller Kleinarbeit und im Großen und Ganzen in einem Schritt erstellt werden. Um beispielsweise ein Betriebssystem wie Windows XP annähernd in der heute vorliegenden Qualität zu realisieren, wären übermenschliche Planung- und Fingerfertigkeiten vonnöten. Daher ist leicht einsichtig, dass nur Modularisierung zum Erfolg führen kann, wenn einigermaßen große Computersysteme erstellt werden müssen.

Diese Modularisierung begann auf prozeduraler Ebene (Funktionen und Funktionsbibliotheken) und führte in Form der Objektorientierung zu modularisierten Daten, um schließlich das Paradigma von semantisch in Komponenten gekapselter Software zu begründen. Heute existieren mehrere Standards für so genannte Komponententechnologien nebenher.

Moderne Bestrebungen, welche darauf abzielen, Geschäftsabläufe und die dabei benötigten Daten wiederverwendbar und zueinander kompatibel zu machen, führen die Modularisierung auf einer abstrakteren Stufe sogar noch weiter. Diese Service-orientierten Architekturen (SOA) stellen momentan die gröbste Granularität an Modularisierung dar. Analog zu den Komponententechnologien, die in den 90er Jahren ausreiften, befindet sich die SOA-Technologie derzeit in einer Konsolidierungsphase. Abseits der Webservices, welches derzeit die bekannteste und verbreitetste SOA-Anwendung ist, stehen Umsetzungen noch weitgehend aus.

Im Folgenden soll die Verbindung zwischen den wichtigsten der nunmehr etablierten Komponententechnologien und der noch neuen SOA-Technologie untersucht werden. Aufgeworfene Fragestellungen beziehen sich vor allem darauf, inwieweit die einer Architektur jeweils unterliegende Komponententechnologie die Charakteristika von SOA von sich aus unterstützt und wie flexibel die Technologie auf Entwicklungen in der noch jungen SOA-Historie reagieren kann. Insbesondere soll die Enterprise Java Beans Technologie betrachtet werden, um nachzuweisen, dass sie aktuell am geeignetsten ist, um mit ihr SOA-Systeme zu realisieren.