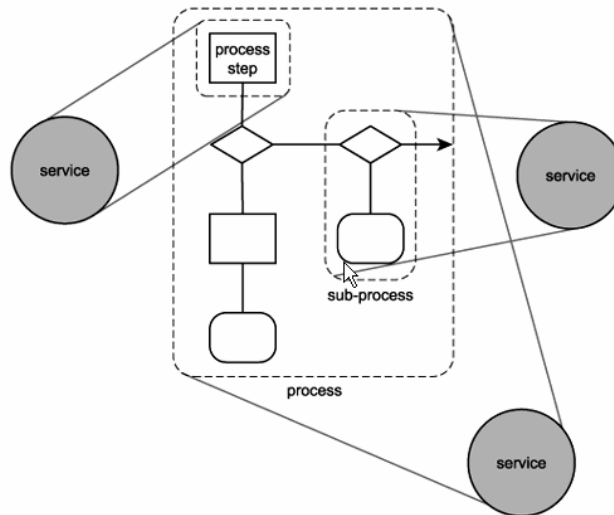


menjadi beberapa langkah. Misalkan penyediaan bahan, pemeriksaan keberadaan bahan, proses masak, dan sebagainya.

Setelah seluruh permasalahan dapat dibagi dalam beberapa *service*, solusi dari permasalahan tersebut harus bisa diselesaikan dengan memungkinkan seluruh *service* berpartisipasi dalam sebuah orkestrasi. Untuk itu ada beberapa permasalahan yang harus dimiliki oleh *service*, yaitu bagaimana *service* berhubungan, bagaimana *service* berkomunikasi, bagaimana *service* didesain, dan bagaimana pesan antar *service* didefinisikan [ERL05].



Gambar II-9 Enkapsulasi *business process* dengan *service*

Pembagian berdasarkan *service* ini sesungguhnya bukan sesuatu yang baru, karena telah banyak diterapkan. Namun hal baru dari pendekatan *service-oriented* ini terkait dengan sifat-sifat yang dimilikinya [ERL05], yaitu:

1. *Loosely coupled*, yaitu setiap *service* berdiri sendiri secara independen dan tidak tergantung *service* lain untuk berjalan. Ketergantungan diminimalisir sehingga hanya butuh mekanisme komunikasi satu sama lain.
2. *Service contract*, yaitu setiap *service* memiliki kesepakatan mengenai cara untuk komunikasi
3. *Autonomy*, yaitu *service* memiliki hak penuh terhadap semua logik yang dienkapsulasi
4. *Abstraction*, yaitu *service* tidak memperlihatkan bagaimana logik diimplementasi di dalamnya.
5. *Reusability*, yaitu logik dibagi menjadi sekumpulan *service* yang dapat memudahkan *reuse*.
6. *Statelessness*, yaitu *service* tidak memiliki status tertentu terkait dengan aktivitas yang dilakukannya.

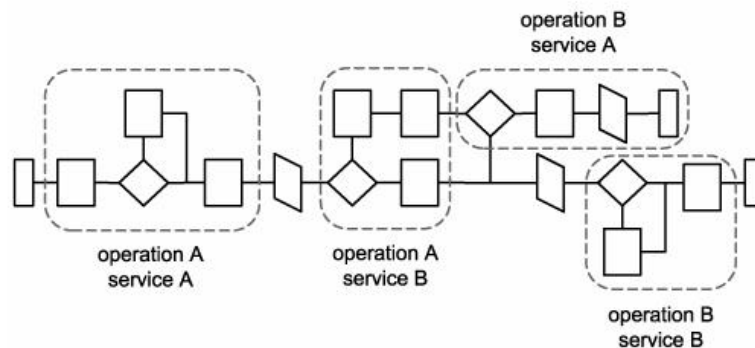
7. *Discoverability*, yaitu *service* didesain untuk deskriptif sehingga bisa ditemukan dan diakses melalui mekanisme pencarian tertentu.

2.2.2 Komponen-Komponen SOA

Bila dilihat pada penjelasan sebelumnya, SOA terdiri atas sekumpulan *service*. Namun sekumpulan *service* tidak cukup untuk membentuk sebuah arsitektur ini. Menurut [ERL05], SOA terdiri atas empat komponen, yaitu:

1. *Message*, yaitu data yang dibutuhkan untuk menyelesaikan sebagian atau sebuah unit kerja, yang dipertukarkan antara satu *service* dengan yang lainnya
2. *Operation*, yaitu fungsi-fungsi yang dimiliki oleh sebuah *service* untuk memproses *message* hingga menghasilkan sesuatu. Fungsi-fungsi inilah yang nantinya akan saling berinteraksi untuk menyelesaikan sebuah unit kerja
3. *Service*, merepresentasikan sekumpulan *operation* yang berhubungan untuk menyelesaikan sekumpulan unit kerja yang berhubungan
4. *Process*, merupakan *business rule* yang menentukan operasi mana yang digunakan untuk mencapai tujuan tertentu.

Ilustrasi dari keempat komponen ini dapat dilihat pada Gambar II-10.



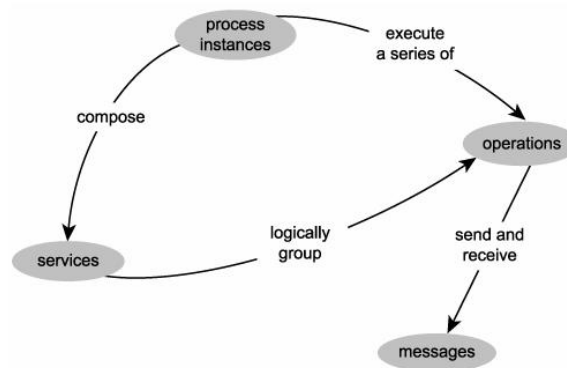
Gambar II-10 Ilustrasi Operations dan Services [ERL05]

Komponen-komponen tersebut terhubung satu sama lain dengan deskripsi sebagai berikut [ERL05]:

1. Sebuah *operation* mengirim dan menerima *message* untuk mengerjakan sesuatu
2. Sebuah *operation* kebanyakan didefinisikan oleh *message* yang memprosesnya
3. Sebuah *service* mengelompokkan sekumpulan *operation* yang berhubungan
4. Sebuah *service* didefinisikan oleh *operation* yang membentuknya
5. Sebuah instans dari *service* dapat mengkomposisi *service* lain
6. Sebuah instans dari *process* tidak harus didefinisikan oleh *service* karena mungkin hanya membutuhkan sebagian dari fungsionalitas yang diberikan oleh *service*

7. Sebuah instans dari *process* memicu sekumpulan *operation* berjalan untuk menyelesaikan proses otomatisasi
8. Setiap instans dari *process* didefinisikan secara parsial *operation* yang digunakannya

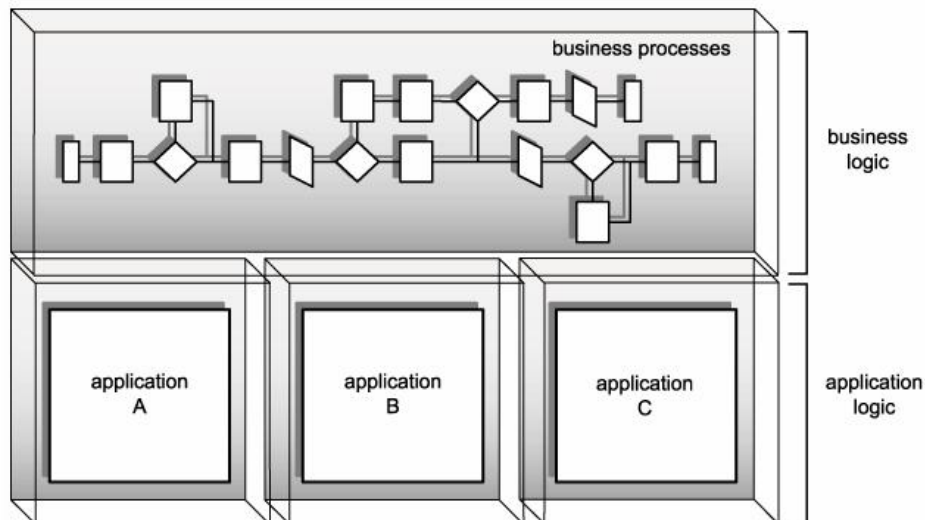
Ilustrasi dari keterhubungan seluruh komponen ini dapat dilihat dari Gambar II-11.



Gambar II-11 Keterhubungan antar komponen SOA [ERL05]

2.2.3 Layering pada SOA

Perangkat lunak yang tidak menggunakan SOA secara umum dapat dibagi menjadi dua *layer* utama, yaitu *application layer* di mana aplikasi dijalankan dan *business process layer* yang mendeskripsikan bagaimana proses bisnis dalam perusahaan berjalan. Proses bisnis organisasi akan didefinisikan dalam aplikasi bersamaan dengan kode program yang bersifat teknis. Hal tersebut dapat dilihat pada Gambar II-12.

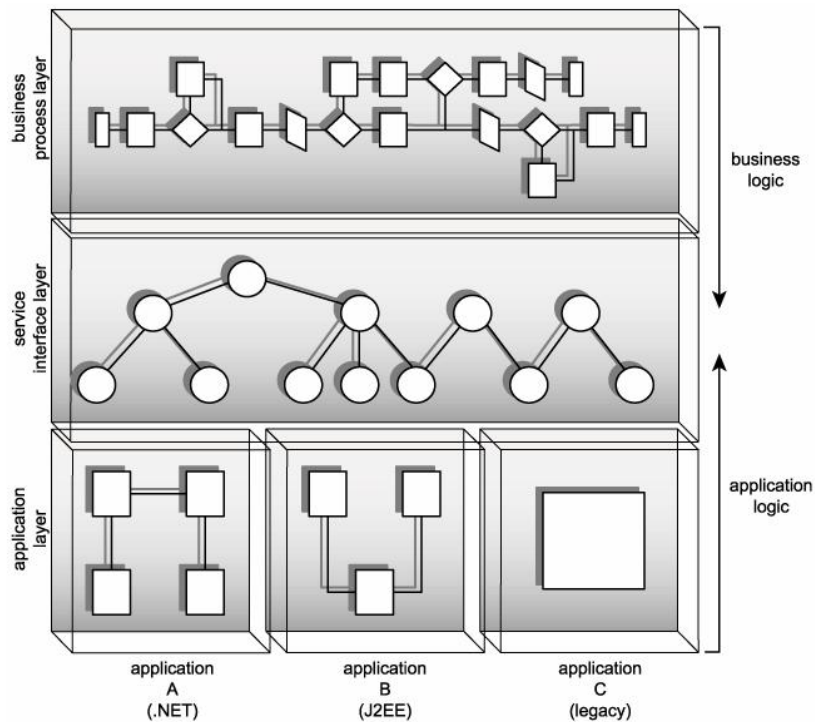


Gambar II-12 *Business Logic* dan *Application Logic* pada Enterprise [ERL05]

Dalam implementasi SOA, konsep *service-oriented* yang dipegangnya diimplementasikan dalam sebuah *layer* di antara *business process layer* dan *application layer* yang mana keduanya merupakan bagian dari *enterprise logic*. *Layer* tersebut dinamakan *service interface layer*, dan dapat dilihat pada Gambar II-13.

Fungsi dari *layer* ini adalah mengenkapsulasi logik yang ada di *application logic*, sekaligus *business process* yang ada di *business logic*. Dengan pendekatan ini, aplikasi bisa lebih dimodularisasi dan menggunakan berbagai macam teknologi. Seperti dapat dilihat pada Gambar II-13, teknologi .NET pada aplikasi A, J2EE pada aplikasi B, dan aplikasi C akhirnya akan dienkapsulasi oleh *service interface layer*.

Service interface layer ini juga terbagi atas berbagai lapisan abstraksi, yaitu *application service layer*, *business service layer*, dan *orchestration service layer*. Ilustrasi dari ketiganya dapat dilihat pada Gambar II-14.



Gambar II-13 Implementasi *layer* pada *enterprise* [ERL05]

2.2.3.1 *Application Service Layer*

Application service layer. menyediakan sekumpulan *service* yang spesifik untuk mengenkapsulasi teknologi tertentu yang terdapat di dalam *application logic*. *Service* yang disediakan dalam *layer* ini akan melakukan abstraksi terhadap semua logik yang tidak terkait