SOA Manifest

Service oriëntatie is een gedachtegoed dat richting geeft aan wat je doet. Service georiënteerde architectuur (SOA) is de technologie architectuur die het gevolg is van het toepassen van service oriëntatie principes.

We passen service oriëntatie toe om organisaties te helpen systematisch duurzame business waarde te leveren, waarbij de wendbaarheid toeneemt en de kosten dalen, in overeenstemming met veranderende behoeften van de business.

Vanuit onze praktijkervaring, zijn we tot de volgende prioriteitstelling gekomen:
Business waarde gaat boven technologie strategie

Strategische doelen gaan boven projectgebonden voordelen

Intrinsiek vermogen tot samenwerking gaat boven maatwerk integraties

Herbruikbare services gaan boven doelspecifieke oplossingen

Flexibiliteit gaat boven optimalisatie

Evolutionaire verfijning gaat boven het nastreven van initiële perfectie

Hoewel wij de onderwerpen rechts waarderen, hechten wij méér waarde aan de onderwerpen links.

Richtinggevende Principes

We hanteren deze principes:

Respecteer de sociale en gezagsstructuur van de organisatie.

Onderken dat SOA uiteindelijk verandering vereist op vele niveaus.

De mate waarin SOA geïmplementeerd wordt kan variëren. Houd de inspanningen beheersbaar en binnen betekenisvolle kaders.

Producten en standaarden alleen zullen niet leiden tot SOA, noch zullen ze het service oriëntatie gedachtegoed voor je implementeren.

SOA kan gerealiseerd worden met behulp van een scala van technologieën en standaarden.

Stel een samenhangende set van bedrijfsbrede standaarden, normen en voorschriften vast, gebaseerd op industrie-, de facto en community standaarden.

Streef naar uniformiteit aan de buitenkant, sta tegelijkertijd diversiteit toe aan de binnenkant.

Identificeer services door business en technologie belanghebbenden te laten samenwerken.

Maximaliseer service gebruik door rekening te houden met zowel huidig als toekomstig gebruik.

Verifieer dat de services voldoen aan de behoeften en doelstellingen van het bedrijf.

Pas services en hoe ze georganiseerd zijn aan op basis van daadwerkelijk gebruik.

Scheid de verschillende aspecten van een systeem op basis van de snelheid waarmee ze veranderen.

Beperk impliciete afhankelijkheden en publiceer alle externe afhankelijkheden met als doel robuustheid te vergroten en de impact van verandering te verkleinen.

Organiseer elke service rond een samenhangende en onderhoudbare eenheid van functionaliteit, dit op elk abstractieniveau.

Authors

Ali Arsanjani
Grady Booch
Toufic Boubez
Paul C. Brown
David Chappell
John deVadoss
Thomas Erl
Nicolai Josuttis
Dirk Krafzig
Mark Little
Brian Loesgen
Anne Thomas Manes
Joe McKendrick
Steve Ross-Talbot
Stefan Tilkov
Clemens Utschig-Utschig
Herbjörn Wilhelmsen

Vertalers

Laurens Van Berge Henegouwen
Arco Keijzer
Johan Kumps
Brian Lokhorst
Edwin Smink