supporting composite services that combines services of heterogeneous SOAs. Based on the SOC standard, services are seen as individual units based on the functionality they provide and not as pieces of software developed using a particular SOA. Unfortunately, in today's real world applications, SOC development is tightly coupled to the particular SOA of the services. Being able to create composite services independently of their particular SOA, will have an impact in the number of services that can be selected. This could result in an increase in flexibility, availability, and functionality of individual and composite services. This approach can also have the effect of increasing scalability, reduce the cost, and require less development time and learning curve.

In this work a composition framework that allows creating composite services independently of the SOA of the individual services is proposed, developed, and tested. The requirements for creating composite services independently of their particular architecture are elicited. A discussion on how to incorporate these requirements into the SOC standard is presented. A SOA-independent language to specify the workflow of composite services, inspired in technology currently under use is defined. The framework developed, composes the services using this SOA-independent language, and creates an implementation of the composite service that supports services from heterogeneous SOAs. This composition process is completely automatic and platform, language, and SOA independent. Services used during the composition process are included without requiring any changes to their original implementation and without translating services from one SOA to another. During the composition process, the framework automatically checks a set of safety properties of the composite service and the services interactions. The framework is designed with a selection algorithm that capitalizes on performance, as our work focuses on improving this non-functional requirement for composite services.

A mechanism is presented to ensure that these compositions of heterogeneous services satisfy the defined safety criteria. To perform these checks formal software analysis techniques are used, specifically model checking. Different model checking techniques are incorporated as a fundamental part of our composition framework. Based on the services properties, they are categorized as baseline or extended. These different categories of