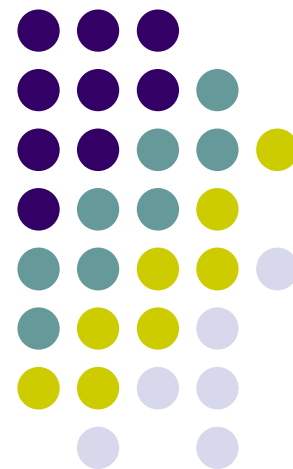


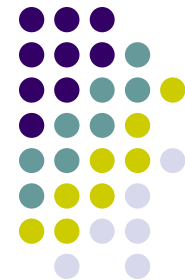
# Мрежово програмиране

SOA (service-oriented architecture)





- Бързо развиващите се бизнес отношения са изисквали мрежовите приложения, използващи различни протоколи, да могат да си обменят данни. Така е възникнала задачата за интеграция на приложенията.

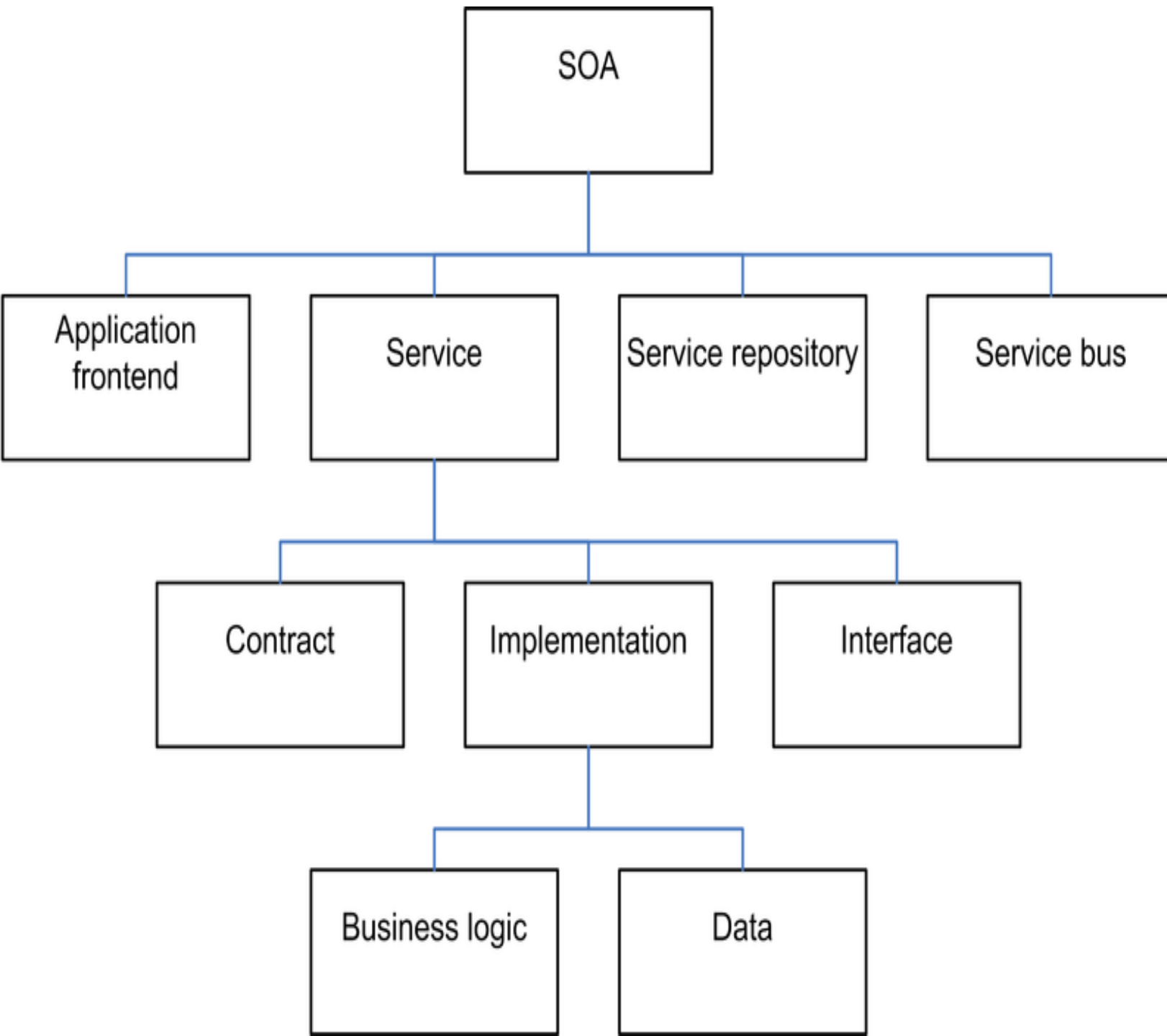


- SOA не е технология, а *философия, концепция, парадигма, подход* за построяване на корпоративни информационни системи, интеграция на бизнеса и на информационните технологии

## Clive Finkelstein, автор на инфотехниката (information engineering):



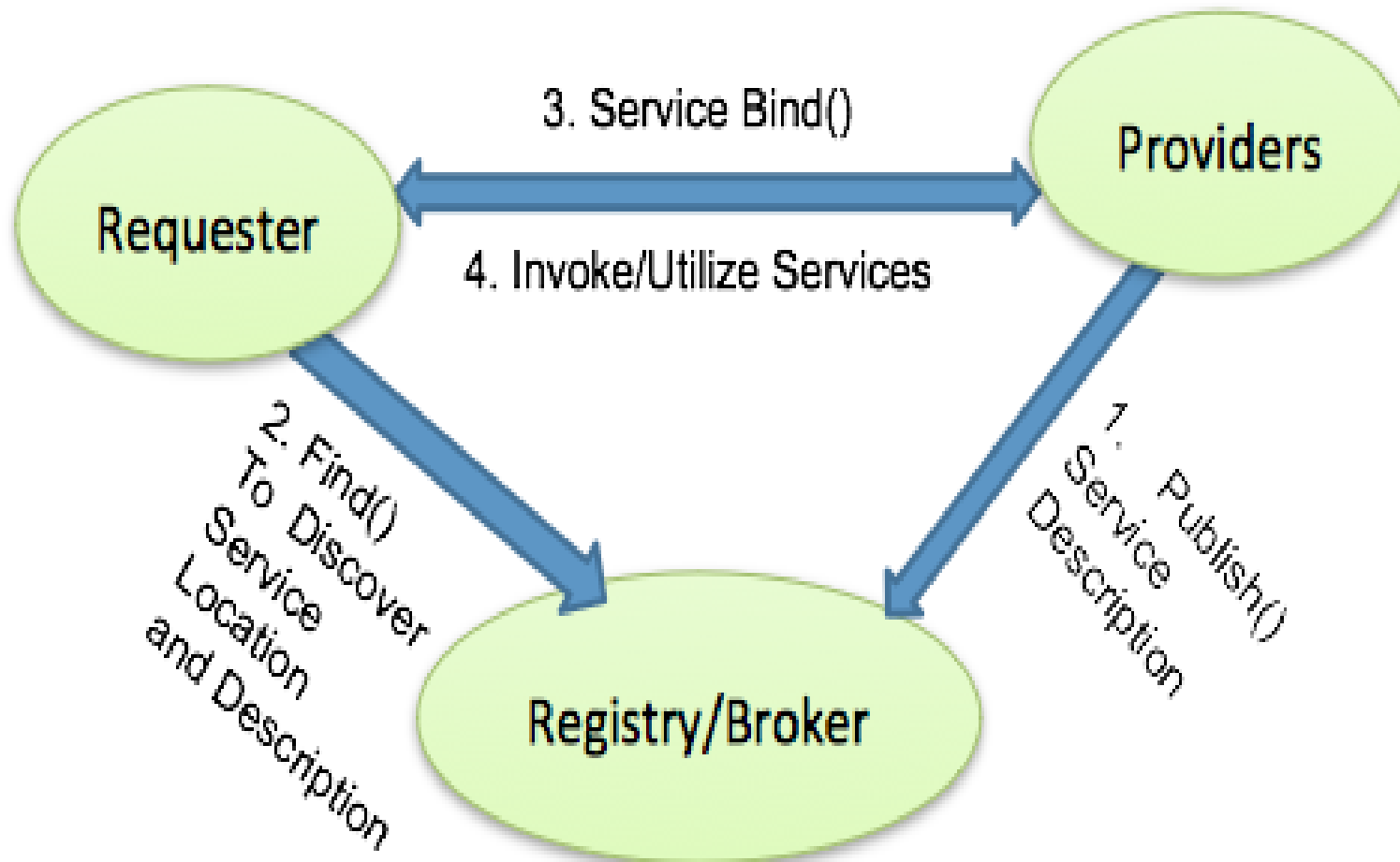
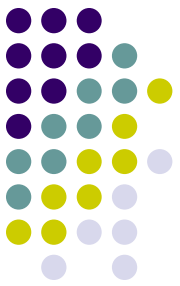
- Тази технология за управления на бизнес процеси е голяма крачка напред от гледна точка на повишаването на ефективността на разработването на системите; по значимост може да се сравни със създаването в края на 50-те години на компилаторите на език от високо ниво.

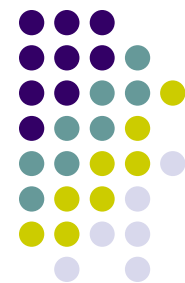




- SOA не е нещо ново: IT секторите на компаниите успешно са разработвали приложения, поддържащи ориентирана към услуги архитектура вече много години - още преди появата на XML и уеб услугите.
- SOA не е технология, а начин за проектиране и организация на информационна архитектура и бизнес-функционалност.
- Покупката на най-новите продукти, реализиращи XML и уеб услуги, не означава построяване на приложения в съответствие с принципите на SOA.

В най-общ вид SOA предполага наличието на трима основни участници: доставчик на услугата, потребител на услугата и регистър на услугите



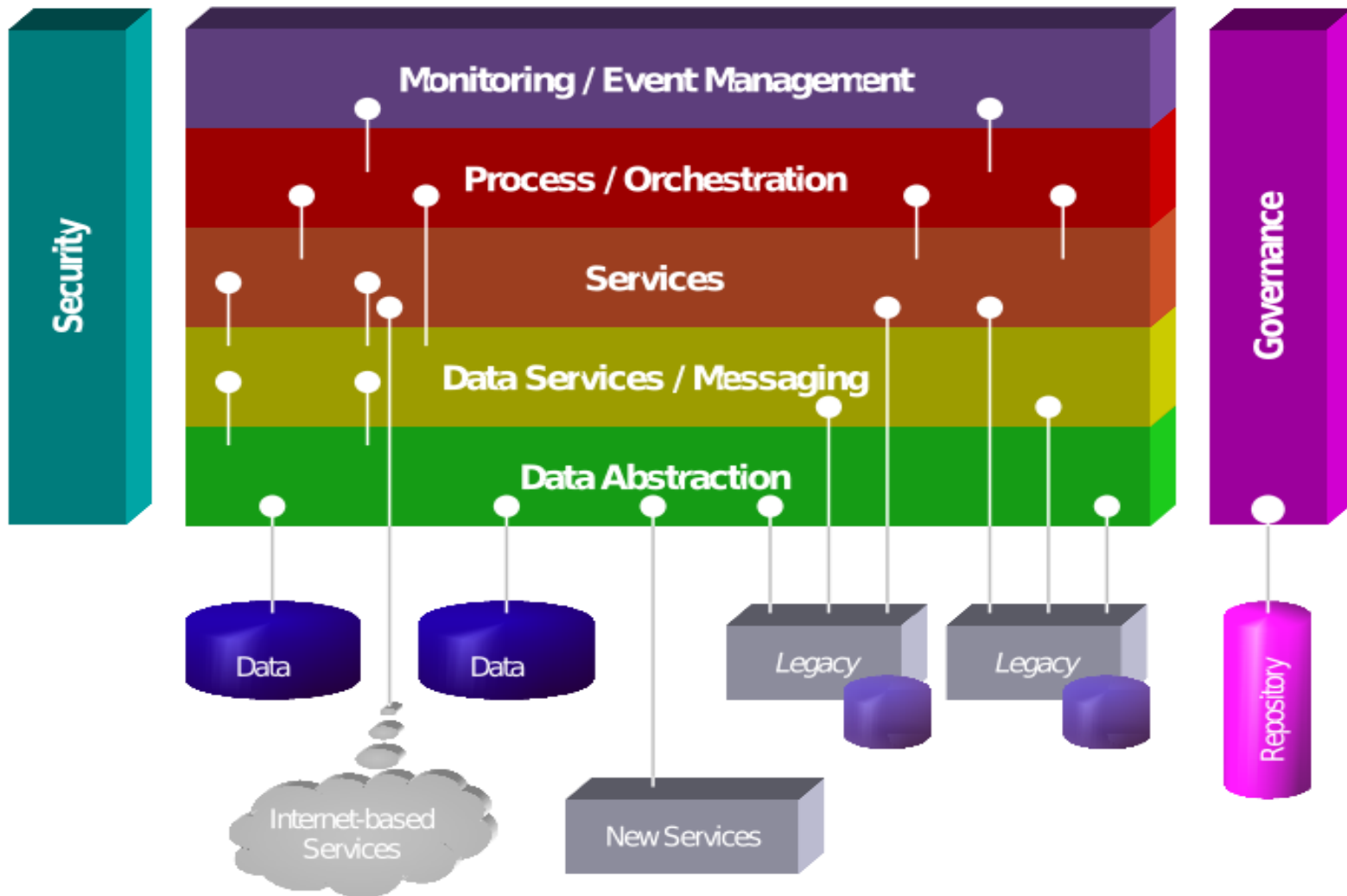


# Тактически предимства на SOA:

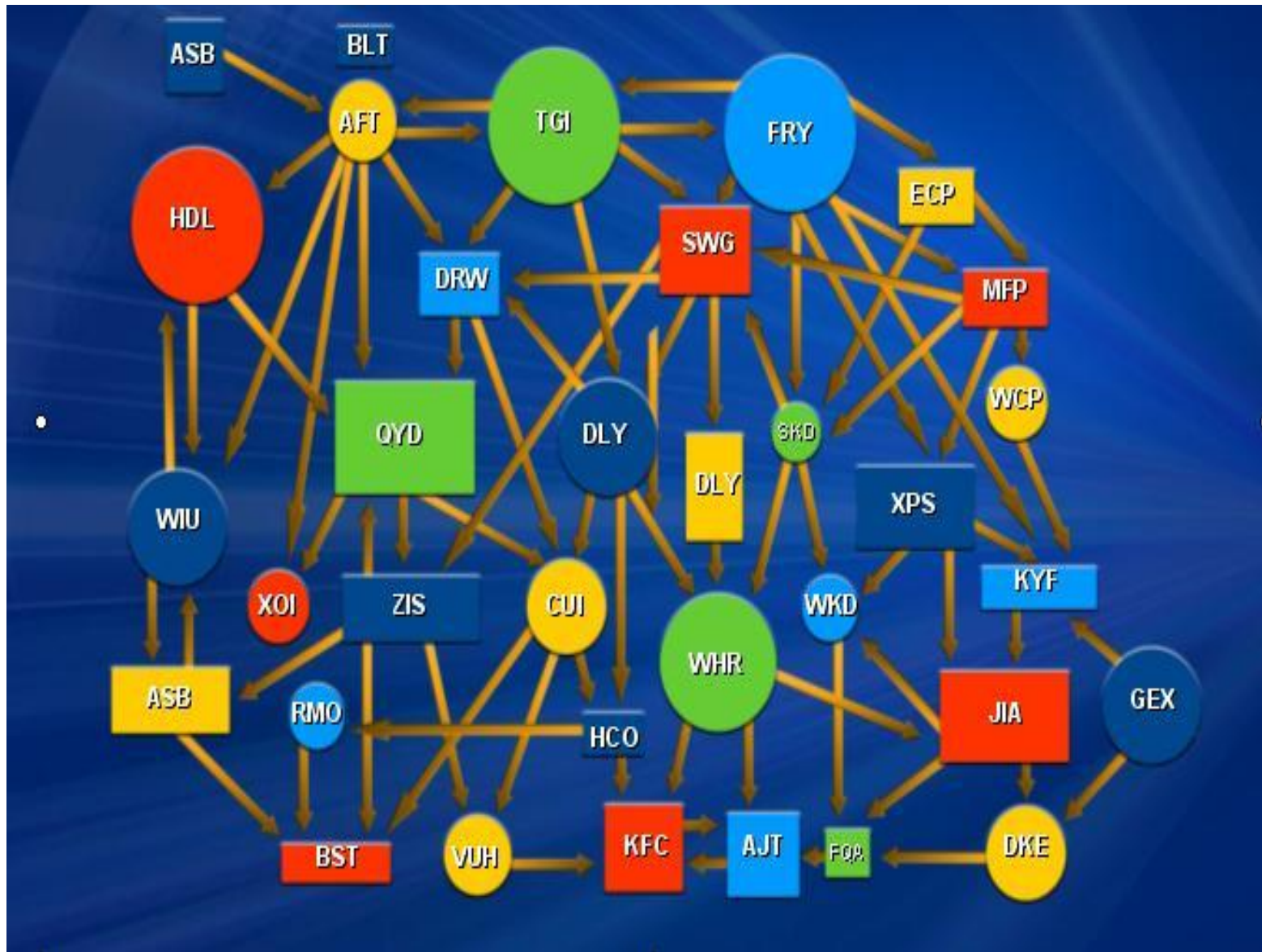
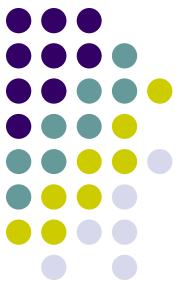
- По-лесно разработване и внедряване на приложенията.
- Използване на текущите инвестиции.
- Намаляване на риска, свързан с внедряването на проекти в областта на автоматизацията на услугите и процесите.
- Възможност за непрекъснато подобряване на предоставяната услуга.
- Съкращаване на броя на обръщенията към техническа поддръжка.
- Повишаване на показателя на възврата на инвестициите (ROI).



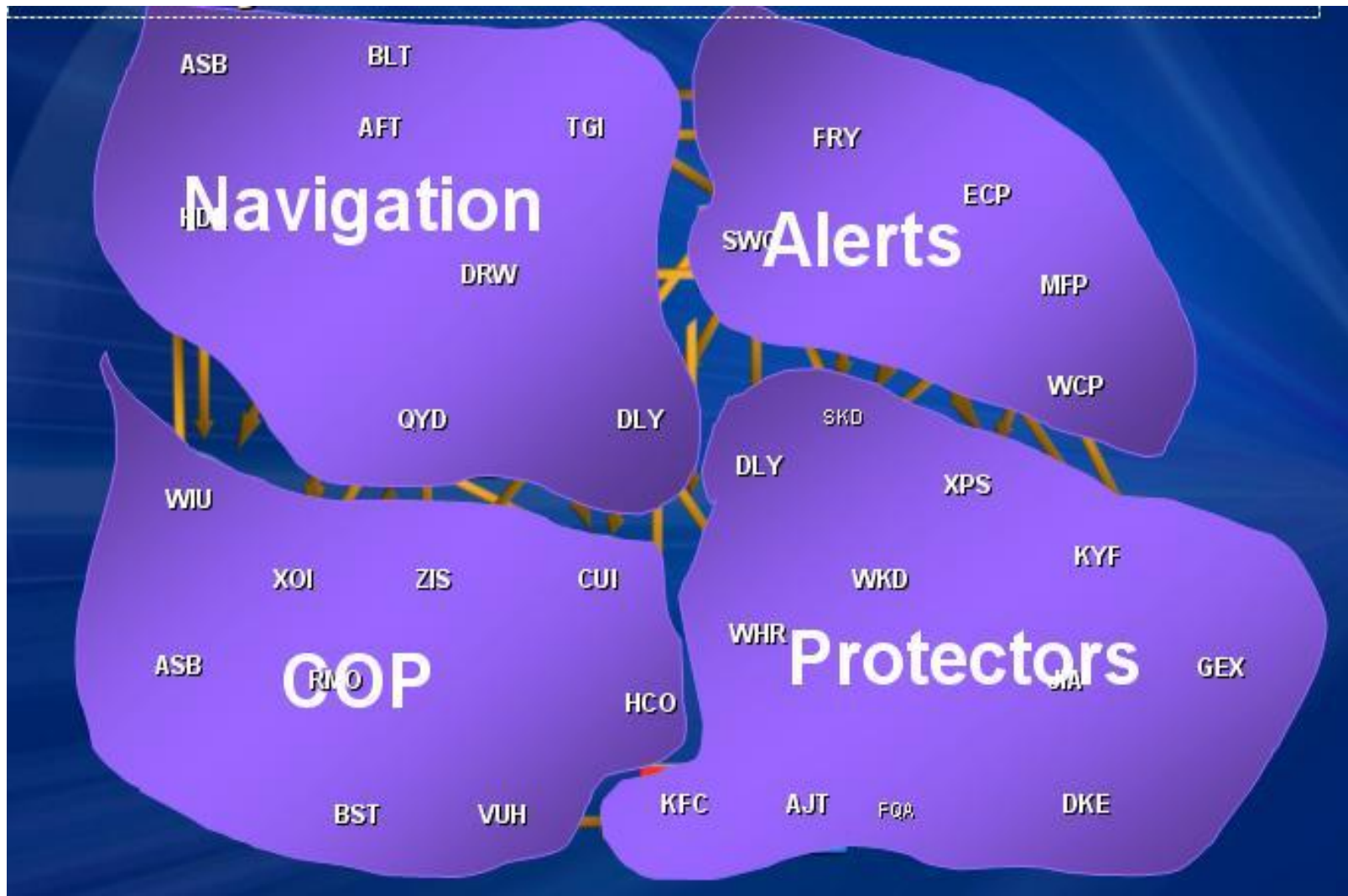
# SOA мета модель



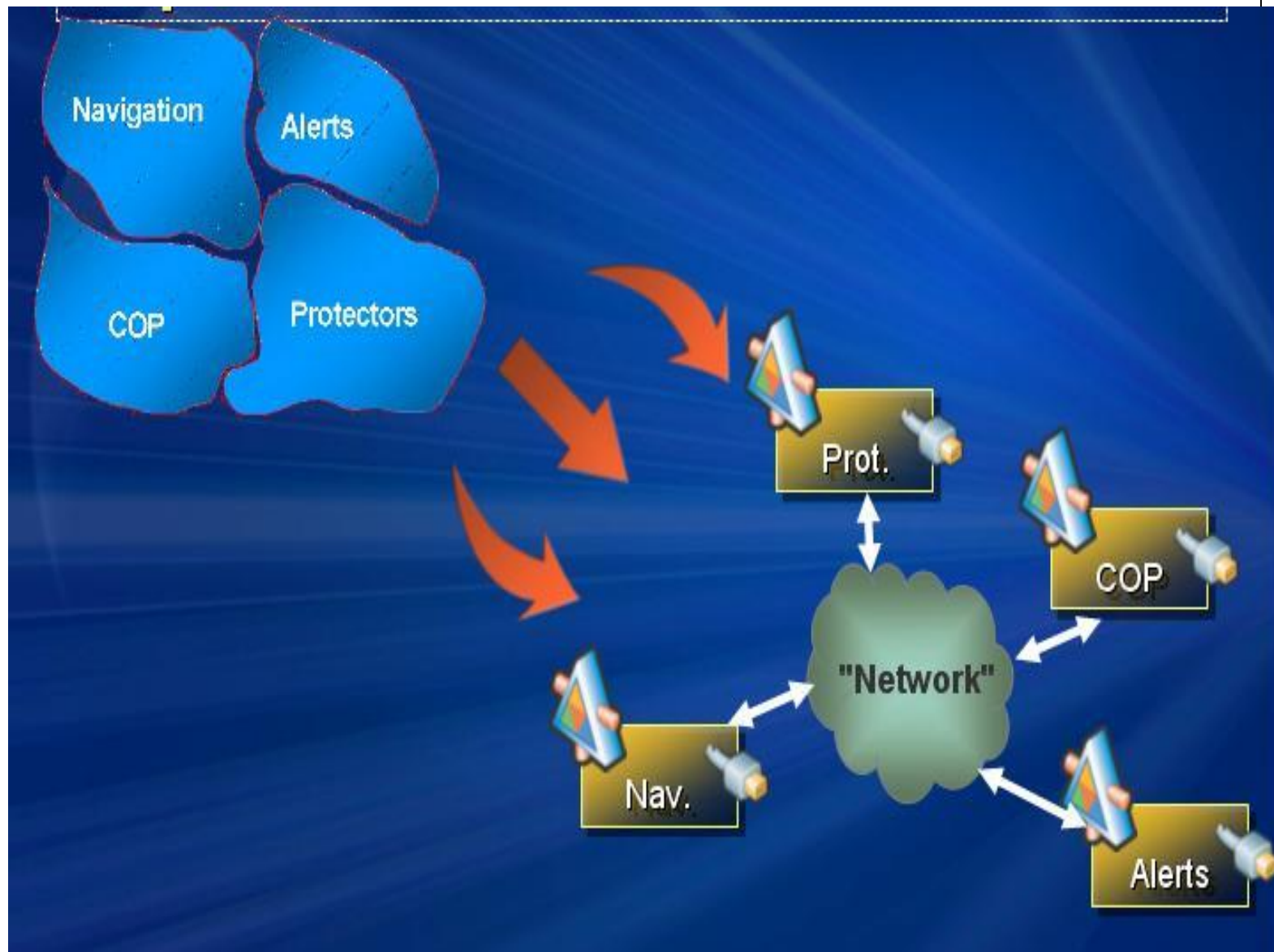
# 1. Анализ на бизнес процесите



## 2. Разделяне на областите

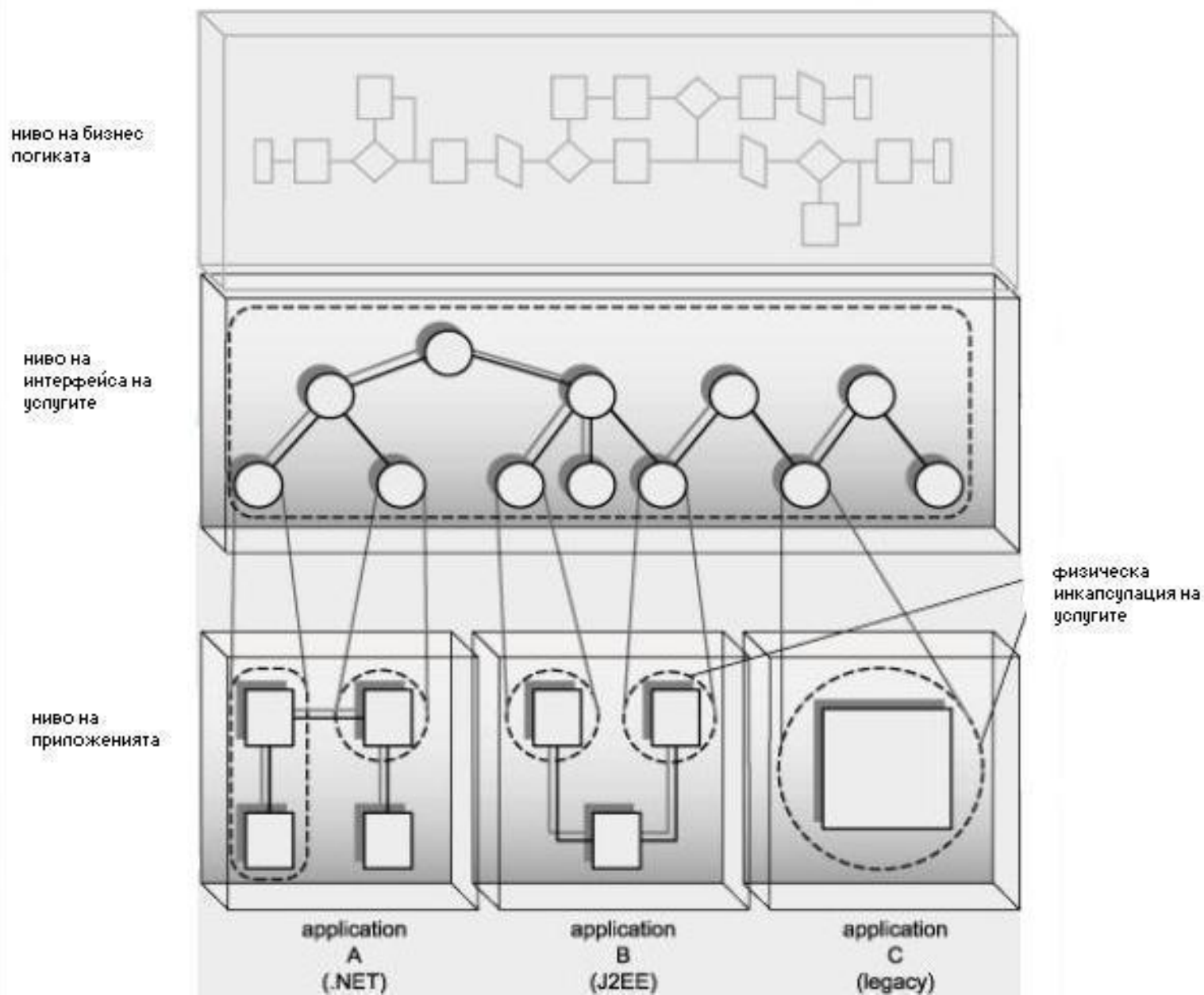


### 3. Проекция на услугите





# Реализация на услугите





# Базови принципи на SOA:

- Разпределено проектиране;
- Постоянство на промените;
- Последователно усъвършенстване;
- Рекурсивност.

# 1. Услугата следва да допуска повторно използване



- SOA системите са длъжни да поддържат повторно използване на всички услуги, независимо от моментните изисквания към техните функционални особености;
- Последното ще позволи да се опрости разширяването и развитието на системите, отказване от “обвивки” над услугите за тяхната пренастройка за решаване на нови задачи;
- Така всяка операция на услугата е длъжна да поддържа повторно използване.

Може да се използва от няколко абоната.



## 2. Услугите са длъжни да осигуряват формален договор за ползване. Договорът за услугата предоставя следната информация:

- Адрес на крайната точка (service endpoint);
- Всички операции, предоставяни от услугата;
- Всички съобщения, поддържани от всяка операция;
- Правила и характеристики на услугата и нейните операции.



### 3. Услугите са длъжни да са слабо свързани



- Необходимо е осигуряване на цялостност на системата в рамките на развитието на системата, независимо от вариантите на развитие;
- Системата от услуги е слабо свързана ако услугата може да получава знания за друга услуга, оставайки независима от вътрешната реализация на логиката на дадената услуга.

## 4. Услугите са длъжни да се абстрахират от вътрешната логика



- Всяка услуга е длъжна да работи като черна кутия;
- Това е едно от изискванията, осигуряващи слабата свързаност на услугите.



## 5. Услугите задължително са съвместими

- Услугата може самостоятелно да реализира логиката, а също и да използва други услуги за своята реализация;
  - Услугите трябва така да са проектирани, че да се поддържа възможността за използването им в качеството на елементи на друга услуга;
- Този процес се нарича оркестрация на услугите.



## 6. Услугите трябва да са автономни

- Областта на бизнес логиката и ресурсите, използвани от услугата, следва да са ограничени в зададени граници;
- Въпросът за автономността е най-важният аргумент при разпределянето на бизнес логиката на отделните услуги.

Има два типа автономност:

- *Автономност на нивото на услугата:* границите на отговорност на услугите са разделени, но те могат да използват общи ресурси;
- *Чиста автономност:* бизнес логиката и ресурсите се намират под пълен контрол от услугата.

## 7. Услугите не са длъжни да използват информация за състоянието



- “Чистите” услуги са длъжни значително да ограничават обема и времето за съхраняване на информацията (в идеалния случай - само по време на изчисленията);
- Независимостта от състоянието (Statelessness) позволява да се увеличи възможността за мащабиране и повторно използване на услугите;

Това ограничение не винаги е възможно да бъде удовлетворено.



## 8. Услугите трябва да бъдат откриваеми

- Откриването на услугите позволява да се избегне случайното създаване на излишни услуги, осигуряващи логика в излишък;
- Всички методи на услугата трябва да съдържат метаданни, описващи възможностите им в системата за търсене;
- Всяка услуга трябва да предоставя колкото може повече информация за своите възможности.



Основата на Интернет: TCP/IP; HTML; XML, основани на общоприети, отворени и формално независими технологии.

- *универсалност на всяка от тези технологии*

Универсалността се изразява във възможността за използване при различните ОС, езици за програмиране, сървъри за приложения и т.н.

Така уеб услугите решават първоначалната задача за интеграцията на приложенията с различна природа и построяването на разпределени информационни системи.

В това се изразява основната принципна разлика на уеб услугите от предшествениците им.

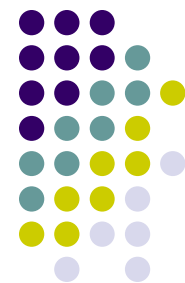


# Описание на слоевете на технологията на уеб услугите

- Функционалност;
- Качество на обслужването.

Този протоколен стек на технологиите показва йерархията в множеството от технологии на уеб услугите в съответствие с тяхното функционално предназначение.

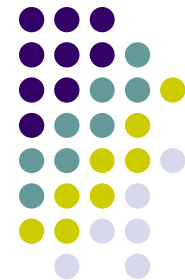




# Функционалност

- Транспортен слой;
- Комуникационен слой;
- Слой за описание на услугите;
- Слой на услугите;
- Слой на бизнес процесите;
- Слой на регистрите на услугите.

# Качество на обслужването

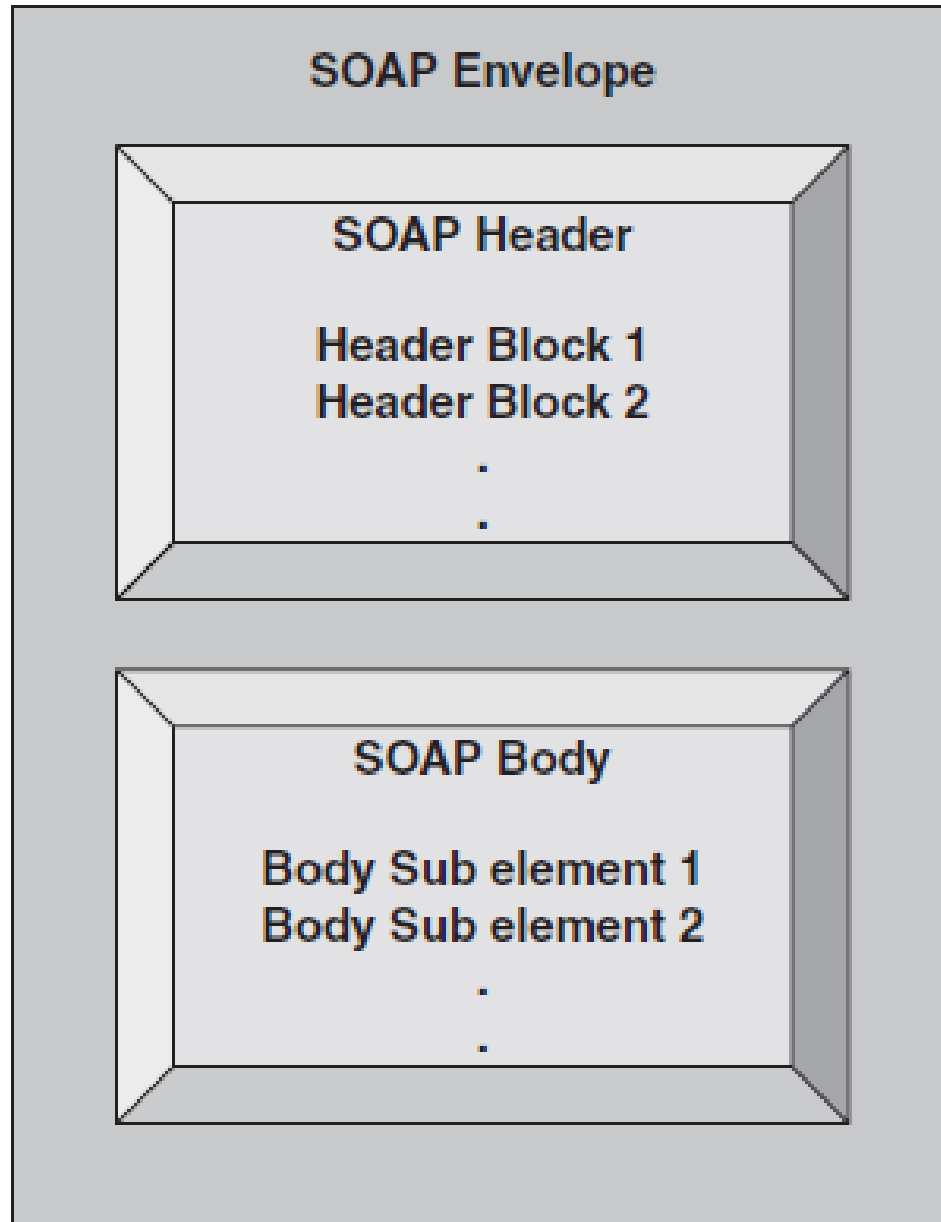


- Слой на политиките;
- Слой на безопасността;
- Слой на транзакциите;
- Слой за управление.

# Към настоящия момент технологическият фундамент на уеб услугите се образува от следните технологии:



- **XML** (Extensible Markup Language) — стандартен език за представяне на информацията
- **SOAP** (Simple Object Access Protocol) — стандартен формат (на базата на XML) за представяне на предаваните съобщения по мрежата
- **WSDL** (Web Service Description Language) — стандарт за спецификация на услугите (на базата на XML)
- **UDDI** (Universal Description, Discovery and Integration) — стандартни средства за търсене на услуги
- **WS-I Basic Profile** — стандарт за осигуряване на способност за взаимодействие (interoperability) на услугите



# Структура на SOAP съобщение



## Listing 7-2

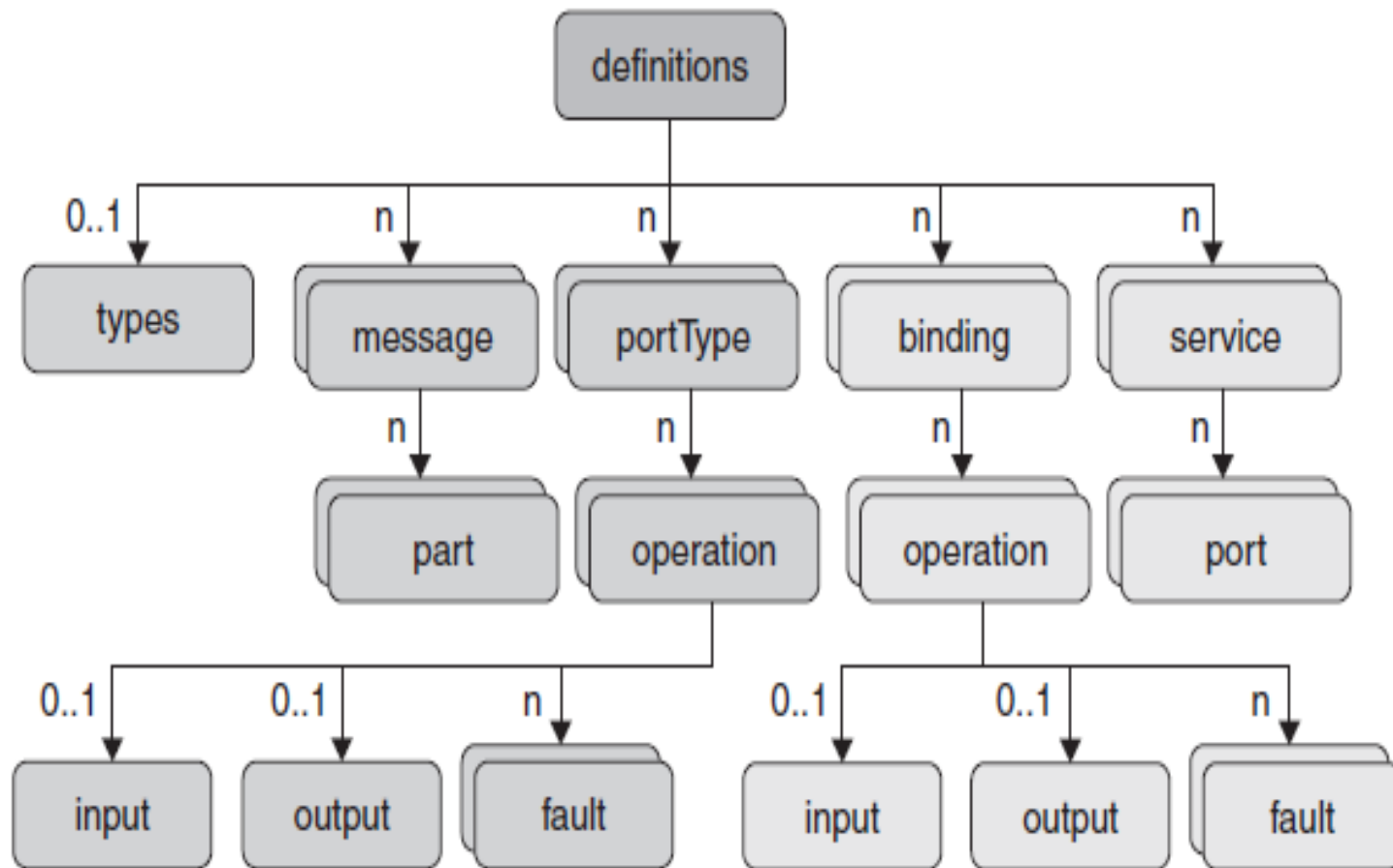
Listing 7.2: An example of SOAP message


```
<soap:envelope xmlns:soap="http://schemas.xmlsoap.org/soap/
envelope/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:header>
  </soap:header>
  <soap:body>
    <m:GetLastTradePrice xmlns:m="http://example.org/
Tradeprice" >
      <tickerSymbol> IBM </tickerSymbol>
    </m:GetLastTradePrice>
  </soap:body>
</soap:envelope>
```




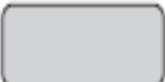
# Елементи на WSDL документ

- **Types** — определя типа на данните в съобщението от използваната услуга
- **Message** — определяне на съобщенията, които ще си обменят услугите
- **PortType** — абстрактно описание на операциите и съобщенията, използвани от услугата
- **Binding** — настройка на портовете и на операциите, свързани с протоколите и съобщенията
- **Service and port** — определя името на услугата и адреса за свързване с услугата



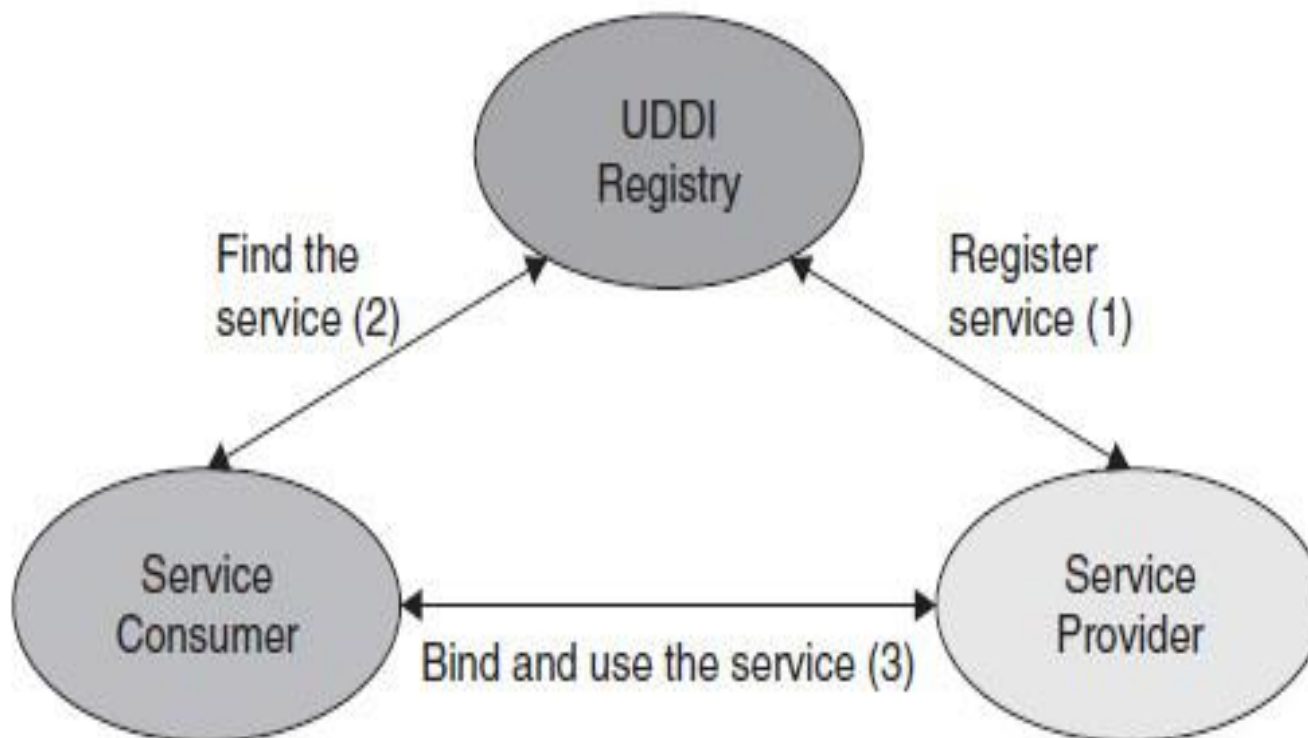
 Top container element

 Concrete implementation element

 Abstract interface element



# Използване на UDDI







## WS-I Basic Profile

WS-I (Web Service Interoperability) Profile  
определя ограниченията при използването на  
спецификациите XML, SOAP, WSDL и UDDI

- За свързване се използва само SOAP
- В качеството на приложни протоколи се използват само HTTP и HTTPs
- Забранява се използването на кодиране на съобщения, даже и SOAPencoding
- Забранява се “претоварване” с имена на операции



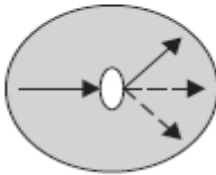
# Проблеми

- Несъответствие на протоколите на интегрируемите приложения
- Несъответствие на форматите на съобщенията, използвани при комуникацията

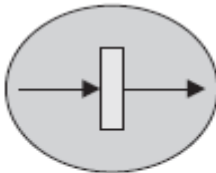


# Решение

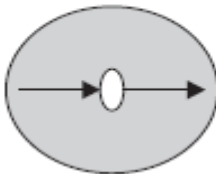
Routing



Protocol  
Switch



Data  
Transform



- ❑ Асинхронна обработка на съобщенията – по-скъпа и трудоемка
- ❑ Маршрутизация според контекста и съдържанието
- ❑ Преобразуване и превключване на протоколите
- ❑ Преобразуване на форматите на данните (съобщенията)