SOA (Service-oriented architecture)
Дефиниции, понятия (терминология)

SOA е компютърно-системен архитектурен стил за създаването и използването на бизнес процесите, опаковани като ОС услуги през жизнения им цикъл. SOA също така определя и снабдява ИТ инфраструктурата, за да позволи на различни приложения да обменят данни и да участват в бизнес процесите. Тези функции са слабо свързани с операционните системи и програмните езици, които са в основата на приложенията. SOA разделя функциите на отделни единици (ОС услуги), които могат да бъдат разпространявани по мрежа и могат да бъдат обединявани и използвани на ново за създаването на бизнес приложения. Тези ОС услуги общуват по между си като си препредават данни от една услуга на друга или като си координират действията между две или повече услуги. Понятията за SOA са често смятани за изградени върху и разширение на по-стари понятия на разделното изчисляване и модулното програмиране.

Компаниите имат да извървят дълъг път, за да интегрират съществуващите системи, за да изпълнят ИТ поддръжката за бизнес процесите, които ще обхванат всичките сегашни и бъдещи системни изисквания нужни за управлението на бизнеса от „край до край”. Много модели могат да бъдат използвани за този край, започвайки от много трудния electronic data interchange (EDI) и стигайки до Web auctions. Модернизирайки старите техногологии като се позволява да се ползва Интернет на IDE-базирани системи, компанните могат да направят тяхните ИТ системи достъпни за външни и вътрешни клиенти; но резулатът от тези модернизации показа, че системите не са достатъчно гъвкави, за да изпълнят изискванията на бизнеса. Една гъвкава, стандартизирана архитектура трябва да има по-добра поддръжка за връзка с различни приложения и споделянето на данни. SOA точно такава архитектура. Тя обединява бизнес процесите като структурира голям набор от приложения като ad hoc колекция от малки модули наречени услуги.

Тези приложения могат да бъдат изпозлвани от различни групи от хора намиращи във и извън компанията, и новите приложения изградени от комбинирани услуги предоставят по-голяма гъвкавост и ендообразие. Изграждейки всички приложения от един и същ запас от услуги прави постигането на целта много по-лесно и по-лесна за внедряване в компаниите.

SOA изгражда своите приложения на базата на софтуерните услуги. Услугите са свойствено несвързани функциониращи единици, които нямат вградена необходимост една от друга. Те обикновено изпълняват функции, които повечето хора разпознават като услуга, като попълване на сметка онлайн, разглеждане на банков отчет онлайн, или запазване на самолетен билет през интернет. Вместо услугите да имат вградени връзки помежду си в сорс кода им, има дефинирани протоколи, които описват как една или повече услуги могат да общуват по между си. Тази архитектура после разчита на специалистите на бизнес процесите да насочат и подредят услугите – процес познат като аранжиране(orchestration), за да се изпълнят нови или вече съществуващи бизнес системни изисквания.

1. Дефиниции на SOA

Трудно е да се даде точно определение на термина SOA. Проблема не в това, че не съществуват никакви дефиниции; проблема е в това че има много различни дефиниции. ??? Все пак всички определения са съгласни че SOA е образец за подобрена гъвкавост.

1.1. SOA е образец

SOA не е конкретна архитектура: то е нещо, което води до конкретна архитектура. Може да го наречете стил, образец, перспектива, философия или представяне. SOA не конкретен инструмент или рамка, която можеш да поръчаш. То е подход, начин на мислене, ценна система, която води до конкретни решения когато се разработва конкретна софтуерна архитектура.

От тази гледна точка SOA има едно много важно значение: не можеш да купиш SOA. Няма устройство или рецепта, с която да направиш, така че всичко да заработи. Докато прилагаш този образец към твоята конкретна ситуация, трябва да направиш характерни решения, които са подходящи за твоята ситуация.

1.2. SOA цели да подобри гъвкавостта

Основната причина да използваш SOA е заради, това че би трябвало да ти помогне във твоя бизнес. Например, ти можеш да имаш нужда от ИТ решения, които да съхраняват и управляват информация, и да ти позволяват да автоматизираш обичайните процеси които се занимават с тази информация. За да доставиш качествени решения на време, ти трябва гъвкавост. Но гъвкавостта се постига с ясна организация, роли, процеси и т.н. За това SOA трябва да се справи с всички тези аспекти.

2. На какви системи е приложима SOA

Разбира се гъвкавостта се определя много различно на различни платформи и в различни компоненти. За това важен въпрос е за какви видове софтуерни системи този образец е подходящ. И както се оказва, SOA се справя добре с голям брой системи.

2.1. Разпространени системи

Когато бизнеса расте, той започва да става все повече и повече цялостен, и все повече и повече системи и компании се въвличат. Има постоянно обединяване и постоянна промяна. SOA е добре пригодена да се справя с цялостни разпространени системи. SOA дава единиците, които се нуждаят от определени разпространени възможности, за да намерят и се възползват от тези възможности. С други думи улеснява взаимодействията между доставчиците на услуги и клиентите на услуги, позволявайки реализацията на бизнес функционалностите.

2.2. Различни собственици

В OASIS SOA Reference Model в дефиницията на SOA се казва, че разпространените възможности мога да бъдат под контрола на домейни с различна собственост. Това е много важна точка, която много често бива пренебрегвана в дефинициите за SOA.

SOA включва в себе си практики и процеси, които се базират на факта, че мрежите на разпространените системи не се контролират от едни собственици. Различни екипи, различни отдели или дори различни компании могат да управляват различни системи. Поради това различни платформи, графици, приоритети, бюджети и т.н. трябва да бъдат вмъкнати в акаунт. Тази концепция е важна за разбирането на SOA и големите разпространени системи като цяло. Начина по който се справяш с проблемите и правиш промени в средите с различни собственици и в средите в които ти контролираш всичко ще се различават неизбежно. Не можеш да осъществиш функционалност и да променяш поведението в големи системи по същия начин както в по-малки системи.

2.3. Разнородност

Друга много важна разлика между малките и големите системи е липсата на „хармония”. Всички ние знаем това от опит. Големите системи използват различни платформи, различни програмни езици и дори различен „middleware”. Те са безпорядък от главни сървъри, SAP хостове, бази от данни, J2EE приложения и т.н. С други думи те са разнородни.

В миналото много подходи са били предложени да разрешат проблема с интегрирането на разпространените системи като се премахне разнородността: „Нека заменим всички системи с J2EE приложения”, „Нека използваме CORBA навсякъде”, „Нека използваме MQ сериите” и т.н. Но всички знаем че тези подходи не работят. Големите разпространени системи с различни собственици са разнородни.(фиг.1) Това е факт, който трябва да приемем когато разработваме големи разпространени решения.

Фиг.1 Разпространените системи са разнородни

Подходът на SOA прием разнородността. Справя се с големи разпространени системи като приема и поддържа този атрибут.

Това е една от основните идеи за SOA, и може да даде на SOA силата да създаде революция. Подобна на „живите” методи на софтуерното разработване, което приема, че условията се променят вместо да се опитват да се борят срещу тези промени, SOA просто приема, че има разнородност в големите системи. Това води до много различен начин на мислене, и внезапно ние имаме начин да се справим с големите разпространени системи, който наистина работи.

3. SOA концепция

Тука са основните технически концепции на SOA, които й позволяват да се справи със системните характеристики:

Услуги
Взаимодействие между съществуващите системи
Слаба свързаност

3.1. Услуги

Софтуерното разработване е изкуството на абстракцията. Ние трябва да абстрахираме реалността по такъв начин, че само уместните аспекти на проблема да бъдат разгледани. Все пак всички знаем, че можем да се абстрахираме от различни перспективи. SOA цели да се абстрахира като се концентрира само на бизнес аспектите на проблема. Основния термин представен тука е услуга. В основата си услуга е ИТ представяне на някаква бизнес функционалност. Целта на SOA е да структурира големите системи на базата на абстрахирането от бизнес правилата и функции.

Това дава ясна структура на ИТ системата, която проектираме и разработваме. Въпреки, че вътрешно те са, разбира се, технически системи, външните им интерфейси трябва да бъдат проектирани по такъв начин, че бизнес хората да ги разберат. Външно ти не би трябвало да виждаш техническите детайли. Последствието от този подход е, че на това ниво от абстрахиране специфичните детайли за платформата нямат значение. Поради това платформите могат да бъдат разнородни.

Въпреки тази широка дефиниция, не е много ясно какво услуга е или може да бъде. Има много различни мнения за това какви атрибути услугата трябва, може и би трябвало да има. Все пак можеш да смяташ услугата за ИТ изображение на самостоятелна бизнес функционалност, като „създаване на акаунт”, „ извеждане на договора на клиента”, „трансфер на пари”, „включване на радио”, „изчисляване на най-добрия маршрут за моята кола” и т.н.

3.2. Високо взаимодействие

С разнородните системи, първата цел е да можеш да свържеш тези системи лесно. Това обикновено се нарича „високо взаимодействие”. Високото взаимодействие не е нова идея. Преди SOA, имахме концепцията на корпоративното интегриране на приложение (EAI) и относителното взаимодействие, SOA не е нещо ново.

Въпреки това, за SOA, високото взаимодействие е началото, не краят. Това е основата, с която започваме осъществяването на бизнес функционалностите (услуги), които са разпространени върху множество разпространени системи.

Слаба свързаност

Ние живеем в свят управлявам от хората от маркетинга. Не винаги имаме време да анализираме, проектираме и осъществяваме внимателно. Трябва възможно най-бързо да продаваме и заради това гъвкавостта често е преди качеството. Това води до няколко проблема.

Нека примерно по едно и също време интегрираме една или повече системи и осъществяваме бизнес процеси като ги разпространяваме по различни системи. Общо взето информацията се движи по такъв начин, че процесите да вървят успешно на всички засегнати системи: ние създаваме клиент на всички наши системи, трансферираме пари от една система на друга или задействаме поръчка на клиент като му изпращаме продукта и сметката за него.

Множество от системи са замесени, но няма време за спиране на системата. Това не звучи добре. С цялата си сложност един малък проблем може да спре целия бизнес. Това трябва да бъде избегнато. За това имаме нужда от допустимост на грешка. Така че това са нашите цели:

Гъвкавост
Стабилност при голяма натовареност
Допустимост на грешка

В основата на тези цели стои слабата свързаност. Слабата свързаност е концепцията на минималните зависимости. Когато зависимостите са минимални, модификациите имат минимален ефект и системите все още работят, когато част от тях са счупени или изключени. Минималните зависимости допринасят за допустимост на грешка и гъвкавостта, и това е точно от което се нуждаем.

В допълнение, слабо свързаността води до стабилност при голямо натоварване. Големите системи имат навика да стигат предела си. За това е важно да се избегнат безизходиците; иначе разрастването може да стане много скъпо. Избягването на безизходиците е много важно както от техническа така и от организационна гледна точка. Всички големи системи работят само ако общоприетия бизнес може да бъде направен в децентрализиран начин, ако е възможно. Един от начините за въвеждането на слабата свързаност е да се избегна въвеждането на централизация не повече от нужното (за съжаление, ще имаш нужда от централизация, за да установиш SOA, защото ще имаш нужда от обща база).

В практиката има много начини да реализираш слабата свързаност. И запомни SOA е само образец, не рецепта. Колко голяма ще е слабо свързаността зависи само от теб.

4. Съставките на SOA

Вече прочете, че основните технически понятия на SOA са услугите, взаимодействието и слабо свързаността, може би си заключил, че всичко което трябва да направиш, за да задействаш SOA е да въведеш услугите, взаимодействието и слабо свързаността.

Но както определих по-рано, не можеш да купиш SOA. Важното е да въведеш тези понятия по подходящия начин. И той е да намериш правилната степен на централизация, да нагласиш съответните процеси, и да си напишеш домашното както се казва. Липсата на тези „съставки” е най-честия проблем в реалните SOA проекти. За да

създадеш SOA успешно, трябва да се грижиш за инфраструктурата, архитектурата и процесите (включително мета-процеса управление).

4.1. Инфраструктура

Инфраструктурата е техническата част от SOA, която разрешава високото взаимодействие. Инфраструктурата от пейзажа на SOA се нарича enterprise service bus (ESB). Термина е взет от корпоративното интегриране на приложение, където е било наречено EAI bus или просто enterprise bus.

Характерна особеност на ESB е това, че позволява услугите да се свързват между разнородни системи. Неговите отговорности включват преобразуване на информация, (интелигентно) рутиране, справяне със сигурността и надежността, управление на услугите, мониторинг и записване на логове.

4.2. Архитектура

Архитектурата е необходима да ограничи всичките възможности на SOA по такъв начин, че да доведе до работеща, поддържаща се система. SOA понятията, SOA инструментите и SOA стандартите оставят място за специфични решения, които трябва да направиш, ако искаш да избегнеш неразборията. Трябва да класифицираш различни видове услуги, трябва да прецениш колко голяма ще е слабо свързаността, трябва да ограничиш информационния модел на интерфейса на услугата, трябва да определиш курса на поведение, правилата и модела, трябва да изясниш ролите и отговорностите, трябва да решиш коя технология на инфраструктурата ще използваш, трябва да решиш кои стандарти ще използваш и т.н. Правейки тези решения ти ще можеш да изградиш твоя системна архитектура.

4.3 Процеси

Едно нещо прави големите системи сложни и това е, че много и различни хора и екипи са замесени в тях. Като последствие от това трябва да се

извърви дълъг път от първоначалната идея или изискване докато се намери решение, което да върви по време на продукция.

Понеже обикновено не един или няколко хора контролират всичко, ти трябва да нагласиш подходящите процеси (тези процеси могат да бъдат изрично определени или просто се подразбират). Тези процеси включват:

Business process modeling (BPM)

BMP е задача, която разбива бизнес процесите на по-малки дейности и задачи, които са услуги в този случай.

Service lifecycles

За да дефиниране service lifecycle трябва да дефинираме различните стъпки, които услуга трябва да направи, за да стане реалност.

Model-driven software development (MDSD)

MDSD е процеса на генериране на код, за да се справи с услугата.

4.4 Управление (Governance)

Мета-процесите от всички процеси и SOA стратегията като цяло е управлението. Трябва да създадеш правилния процес, за да установиш SOA в твоята организация. Това включва да намериш правилните хора, които могат да комбинират всичките различни SOA съставки, за да създадат резултат, който работи и е подходящ. Обикновено има централен екип (понякога се нарича SOA компетентен център), който се занимава с инфраструктурата, архитектурата и процесите. Този екип също отговаря за установяване на общото разбирателство. Това изисква управление на персонала, защото когато става дума за време и ресурси, решителността е изискване, за да се справиш с организационните сблъсъци на SOA. Разбирателството, управлението и управлението на персонала са важни фактори за успеха на SOA.

SOA Терминология

Нека поговорим малко за терминологията. Както обикновено с развиващи се понятия, различни термини често се използват за едно и също нещо, и даден термин може да има различни значения. Ето и някой от термините, които ще представя.

За начало някой термини за ролите, които системите имат когато комуникират чрез услуги:

Provider – provider е система, която осъществява услуга, така че другите системи мога да я извикат
Consumer – consumer е система, която извиква услуга

Requestor е често използвана дума за consumer (особено в света на Web Services). Този е по-технически термин базиран на факта, услуга консуматор изпраща искане на provider, за да извика услуга.

Също така може да чуете обичайните термини за роли в разпространените системи – клиент и сървър. Действително provider е сървъра на услугите, докато consumer е клиента, който ги използва. Въпреки това сървър и клиент се използват на много различни места, за това аз предпочитам и препоръчвам да се използват consumer и provider вместо тях.

Ако искате да използвате по-общ термин за consumer и provider като цяло, може да използвате термина участници (participant). Участник на услуга в система е както provider на услуга, така и consumer на услуга. Отново има алтернативи за по-общо използване. Например в JAVA света термина service agent се е развил.