وزارت علوم، تحقیقات و فناوری

مجتمع آموزش عالی سراوان

درس یکپارچه سازی کاربردهای سازمانی

(خلاصه مباحث)

استاد مربوطه : فریدون محمدی

دانشکده فنی و مهندسی – گروه فناوری اطلاعات

معماری سازمانی :

معماری سازمانی (Enterprise Architecture)رویکردی جامع و یکپارچه است که جنبه‌ها و عناصر مختلف سازمان (سیستم) را با نگاه مهندسی تفکیک و تحلیل می‌نماید و شامل مجموعه مستندات، مدل‌ها، استانداردها و اقدامات اجرایی برای تحول از وضعیت موجود به وضعیت مطلوب با محوریت فناوری اطلاعات است که در قالب یک طرح مشخص اجرا شده و سپس به صورت مداوم توسعه و بروزرسانی می‌شود.

معماری سازمانی از معماری سیستم‌های اطلاعاتی و معماری اطلاعات نشأت گرفته و چارچوبی برای تبیین، هماهنگ‌سازی و همسوسازی کلیه فعالیت‌ها و عناصر سازمان در جهت نیل به اهداف راهبردی سازمان است و تمام جنبه‌های سازمان نظیر کاربران، موقعیت جغرافیایی سیستم‌ها، نحوه توزیع آنها، فرایندهای حرفه، انگیزه کارها، راهبردها، مأموریت‌های سازمان و نظایر آن‌ها را در نظر می‌گیرد. در واقع معماری سازمانی نوعی مهندسی مجدد را در کل سازمان، از منظر سیستم‌های اطلاعاتی در جهت بهبود فرایندهای کاری سازمان از طریق به‌کارگیری فناوری اطلاعات شکل می‌دهد، در حالیکه تمرکز معماری اطلاعات بر اطلاعات جاری در سازمان است. معماری سازمانی ناشی از گسترش بکارگیری فناوری اطلاعات در سازمان‌ها و افزایش تعاملات درون و برون سازمانی، و همچنین شتاب روزافزون تغییرات در سازمان‌ها است .

معماری سازمانی مجموعه‌ای از نقشه‌های فنی، نمودارها، و مستندات است که به منظور تعریف مأموریت‌ها، اطلاعات لازم جهت انجام مأموریت‌ها، فناوری‌های مورد نیاز و فرفرایندهای انتقالی لازم جهت انجام آن‌ها استفاده می‌شوند و شامل معماری وضع موجود، معماری وضع مطلوب، و یک طرح انتقالی است .فرایند معماری نیز شامل سه فاز اصلی برنامه‌ریزی راهبردی فناوری اطلاعات، برنامه‌ریزی معماری سازمانی، و اجرای معماری سازمانی است .

تاریخچه معماری سازمانی در ایران

اولین اقدامات مرتبط با معماری سازمانی در سال ۱۳۷۷ در وزارت جهاد کشاورزی آغاز شد، شروع رسمی فعالیت‌های مرتبط با معماری سازمانی در مقیاس کشور در سال ۱۳۸۲ و با تأسیس کمیته فنی معماری اطلاعات ایران – تحت پوشش دبیرخانه شورای عالی اطلاع‌رسانی- رقم خورد و به دنبال آن کسب و ترویج دانش بومی معماری سازمانی در اولویت قرار گرفت. از اوایل دهه ۸۰ در کنار تحقیقات علمی و فنی، اجرای پروژه‌های کاربردی برای دستگاه‌های دولتی و شرکت‌های خصوصی آغاز شد و شرکت‌های مشاور در حوزه تدوین و اجرای طرح‌های معماری سازمانی فعال شدند که مشتریان متنوعی از وزراتخانه‌ها و سازمانهای دولتی تا شرکت‌های خصوصی و تولیدی داشته‌اند. دستاورد نزدیک به دو دهه تجارب معماری سازمانی در ایران، اجرای نزدیک به ۱۰۰ پروژه بزرگ معماری سازمانی و طرح جامع فاوا بوده‌است. در سال ۸۶ با گسترش فعالیت‌های پژوهشی و کاربردی حوزه معماری سازمانی، دانشگاه شهید بهشتی موفق شد به عنوان اولین دانشگاه ملی مجوز پذیرش دانشجوی کارشناسی ارشد در رشته معماری سازمانی را کسب نموده و از سال ۸۷ تا اکنون در این مقطع هرساله دانشجویان کارشناسی ارشد مهندسی فناوری اطلاعات را با تمرکز بر معماری سازمانی پذیرش می‌کند. با گسترش فعالیت‌های پژوهشی و نیز اجرای پروژه‌های موفق متعدد، سرانجام در سال ۹۰ با حمایت سازمان فناوری اطلاعات ایران، اولین آزمایشگاه معماری سازمانی سرویس‌گرا با هدف ترویج، آموزش و استانداردسازی متدهای معماری سازمانی سرویس‌گرا در دانشگاه شهید بهشتی تأسیس شد. سرانجام در سال ۱۳۹۴ و با تصویب طرح تدوین چارچوب و برنامه ملی معماری سازمانی ایران، مجموعه فعالیت‌های علمی، فنی و اجرایی معماری سازمانی در کشور منسجم تر از گذشته تحت هدایت کمیسیون توسعه دولت الکترونیکی قرار گرفت. در همین راستا نهایتاً نسخه اول چارچوب معماری سازمانی در سال ۹۵ منتشر شد. در سال ۱۳۹۶ اولین همایش ملی پیشرفت‌های معماری سازمانی در ایران به میزبانی دانشگاه شهید بهشتی در تهران برگزار شد.

مزایای معماری سازمانی

مزایای معماری سازمانی (فناوری اطلاعات) برای سازمان‌های متفاوت بر اساس تحقیقات صورت گرفته در چند دسته متفاوت قابل تقسیم‌بندی است.

مزایای «معماری سازمانی» در مقیاس کشوری (دولت الکترونیک) :

- نگاه جامع کشوری به راهکارها، استانداردها و فناوری‌ها
- هماهنگ سازی معماری سازمان‌ها با معماری دولت الکترونیک* تحقق تعامل پذیری بین سازمانی (e-GIF)
- کاهش هزینه‌ها و جلوگیری از دوباره کاری در صنعت فاوا
- تحقق ارائه خدمات یکپارچه از پنجره واحد به شهروندان
- زمینه‌سازی برای پایش متمرکز خدمات الکترونیک دستگاه‌ها و بهبود مستمر شاخص‌ها

مزایای معماری سازمانی (فناوری اطلاعات) برای دستگاه‌های دولتی و شرکت‌های خصوصی :

- همراستا سازی اهداف و فعالیت‌های فاوا دستگاه با مأموریت و اهداف سازمانی (کارامدی سازمان)
  یکپارچگی خدمات، سیستم‌ها و بانک‌های اطلاعات سازمانی (یکپارچگی)
- نگاه جامع در انتخاب راهکارهای فاوا مناسب و متناسب(ERP, BPMS, Portal, ITIL, ISMS,...)
- مدیریت پذیری و کاهش هزینه‌های فناوری اطلاعات
- ارائه خدمات الکترونیکی کارامد و یکپارچه سازمانی به شهروندان

فرایند معماری سازمانی

هدف از فرایند معماری سازمانی ایجاد و اجرای معماری و ارائه خروجی‌های معماری در سازمان است. این فرایند در کنار دیگر فرایندهای اصلی سازمان قرار گرفته و به صورت پیوسته اجرا می‌شود. بطور کلی، این فرایند شامل سه مرحله اصلی است که عبارتند از :

۱. برنامه‌ریزی راهبردی فناوری اطلاعات،
۲. برنامه‌ریزی معماری سازمانی،
۳. اجرای معماری سازمانی.

چارچوب‌های معماری سازمانی

نیاز به ارائه تفکرهای سازماندهی شده و ساختارهای منطقی طبقه‌بندی اطلاعات پیچیده و توصیف و مفهوم کارهای معمار پیشگامان معماری را بر آن داشت تا روش‌ها و الگوهای مختلفی را ابداع و به جامعه معماری این امکان را بدهند تا اطلاعات کار معماری را تبدیل به یک چارچوب نموده و به روش علمی و مدون نموده و بدین طریق هنر و علم خود را در اختیار دیگران بگذارد. چارچوب معماری Zachmanبه عنوان اولین چارچوب معماری سازمانی، جان زکمن در سال ۱۹۸۷ برای صنعت و تجارت ارائه کرده‌است. وی در ابتدا این کار را با ارائه یک الگوی جامع در زمینه معماری اطلاعات شروع نمود و چون معتقد به تحلیل سازمان با استفاده از چارچوب معماری بود جنبه‌های مختلف طراحی یک سیستم از نظر محتوی، مفهوم، منطق، فیزیک و توصیف‌های دقیق غیر محتوایی را به صورت یک سری سؤال از دیدگاه داده، وظیفه، شبکه، افراد، زمان و انگیزه در یک جدول که تکمیل آن توصیف معماری انجام شده و پاسخ چه چیز، چطور، کجا، چه کسی، کی و چرا ی آینده سازمانی پویا می‌باشد .

چارچوب‌ها و مدل‌های مرجع معماری سازمانی عمومی (همه منظوره)

این چارچوب‌ها برای دولت یا صنعت خاصی تولید نشده‌اند و به صورت عمومی برای سازمان‌ها و مقاصد مختلفی قابل استفاده هستند. البته به دلیل عمومی بودن نیاز به سفارشی سازی در هر صنعت یا کاربرد خاص دارند، معروف‌ترین چارچوب‌های معماری سازمانی عمومی به قرار زیر است :

- The Zachman Framework
- (The Open Group Architecture Framework)TOGAF
- (Gartner's Enterprise Architecture Framework)GEAF
- (The Oracle Enterprise Architecture Framework)OEAF
- ...

چارچوب زکمن :

چارچوب زکمن (به انگلیسیZachman Framework) : یک هستان‌شناسی سازمانی و یک ساختار پایه ای برای معماری سازمانی است که روشی باقاعده و ساخت یافته برای نمایش و تعریف یک سازمان ارائه می‌کند. هستان‌شناسی یک طرح طبقه‌بندی دو بعدی است که تقاطع بین دو طبقه‌بندی تاریخی را نشان می‌دهد. طبقه اول کلمات پرسشی اصلی: "چه چیز" ,"چطور", "کِی", "کِه", "کجاً و "چراً هستند. طبقه دوم برگرفته از مفهوم فلسفی جسمیت بخشی یعنی تحول از یک ایده انتزاعی به یک نمونه واقعی است. مراحل انتقالی برای جسمیت بخشی در چارچوب زکمن عبارتند از: شناسایی، تعریف، ارائه، تعیین مشخصات، پیکربندی و نمونه سازی است.

چارچوب زکمن یک متدلوژی نیست، چرا که در آن هیچ متد یا پروسه ای برای جمع‌آوری، مدیریت و استفاده از اطلاعاتی که تشریح می‌کند، بیان نمی‌شود، بلکه یک هستان‌شناسی است که به موجب آن یک طرح کلی برای سازماندهی آثار معماری (به عبارت دیگر اسناد طراحی و مدارک و مشخصات و مدل‌ها) جهت تعیین اینکه آثار معماری برای چه کسی است (به عنوان مثال مالک کسب و کار یا سازنده) و چه مسائل خاصی وجود دارند (به عنوان مثال داده‌ها و عملکرد) استفاده می‌شود .

چارچوب از روی نام خالق آن جان زکمن که برای اولین بار مفاهیم آنرا در سال ۱۹۸۰ در آی بی ام توسعه داد، نامگذاری شده‌است و از آن تاریخ تاکنون چندین بار به روز رسانی شده‌است .

مفهوم

ایده اصلی در پشت چارچوب زکمن این است که یک چیز یا مورد پیچیده را می‌توان برای مقاصد مختلف با استفاده از توصیف‌های مختلف (به عنوان مثال متنی، گرافیکی) به راه‌های مختلف تعریف نمود. چارچوب زکمن سی و شش طبقه لازم برای توصیف کامل هر چیزی ارائه می‌کند، به خصوص چیزهای پیچیده مانند کالاهای تولید شده)به عنوان مثال لوازم خانگی) سازه‌های ساخته شده (مانند ساختمانها) و سازمانها (به عنوان مثال سازمان و همه اهداف، اشخاص و فناوری‌های آن). این چارچوب شش تحول مختلف را برای یک ایده انتزاعی (نه افزایش جزئیات، تنها تحول) را از شش دیدگاه مختلف فراهم می‌کند .

این ویژگی اجازه می‌دهد تا افراد مختلف به یک چیز یکسان از دیدگاه‌های مختلف نگاه کنند. این کار باعث ایجاد یک نگرش همگانی از محیط می‌گردد، قابلیت مهمی که در شکل نشان داده شده‌است .

نمایش سطرها

هر سطر نشان دهنده یک دید کلی از راه حل از یک دیدگاه خاص است. یک سطر یا دیدگاه بالاتر لزوماً نباید نسبت به دیدگاه پایین‌تر درک جامعتری از کل مسئله داشته باشد. هر سطر نشان دهنده یک دیدگاه متمایز و منحصر به فرد است، با این حال، خروجی‌ها از هر دیدگاه باید جزئیات کافی برای تعریف راه حل در سطح خود دیدگاه را فراهم کنند و همچنین باید برای استفاده در دیدگاه سطر پایین‌تر به صورت صریح ترجمه گردند .

هر دیدگاه باید نیازها و محدودیت‌ها و الزامات دیگر دیدگاه‌ها را مد نظر قرار دهد. محدودیت‌های دیدگاه‌ها به صورت افزودنی هستند. برای مثال محدودیت‌های سطر بالاتر سطر پایین‌تر را تحت تأثیر قرار می‌دهد . محدودیت‌های سطر پایین‌تر می‌تواند، اما نه لزوماً، بر سطرهای بالاتر تأثیر بگذارد. درک الزامات و محدودیتها مستلزم تبادل دانش و داشتن درک متقابل از دیدگاهی به دیدگاه دیگر است. چارچوب از مسیر عمودی برای ارتباطات بین دیدگاه‌ها استفاده می‌کند .

نسخه فعلی (۳) چارچوب زکمن سطرها را مطابق زیرطبقه بندی می‌کند :

- **دیدگاه اجرایی (محدوده محتویات)** - اولین طرح معماری یک" نمودارحبابی " یانمودار ون می‌کشد که به‌طور ناخالص اندازه، شکل، روابط نسبی و هدف اساسی ساختار نهایی را نشان می‌دهد. این دیدگاه یک خلاصه اجرایی را به برنامه‌ریز یا سرمایه‌گذار نمایش می‌دهد که می‌خواهد تنها یک دیدگاه کلی از سیستم داشته باشد یا می‌خواهد محدوده سیستم، هزینه‌های آن و نحوه تعاملش با محیط عمومی که در آن فعالیت می‌کند را برآورد کند.
- **دیدگاه مدیریت کسب و کار (مفاهیم کسب و کار)** - طبقه بعدی نقشه‌های معمار هستند که ساختمان نهایی را از دیدگاه صاحب سیستم به تصویر می‌کشند، یعنی کسی که می‌خواهد با این معماری زندگی

کند واز آن در روالهای روزانه کسب و کار استفاده کند. این دیدگاه‌ها مدلهای سازمانی (کسب و کار) را نمایشمی دهند که طرح‌های کسب و کار و موجودیت‌های کسب و کار و فرایندها و نحوه ارتباط آنها را نشان می‌دهد.

- **دیدگاه معمار (منطق سیستم)** - برنامه‌های معمار که ترجمه‌هایی از نقشه‌های معمار به جزئیات مورد نیاز از دیدگاه طراح هستند. آنها مدل سیستمی نمایش می‌دهند که توسط یک تحلیلگر سیستم طراحی شده که باید عناصر داده‌ای، جریان‌های منطقی فرایند و توابعی که موجودیت‌های کسب و کار و فرایندها را معرفی می‌کنند را تعیین کند.
- **مهندس دیدگاه (فیزیک فناوری)** - پیمانکار محبور است برنامه‌های معمار را برای آماده‌سازی دیدگاه سازنده دوباره ترسیم کند تا بتواندجزئیات کافی به منظور درک کامل محدودیت‌های ابزار، فناوری و مواد را ارائه کند .برنامه‌های سازنده مدل‌های فناوری را نمایش می‌دهند که باید مدل سیستم‌های اطلاعات را با جزئیات زبان‌های برنامه‌نویسی، دستگاه‌های ورودی/خروجی (I/O) یا سایر فناوریهای پشتیبانی شده منطبق سازند.
- **دیدگاه تکنسین (قطعات ابزار)** - پیمانکاران فرعی کار از روی برنامه‌های کارگاهی انجام می‌دهند که جزئیات قطعات یا زیر مجموعه‌ها را بیان می‌نمایند. این برنامه‌ها مشخصات دقیقی نمایش می‌دهند که به برنامه نویسان اجازه می‌دهد ماژول‌های مجزایی را بدون نگران بودن راجع به مضمون یا ساختار کلی سیستم کدنویسی کنند، یا به جای آن، آنها می‌توانند نیازمنده‌های دقیقی را برای استفاده از برنامه‌های تجاری آماده (COTS) یا برنامه‌های دولتی آماده (GOTS) یا اجزای نرم‌افزاری سیستم‌های ماژولار که از پیش موجود بوده‌اند به جای کدنویسی و ساخت این ماژولها ارائه نمایند.

دیدگاه سازمان (نمونه‌های عملیاتی)

تمرکز بر روی ستونها

به‌طور خلاصه، هر دیدگاه تمرکز را متوجه یک پرسش اساسی می‌کند و سپس به این سؤال از همان دیدگاه به نحوی پاسخ می‌دهد که ارائه‌های توصیفی (به‌طور مثال مدل‌ها) متفاوتی ایجاد شوند که از دیدگاه‌های بالاتر به دیدگاه‌های پایین‌تر ترجمه بشوند. مدل اساسی برای تمرکز (یا ایجاد انتزاع) ثابت می‌ماند. مدل پایه هر ستون به‌طور منحصر به فرد تعریف شده و در عین حال هم به‌طور عرضی و هم از بالا به پایین ماتریس با سایر ستونها در ارتباط است .علاوه بر این، شش طبقه از قطعات معماری سازمانی و پرسش‌های مربوطه که آنها پاسخ می‌دهند، ستونهای چارچوب زکمن را تشکیل می‌دهند که به شرح زیر هستند :

۱) محموعه موجودی ها— چه
۲) جریان‌های فرایند — چگونه
۳) شبکه‌های توزیع — کجا
۴) تخصیص مسئولیتها — کِه
۵) چرخه‌های زمانی — کِی
۶) نیت‌های انگیزشی — چرا

به اعتقاد زکمن، تنها عاملی که چارچوبش را منحصر به فرد می‌سازد، این است که هر عنصر در هر کدام از دو محور ماتریس به صراحت قابل تشخیص از تمام عناصر دیگر در این محور است. آنچه هر سلول ماتریس ارائه می‌دهد نه صرفاً افزایش جزئیات نسبت به سطوح پیشین، بلکه در واقع ارائه ای متفاوت - متفاوت در مضمون، معنی، انگیزه و استفاده - می‌باشد. به این دلیل که هر یک از این عناصر در هر محور به صراحت متفاوت از دیگران است، امکان تعیین آنچه که متعلق به هر سلول است به صورت دقیق وجود دارد .

مدل‌های سلول‌ها

چارچوب زکمن به‌طور معمول به صورت یک ماتریس ۶ * ۶ با پرسشهای ارتباطی به عنوان ستون و سیر تحولات تجسم سازی به عنوان سطر به تصویر کشیده شده‌است. طبقه بندیهای چارچوب توسط سلول‌ها بخش‌بندی شده‌است که تقاطع بین پرسش‌ها و تحولات هستند .

توضیحات سلولها مستقیماً برگرفته از نسخه ۳،۰ چارچوب زکمن هستند :

دیدگاه اجرایی

۱) (چه) شناسایی موجودی

۲) (چگونه) شناسایی فرایند

۳) (کجا) شناسایی توزیع

۴) (کِه) شناسایی مسئولیت

۵) (کِی) شناسایی زمان‌بندی

۶) (چرا) شناسایی انگیزه

دیدگاه مدیریت کسب و کار

۱) (چه) تعریف موجودی

۲) (چگونه) تعریف فرایند

۳) (کجا) تعریف توزیع

۴) (کِه) تعریف مسئولیت

۵) (کِی) تعریف زمان‌بندی

۶) (چرا) تعریف انگیزه

دیدگاه معمار

۱) (چه) نمایش موجودی

۲) (چگونه) نمایش فرایند

۳) (کجا) نمایش توزیع

۴) (کِه) نمایش مسئولیت

۵) (کِی) نمایش زمان‌بندی

۶) (چرا) نمایش انگیزه

دیدگاه مهندس

۱) (چه) مشخصات موجودی

۲) (چگونه) مشخصات فرایند

۳) (کجا) مشخصات توزیع

۴) (کِه) مشخصات مسئولیت

۵) (کِی) مشخصات زمان‌بندی

۶) (چرا) مشخصات انگیزه

دیدگاه تکنسین

۱) (چه) پیکربندی موجودی

۲) (چگونه) پیکربند فرایند

۳) (کجا) پیکربندی توزیع

۴) (کِه) پیکربندی مسئولیت

۵) (کِی) پیکربندی زمان‌بندی

۶) (چرا) پیکربندی انگیزه

دیدگاه سازمان

۱) (چه) نمونه سازی موجودی

۲) (چگونه) نمونه سازی فرایند

۳) (کجا) نمونه سازی توزیع

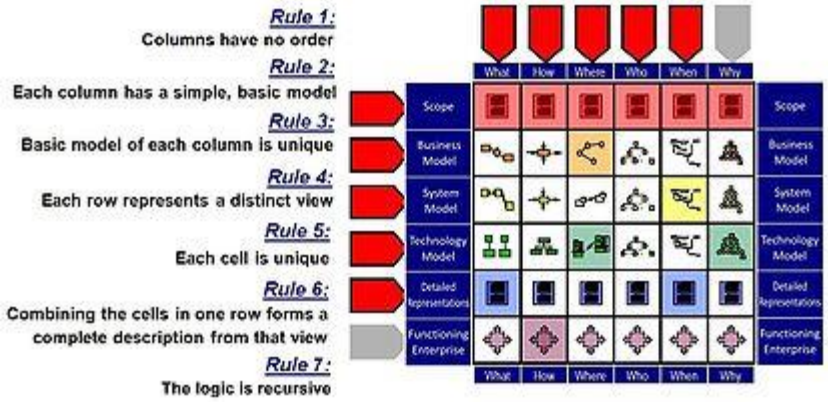
۴) (کِه) نمونه سازی مسئولیت

۵) (کِی) نمونه سازی زمان

۶) (چرا) نمونه سازی انگیزه

از زمان توسعه محصول (به عنوان مثال مصنوع معماری) در ازای هر سلول یا راه حل مشکل که در هرسلول مندرج شده‌است، پاسخی به یک سؤال از یک دیدگاه وجود داردکه عموماً به شکل مدل‌ها یا توصیفهای سطح بالاتر یا پاسخ‌های سطحی می‌باشند. طراحی‌ها یا مدلهای استخراج شده که ازپاسخ پشتیبانی می‌کنند همان توضیحات با جزییات داخل سلولها هستند. تجزیه (یعنی شکستن به سطح بیشتری از جزئیات) در هر سلول اتفاق می‌افتد. اگر یک سلول صریح (تعریف شده (نباشد به صورت غیر صریح (تعریف نشده) است. اگر یک سلول غیر صریح باشد، خطر ایجاد مفروضات در مورد این سلول وجود دارد. اگر مفروضات معتبر باشند، در زمان و پول صرفه جویی شده‌است. اگر مفروضات نامعتبر باشند آنگاه به احتمال زیاد هزینه‌ها و زمان‌بندی پیاده‌سازی افزایش می‌یابند .

مجموعه ای از قوانین چارچوب



مثال از قوانین چارچوب زکمن.

چارچوب دارای مجموعه ای از قوانین است :

- قانون ۱ ستون‌ها هیچ ترتیبی ندارند :ستون‌ها قابل تعویض هستند اما نمی‌توان آنها راکاهش می‌یابد یا ستون جدیدی ایجاد کرد.
- قانون ۲ هر ستون یک مدل ذاتی ساده دارد: هر ستون می‌تواندفرا مدل خاص خودش را داشته باشد.
- قانون ۳ مدل پایه از هر ستون باید منحصر به فردباشد: مدل پایه از هر ستون، ارتباط اشیاء و ساختار آن منحصر به فرد است. هر شی ء رابطه ای دارای وابستگی داخلی است اما هدف نمایش داده شده منحصر به فرد است.
- قانون ۴ هر سطر یک دیدگاه مجزا و منحصر به فرداست: هر سطر یک دیدگاه از یک گروه کسب و کار خاص را شرح می‌دهد و در این خصوص منحصر به فرد است. همه سطرها معمولاً در بسیاری از سازمان‌های سلسله مراتبی ارائه می‌شوند.

- قانون ۵ هر سلول منحصر به فرد است :ترکیبی از ۲ و ۳ و ۴ باید سلول‌های منحصر به فرد تولید کند که در آن هر سلول نشان دهنده یک مورد خاص است. مثال A2 :نشان دهنده خروجی کسب و کار است به گونه ای که آنچه آنها نمایش می‌دهند در نهایت ساخته می‌شود.
- قانون ۶ ادغام یا تجمیع تمام مدل‌های سلول در یک سطر یک دیدگاه کامل از این سطر را نمایش می‌دهد :همین دلیلی است برای اضافه نکردن سطرها و ستون‌ها، تغییر دادن نامها ممکن است باعث تغییر در ساختار منطقی اساسی چارچوب بشود.
- قانون ۷ منطق بازگشتی است :منطق، رابطه ای بین دو نمونه از یک موجودیت می‌باشد.

چارچوب از این رو ذاتی است که همان‌طور که می‌توان آن را برای طبقه‌بندی نمایش‌های هر شی ءفیزیکی به کار برد می‌توان برای اشیاء مفهومی مانند سازمان هم به کار برد. همچنین چارچوب بازگشتی است زیرا که می‌تواند برای تجزیه و تحلیل ترکیب معماری خودش هم مورد استفاده قرار بگیرد. اگر چه چارچوب رابطه هر ستون با ستون دیگر را نمایش می‌دهد، اما هنوز یک نمایش ساختاری از سازمان است نه یک نمایش جریانی .

انعطاف‌پذیری در سطح جزئیات

یکی از نقاط قوت چارچوب زکمن این است که به صراحت مجموعه ای جامع از دیدگاه‌ها را نشان می‌دهد که می‌تواند توسط معماری سازمان استفاده شود. برخی احساس می‌کنند که پیروی کامل از این مدل می‌تواند منجر به تأکید بیش از حد بر روی اسناد و مدارک شود، چرا که برای هر یک از سی سلول چارچوب، مصنوعاتی مورد نیاز است. زکمن اما نشان می‌دهد که تنها حقایق مورد نیاز برای حل مسأله ٔ تحت تجزیه و تحلیل، نیاز به انتشار دارند .

جان زکمن در اسناد و سخنرانی‌ها و سمینارهایش به وضوح بیان نمود که، به عنوان چارچوب، چه درعمق و چه در وسعت جزئیاتی که مورد نیاز هر یک از سلول‌های ماتریس است، بر اساس اهمیت‌ها و اولویت‌های هرسازمان انعطاف‌پذیری وجود دارد. یک خودروساز که اهداف کسب و کارش ممکن است حول محور ضرورت وجود انبار و تمرکز فرایند محور باشد، ممکن است تمرکز بر روی مستندسازی ستون‌های **چه** و **چگونه** را سودمند بیابد. در مقابل، یک آژانس مسافرتی که کسب و کارش بیشتر در رابطه با مردم و رویداد-زمان‌بندی است می‌تواند تمرکز بر روی مستندسازی ستون‌های **کِی** و **که** و **کجا** را مفید بیابد. اگرچه هیچ راه فراری از اهمیت ستون **چرا** وجود ندارد که پیشرانه‌های کسب و کار را برای تمام ستون‌های دیگر فراهم می‌کند .

کاربردها و نفوذها

از سال‌های دهه ۱۹۹۰ چارچوب زکمن به‌طور گسترده‌ای به عنوان وسیله ای برای تهیه ساختار برای مدلسازی سازمانی به سبک مهندسی اطلاعات استفاده می‌شود. چارچوب زکمن می‌تواند هم در شرکت‌های تجاری و هم در سازمان‌های دولتی استفاده شود. در درون یک سازمان دولتی، چارچوب می‌تواند به کل سازمان در یک سطح انتزاعی اعمال شود یا می‌توان آن را به ادارات مختلف، دفاتر، برنامه‌ها، واحدهای زیر مجموعه و حتی به موجودیت‌های پایه ای عملیاتی مستقل اعمال نمود..

چارچوب فدرال (FEAF)

در سال ۱۹۹۶ قانونی موسوم به کلینگر کوهن در کنگره آمریکا به تصویب رسید که مطابق آن، همه وزارتخانه ها و سازمانهای فدرال آمریکا ملزم شدند معماری فناوری اطلاعات خود را ایجاد کنند. مسئولیت تدوین، اصلاح و اجرای معماری فناوری اطلاعات یکپارچه در هر سازمان بر عهده مدیر ارشد اطلاعاتی آن سازمان قرار گرفت. در

سال ۱۹۹۸ بر اساس همین قانون شورای مدیران ارشد اطلاعاتی موظف شدند که جهت توسعه، پشتیبانی و تسهیل پیاده سازی معماری اطلاعات سیستمهای دولتی، راهکار واحدی را ارائه دهد. چارچوب معماری سازمان فدرال در سپتامبر سال ۱۹۹۹ توسط شورای مدیران ارشد اطلاعاتی دولت ایالات متحده آمریکا تهیه و تنظیم شد.

این چارچوب از ۴ سطح و ۸ مولفه تشکیل شده است، در سطوح بالا طرح ها و راهبردهای کلان مطرح می‌شود درحالیکه هر چه به به سمت سطوح پائین تر حرکت می‌کنیم با طرح ها و مشخصات جزئی تر برخورد می‌کنیم تاسرانجام در سطح ۴ به ماتریس محصولات FEAF برمی‌خوریم که همان چارچوب اولیه زکمن است(این ماتریس شامل سه جنبه داده، فرآیند و مکان است که در پنج دیدگاه برنامه ریز، مالک، صاحب، سازنده و پیمانکار طبقه بندی شده است.)

FEAFبرخلاف زکمن تنها به معرفی چارچوب و محصولات نمی‌پردازد، بلکه روش و چگونگی برپاسازی معماری را نیز مشخص می‌کند. در مستندات FEAF ، از متدولوژی "برنامه ریزی معماری سازمانی " آقای اسپیواک به عنوان راهنمای انجام معماری، نام برده شده است.

همانطور که گفته شد این چارچوب دارای هشت مولفه است که عبارتند از:

پیشرانهای معماری :نمایانگر دو نوع محرک یا عامل تغییر معماری سازمانی هستند:

پیشرانهای حرفه که می‌توانند قوانین جدید، تصمیمات جدید مدیریتی، افزایش ناگهانی بودجهٔ حوزه ها و فشارهای بازار باشند.

پیشرانهای طراحی که شامل نرم افزار، سخت افزار و بستر ارتباطی(شبکه) جدید و کارامدتر هستند.

جهت گیری راهبردی: توسعهٔ معماری مقصد را هدایت کرده و شامل چشم انداز، اصول و اهداف می‌شود.

معماری موجود: معماری سازمانی را آنطور که هست تعریف کرده و شامل دو بخش می‌شود: معماری حرفه و اطلاعات (داده ها، کاربردها و فناوری).

معماری مطلوب: معماری سازمانی را آنطور که باید باشد، تعریف نموده و شامل دو بخش می‌شود: معماری حرفه و معماری اطلاعات. این معماری برآوردی است از قابلیتها و فناوریهای آینده که نتیجه بهبود سیستم ها و فناوری های فعلی در جهت پشتیبانی از تغییر در نیازمندیهای حرفه است.

فرآیندهای انتقالی: مهاجرت از معماری فعلی به مقصد را پشتیبانی می‌کنند . فرآیندهای انتقالی حیاتی برای سازمان فدرال شامل طرح کلان سرمایه گذاری فناوری اطلاعات، برنامهٔ انتقال، مدیریت پیکربندی و کنترل تغییرات می‌گردد.

بخشهای معماری: تلاشهای معماری متمرکز بر حوزه های اصلی حرفه مانند سیستمهای مدیریتی رایج، حوزه های برنامه ریزی مانند بازرگانی و سرمایه یا خریدهای کوچک از طریق تجارت الکترونیکی است. هر بخش، تکه ای از کلیت معماری سازمانی فدرال است و یک سازمان درون سازمان اصلی فرض می‌گردد.

مدل های معماری: مدل های حرفه و اطلاعات که نشان دهنده اجزاء و رفتار سازمان هستند، را توصیف می‌نمایند.

استانداردها : همهٔ استانداردها (که بعضی از آنها ممکن است اجباری باشند)، رهنمودها و بهترین تجربیات را در برمی‌گیرند.

نقاط ضعف چارچوب فدرال:

- اگرچه چارچوب فدرال از استانداردها و طرح های انتقالی به عنوان بخشی از معماری بهره برده ولی مشخص نکرده که این استانداردها چگونه می بایست سازماندهی و بکار برده شود.
- در مورد طرح انتقالی هیچ راهنمائی ارائه نشده است
- اگرچه در مستندات چارچوب فدرال نیاز به مخزن مدل ها مطرح شده ولی توضیح یا پیشنهادی داده نشده است.
- راهنمائی درخوصوص امنیت وجود ندارد.
- چارچوب معماری سازمانی فدرال به عنوان "راهنما" برای سازمانهای فدرال ارائه شده و جنبه "اجباری" ندارد.

نقاط قوت چارچوب فدرال:

- این چارچوب گامی مهم در راستای تعریف عناصر و اجزاء معماری سازمانی است. ب
- رخلاف زکمن که تنها به خود چارچوب محصولات و مدل ها می پردازد در اینجا به اهداف استراتژیک و نیازهای گذار(تحول) نیز پرداخته شده است.
- از متدولوژی برنامه ریزی معماری سازمانی اسپیواک به عنوان روش برپاسازی نام برده شده و لایه های معماری تشریح شده اند.

چارچوب توگف (TOGAF)

چارچوب معماری سازمانی است که در آن می توان برای سازمانهای همگون ، یا دارای سطح بلوغ فرایندی (پروفایل) کمتر یا مساوی ۲ ، اقدام به شناسایی وضع موجود و تدوین وضع مطلوب نمود. این چارچوب دارای یک استاندارد در ANSI با شماره ANSI/IEEE 1471:2000می باشد که می تواند برای سازمان و یا یک نرم افزار و یا یک تیم بکار رود .

TOGAFبر اساس چارچوب TAFIM تولید گردیده است . در حال حاضر آخرین نسخه TOGAF در سال ۲۰۰۹ منتشر شده و شماره آن ۹ می باشد . نسخه ۱۰ آن فعلا در حالت Draft می باشد . امروزه در TOGAF می توان از رهنمودهای ارائه شده برای طراحی معماری امنیتی یک سیستم بهره برد .

TOGAFبر اساس ۴ ستون اصلی (Pillar) تدوین شده است ، با عناوین زیر :

1. BAP :Business Architecture Pillar
2. AAP : Application Architecture Pillar
3. DAP: Data Architecture Pillar
4. TAP : Technical Architecture Pillar

تعریفTopic های: TOGAF

مقایسه چارچوب های معماری سازمانی

	TOGAF	TEAF	DODAF	FEAF
ديدهاي پايه معماري	چهار لايه اصلي كسب و كار، داده، نرم افزارهاي كاربردي و تكنولوژي	ماتريس شبيه زكمن شامل چهار ستون وظيفه اي، اطلاعاتي، ساختاري و زير ساختارها	سه ديدگاه عملياتي، سيستمي و فني	ماتريس بر مبناي چارچوب زكمن شامل ۳ ستون داده، نرم افزارهاي كاربردي و تكنولوژي
مشخصات كامل و مشروح محصولات	توضيح مفيدي موجود نمي باشد	توضيحات كامل بر محصولات . همچنين تشريح محصولات برنامه ريزي و طرح انتقال	توضيحات كامل براي تمام محصولات	توضيح بسيار خلاصه در خصوص محصولات معماري
تبين رابطه معماري سازماني با ديدگاه استراتژيك و اهداف ماموريت	استراتژي هاي كسب و كار به عنوان ورودي فاز ديدگاه معماري در نظر گرفته شده است	به عنوان يك محصول مجزا (نقشه راه) در نظر گرفته شده است	بحث شده و به بعضي محصولات نيز اشاره شده ولي محصول مجزائي براي آنها در نظر گرفته نشده است	بحث شده است
تهيه قوانين معماري	وجود دارد	فقط ليست قوانين وزارت خزانه داري و مسئوليت ادارات آورده شده است.	قوانين به عنوان جزئي از توصيف محصولات معماري ديده شده	براي صفات خود چارچوب معماري فدرال تهيه شده است
محصولات براي تعيين استانداردها	در قالب مدل مرجع تكنولوژي(TRM) وجود دارد	تحت عنوان نمايه استانداردها	وجود دارد (TV-1)	وجود ندارد
بحث در خصوص ملاحضات امنيتي	موضوع امنيت مورد بحث قرار گرفته است	در كنترل Cross-Cutting ديده شده است	به طور خلاصه در انتخاب محصولات	خير
بحث پيرامون استراتژي گذار و تعريف محصولات برنامه گذار	بله. در قالب فاز برنامه ريزي گذار(مهاجرت)	بله. در قالب يك محصول مشخص	مقداري بحث شده	بحث شده ولي نه بطور كامل و مشخص
بحث در خصوص مخزن معماري	راهنمائي هائي وجود دارد	مسئوليتهاي مربوط به آن تعيين شده	بحث شده	نياز به مخزن گفته شده
متدولوژي يا راهنماي اجراي معماري سازماني	داراي متد توسعه معماري (ADM) است	در قالب موضوعاتي چون استراتژي معماري سازماني، نقشها و مسئوليتها و مديريت سرمايه گذاري	٦ گام كلان بدون ورود به جزئيات ارائه شده	به متدولوژي برنامه ريزي معماري سازماني اسپيواك اشاره شده

یکپارچه سازی کاربردهای سازمانی(Enterprise Application Integration=EAI)

امروزه سازمان‌ها به شدت متکی به تکنولوژی‌هایی هستند که هر کدام از آن‌ها دارای ویژگی‌ها و کاربردهای متفاوت و منحصربه‌فرد هستند، همچنین نرم‌افزارهای کاربردی مورد استفاده این سازمان‌ها نیز از این قاعده مستثنی نبوده و هر کدام از آن‌ها مبتنی بر تکنولوژی خاصی هستند. این نرم‌افزارها غالباً به صورت مستقل (مستقل از سایر نرم‌افزارهای موجود در سازمان) و به منظور تأمین نیازهای بخش خاصی از سازمان تولید و توسعه می‌یابند که این امر باعث می‌شود تا هر کدام از این نرم‌افزارها، حیطه‌ی کاربرد محدود به همان بخش از سازمان را داشته باشند و در نتیجه توانایی تعامل با سایر نرم‌افزارهای سازمان را نداشته باشند؛ بنابراین یکپارچه‌سازی نرم‌افزارهایی از این دست در قالب یک مجموعه واحد از فرایندهای کسب‌وکار، به عنوان یک نیاز و اولویت نمود پیدا می‌کند.

با توجه به توضیحات فوق این مفهوم را می‌توان در یک جمله تعریف کرد:

EAI"مجموعه‌ای است از فرایندها، استانداردها، نرم‌افزارها و سخت‌افزارها که در راستای یکپارچه‌سازی دو یا چند سیستم کاربردی سازمان (نرم‌افزار کاربردی) عمل نموده و برای این سیستم‌ها شرایطی را فراهم می‌سازد تا بتوانند در قالب یک سیستم واحد عمل کنند"

از فواید EAI می‌توان به موارد ذیل اشاره کرد:

- به سیستم‌ها و نرم‌افزارها کمک می‌کند تا بتوانند به صورت همزمان و لحظه‌ای به اطلاعات یکدیگر دسترسی داشته باشند.
- با ساده‌سازی فرایندهای کسب‌وکار، راندمان و کارایی سازمان را افزایش می‌دهد.
- یکپارچگی اطلاعات را فراهم می‌سازد.
- توسعه و نگهداری سیستم‌ها را آسان می‌سازد.

- ارتباطات با مشتری را بهبود می‌بخشد.
- ارتباط با زنجیره تأمین (Supply Chain) را بهبود می‌بخشد.
- نرم‌افزارهای کاربردی قدیمی (Legacy) را همچنان فعال و زنده نگه می‌دارد.
- فرایندهای کسب‌وکار را بهبود می‌بخشد و در نتیجه مدت زمان مورد نیاز برای عرضه محصول به بازار (Time-To-Market)را کاهش می‌دهد.
- تا حدودی از تغییرات در سطح فرایندی و یا سازمانی پشتیبانی می‌کند.
- نرم‌افزارهای کاربردی را استانداردسازی می‌نماید.
- تکنولوژی واکنشی (Responsive Technology) را برای نیازهای متغیر کسب‌وکار به خدمت می‌گیرد.
- نرم‌افزارهای کاربردی را مطابق با نیازهای تجاری حال و آینده، تغییر می‌دهد (تغییر شکل سیستم‌های کاربردی).

انواع EAI

یکپارچه‌سازی سیستم‌های کاربردی سازمان می‌تواند در سطوح مختلفی اجرا شود:

۱. سطح داده.(Data Level)
۲. سطح رابط برنامه کاربردی.(Application Interface Level)
۳. سطح متد.(Method Level)
۴. سطح رابط کاربری.(User Interface Level)

EAI در سطح داده

این سطح از یکپارچه‌سازی شامل مجموعه‌ای از تکنیک‌ها، فرایندها و تکنولوژی‌هاست که انتقال داده میان منابع داده را امکان‌پذیر می‌سازند .مزیت اصلی این روش در عدم نیاز آن به ایجاد تغییرات در سطح کد (Source Code)می‌باشد که این امر، کاهش هزینه‌های توسعه‌ی مجدد نرم‌افزارها را به همراه دارد.

EAI در سطح رابط برنامه کاربردی

در این سطح، داده‌ها و فرایندهای کسب‌وکار از طریق رابط‌های نرم‌افزاری (مختص توسعه‌دهندگان) دسترس‌پذیر هستند. هر کدام از این رابط‌ها دارای خصوصیات و توابع معینی هستند. با استفاده از این رابط‌ها ، توسعه‌دهندگان قادر خواهند بود تا بسیاری از نرم‌افزارهای کاربردی را کنار هم آورده و امکان اشتراک‌گذاری منطق تجاری (Business Logic)و اطلاعات را برای این نرم‌افزارها فراهم سازند .

در سطح متد

در این سطح ازEAI ، منطق تجاری نرم‌افزارها به اشتراک گذاشته می‌شود. یک متد(مفهوم برنامه‌نویسی) methodمی‌تواند توسط تعداد زیادی از نرم‌افزارهای کاربردی قابل‌دسترس باشد و همچنین نرم‌افزارها می‌توانند به متدهای یکدیگر دسترسی داشته باشند .

در سطح رابط کاربری

در این روش، معماران و توسعه‌دهندگان می‌توانند از رابط‌های کاربری به عنوان یک نقطه اشتراک جهت یکپارچه‌سازی استفاده نمایند (استفاده از رابط کاربری مشترک برای نرم‌افزارها.)

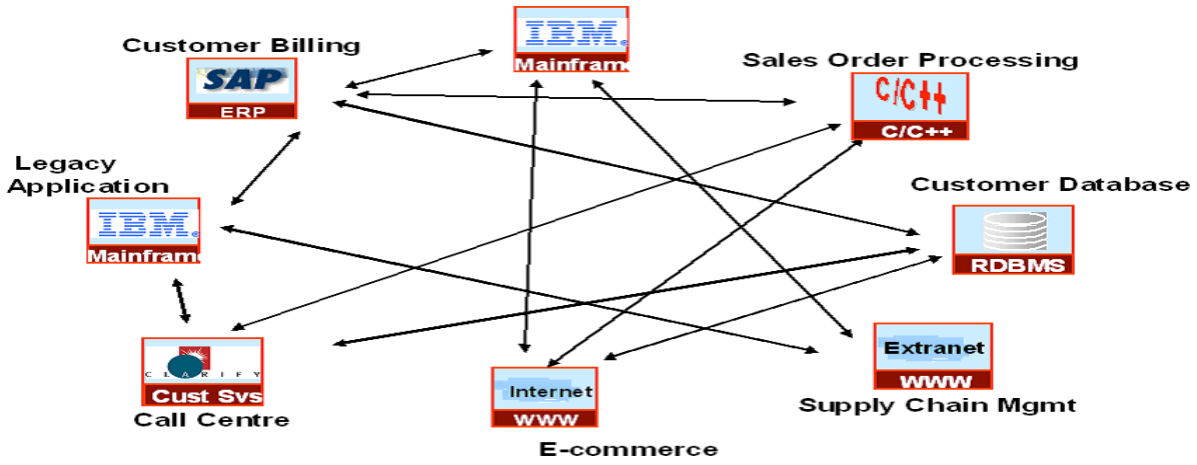
معماری EAI

همچنین EAIمی‌تواند بر اساس طراحی ساختار دسته‌بندی شود:

- توپولوژیPoint To Point
- توپولوژی Hub-Spoke یاBroker
- توپولوژیBus

توپولوژیPoint To Point

Point To Pointبه عنوان یک روش قدیمی یکپارچه‌سازی شناخته می‌شود؛ این توپولوژی به نرم‌افزارها این امکان را می‌دهد تا با استفاده از یک "لوله (Pipe) "به یکدیگر متصل شوند. هر نرم‌افزار کاربردی با استفاده از یک "پیام "یا یک "رویه‌ی فراخوانی (Call Procedure) "با نرم‌افزار مقابل خود ارتباط برقرار می‌کند. به ازای هر جفت از نرم‌افزارهای مرتبط، یک اتصال دهنده (Connector)به منظور برقراری ارتباط، ساخته و پیاده‌سازی می‌شود. این اتصال دهنده وظیفه‌ی تبدیل و یکپارچه‌سازی داده برای هر جفت معین از نرم‌افزارها را بر عهده دارد. البته امکان اتصال بیش از دو نرم‌افزار کاربردی هم وجود دارد، اما این کار پیچیدگی‌های بسیاری را به همراه خواهد داشت (در صورت زیاد بودن تعداد نرم‌افزارها).



شکل ۳ - یکپارچه‌سازی با معماری Point To Point

توپولوژی Hub-Spoke یا Broker

توپولوژی یکپارچه‌سازی Hub-Spoke از یک "واسط متمرکز (Hub) "و تعدادی تطبیق‌دهنده یا آداپتور (Spoke)تشکیل شده است. در واقع Spoke اتصال دهنده ایست که نرم‌افزار کاربردی را به Hub متصل می‌کند. این تطبیق‌دهنده، داده‌ها را به منظور برقراری ارتباط میان نرم‌افزار کاربردی و Hub ترجمه می‌نماید، به این صورت که پیام‌ها تبدیل گشته، ترجمه‌شده و به سمت مقصد (نرم‌افزار به Hub یا بالعکس)هدایت می‌گردند.

توپولوژی Bus

طبیعت متمرکز مدلBroker ، تنها نقطه‌ی ضعف این مدل بود، چرا که اگر یک مؤلفه (Component) دچار مشکل شود، باعث ایجاد نقص در تمام شبکه می‌گردد .مدل Bus به عنوان راه‌حلی برای مشکلات مدل Broker ، پدید آمد. این مدل هم از یک مؤلفه مسیریابی متمرکز استفاده می‌کند با این تفاوت که مابقی وظایف را میان سایر مؤلفه‌ها تقسیم و توزیع می‌کند. این مؤلفه‌ها می‌توانند در نقاط مختلفی از شبکه، گروه‌بندی و میزبانی شوند. از قابلیت‌های دیگر این مدل می‌توان به "پردازش تراکنش‌های امنیتی (Security Transaction) "

(Processingو قابلیت رفع خطا (Error Handling) اشاره کرد؛ این قابلیت‌ها جزو مشخصه‌های مدل Bus می‌باشند. در مدلBus ، هر کدام از این قابلیت‌ها در مؤلفه‌های مجزا گنجانده‌شده‌اند. مدل Bus یک راهکار مختصر با الگویی مستحکم است که می‌تواند با کمترین حجم کد نویسی و بدون اعمال تغییر بر نرم‌افزار کاربردی، طراحی و مورد استفاده قرار گیرد. امروزه این مدل با نام ESB (Enterprise Service Bus) شناخته می‌شود.

برنامه ریزی منابع سازمان(ERP=Enterprise Resource Planning)

برنامه‌ریزی منابع سازمانی (Enterprise resource planning) که به اختصار ERPنامیده می شود، طیف وسیعی از فعالیت‌های مختلفی را شامل می شود که هدف آن، گردآوری تمام داده‌ها و فرایندهای یک سازمان در یک سیستم واحد و در نهایت بهبود عملکرد سازمان می باشد.

در واقعERP ، سامانه یکپارچه ای است که دارای اهداف، اجزا و محدوده مشخص و معینی می باشد. در ERPقرار است تمام داده ها و فرآیندهای سازمان در یک سیستم واحد جمع شود، برای نیل به این هدف، ERPاز اجزای سخت افزاری و نرم افزاری متعددی برای دستیابی به این مجموعه عظیم اطلاعات استفاده می کند.

سیستم های ERP بر اساس نوع، تلاش می کنند همه فعالیت های یک سازمان را تحت پوشش قرار دهد.

- ERP، یک علم و فن برای مدیریت منابع است.
- یک راه حل نرم افزاری است که تمام فعالیت‌های واحدهای مختلف سازمان را به طور یکپارچه در یک سیستم نرم‌افزاری واحد تعریف و ایجاد می‌کند.

در یک تعریف جامع با توجه به ویژگی‌های یک ERP می‌توان تعریف زیر را برای برنامه‌ریزی منابع سازمان ارائه نمود:

ERP یک بسته نرم افزاری کاربردی درون سازمانی، جامع و سازمان نگر، ماژولار، استاندارد و شامل یک مجموعه از ماژول‌های یکپارچه، آماده راه اندازی، از پیش طراحی شده و از پیش مهندسی شده ولی قابل تنظیم و پیکربندی بر اساس نیازهای پویای سازمان‌ها است. این راه حل کاملاً انعطاف پذیر، فرآیندگرا و اطلاعات محور بوده و شامل

کلیه فعالیت‌ها و فرآیندهای اصلی و مؤثر در ایجاد ارزش افزوده سازمان است. فرآیندهای از پیش تعریف شده در سیستم بر اساس تجربیات و گزینه‌های برتر استخراج شده‌اند.

با توجه به تعاریف فوق می‌توان دریافت مهم‌ترین ویژگی یک سیستم ERP فرآیندگرا بودن آن است. در واقع یکی از مهم‌ترین ویژگی‌های یک ERP و یکی از ضرورت‌های شکل‌گیری این‌گونه سیستم‌های سازمانی توجه به فرآیندهای سازمانی در بین حوزه‌های کارکردی است.

مزایای ERP

- کاهش هزینه‌های حمل موجودی
- کاهش هزینه‌های سفارش
- کاهش هزینه‌های تولید
- کاهش هزینه‌های نگهداری سوابق
- کاهش هزینه‌های حمل و نقل
- کاهش سرمایه گذاری در تجهیزات
- فرایندهای تولید انعطاف پذیرتر
- بهبود کارآیی که به سوددهی بیشتر یا افزایش سهم بازار منجر می‌شود
- افزایش شفافیت فرایند برای مشتری
- افزایش رضایت مشتری

معایبERP

- نصب و نگهداری این سیستم‌ها بسیار گران است.
- استفاده از بعضی از این سیستم‌ها دشوار است.
- برای به اشتراک گذاشتن برخی از اطلاعات حساس که برای یک فرایند ضروری است، با مقاومت فراد مواجه می‌شوید.

یکپارچه سازی واژه ای است که به وفور در ادبیات سیستم و فناوری اطلاعات مورد استفاده قرار می گیرد. یکپارچه سازی یک هدف است، اما راه های رسیدن به آن متفاوت است. برای رسیدن به یکپارچه سازی دو روش اصلی وجود دارد که البته کاربرد هر یک از آنها، وابسته است به سطح بلوغ سیستم های اطلاعاتی سازمان.

روش متداول رسیدن به یکپارچه سازی، کنار گذاشتن سیستم های جزیره ای و توسعه یک سیستم اطلاعاتی یکپارچه است که معمولاً در ادبیات IT با عنوان توسعه سیستم های برنامه ریزی منابع سازمانی ((ERPشناخته می شود. این روش، معمولاً مطلوب است (استراتژی مطلوب سازمان)، چون سیستم یکپارچه مطلوب را برای سازمان به ارمغان دارد، اما در بسیاری از موارد به دلایل مختلف فنی و مدیریتی ممکن نیست.

در مواردی که امکان یکپارچه سازی نوع اول ممکن نیست، استفاده از روش دوم مرسوم است، یعنی یکپارچه سازی سیستم های اطلاعاتی سازمانی با همدیگر . در این روش، به جای کنار گذاشتن سیستم های اطلاعاتی موجود سازمان، تلاش می شود تا با اتصال و یکپارچه سازی سیستم ها با هم، یکپارچه سازی محقق شود.

نتیجه آن که تفاوت این دو نوع از یکپارچه سازی به کنار گذاشتن یا استفاده نمودن از سیستم های اطلاعاتی موجود مربوط است.

مدیریت فرایندهای کسب و کار (Business Proccess Management=BPM)

مدیریت فرآیند کسب و کار(Business Process Management) ، بسته به رویکرد افراد مختلف، تعاریف متفاوتی دارد. به طور کلی BPM عبارت است از تحلیل فرآیندهای کسب و کار با استفاده از مدل‌ها و سناریوهای متفاوت به منظور درک شیوه انجام فعالیت‌ها توسط یک سازمان بزرگ .این امر موجب ایجاد بینشی جدید برای ارائه توصیه و آزمون بهینه‌سازی‌ها و بهبودهای سازمانی شده و تکرار مجدد فرآیند برای بهبود مداوم و بهینه‌سازی فرآیند را باعث می‌شود.

مدیریت فرآیند کسب و کار ، شامل متدها، تکنیک‌ها و ابزاری است که برای پشتیبانی از طراحی، اجرا، مدیریت و آنالیز فرآیندهای عملیاتی کسب و کار بکار می‌رود.

مدیریت فرآیند کسب و کار با داشتن الگوهای متعدد مورد نیاز سازمان‌ها، روشی یکپارچه برای تعریف، اجرا، بازبینی و مدیریت فرآیندهای کسب و کار سازمان ها را ارایه می‌کند و با استفاده از متدها و ابزارهای مربوطه، حجم کار توسعه راه کارها را به حداقل رسانده و مدیریت امور را سهل و کارآمد می‌کند. بر این اساس می‌توان گفت مدیریت فرایند کسب و کار مجموعه‌ای است از رویکردهای مدیریتی و فناوری مدرن که به نحوی ساختاریافته، منسجم و هماهنگ، برای درک و مستند سازی، مدل‌سازی، تحلیل و بهبود مستمر فرایندهای کسب و کار مورد استفاده قرار می گیرد BPM مزایای ذیل را بدنبال دارد:

- توسعه و تعالی سازمانی
- ایجاد مزیت رقابتی و جریان ارزش پایدار
- جلب رضایت مشتریان
- چابکی سازمان و فرایندهای سازمانی

- یکپارچه سازی سازمان و فرایندهای سازمانی

در واقع **BPMیا مدیریت فرایندهای کسب و کار**، ادغامی از دانش و تحقیق در هم‌گرایی و اتصال بین مدیریت و فناوری اطلاعات است که شامل متدها، تکنیک‌ها و ابزاری است برای طراحی، فعال شدن، کنترل و آنالیز فرآیندهای کسب و کار که خود مستلزم نیروی انسانی، سازمان‌ها، برنامه‌های نرم‌افزاری کاربردی، اسناد و دیگر منابع اطلاعاتی است.

تفاوت بین مدیریت فرایند کسب و کار و سیستم های برنامه ریزی منابع سازمانی چیست؟

تفاوت اصلی بین BPM و سیستم های ERP این است که BPM بسیار متمرکز بر **فرآیند** است، در حالیکه ERPعمدتاً محدود به عملکرد سازمانی است. اگر شما به دنبال یک سیستم برای درونی کردن و خودکارسازی برخی از فرایندهای کسب و کار اصلی تان هستید، پس ERP احتمالا گزینه خوبی برای شما خواهد بود. با این حال اگر به تجزیه و تحلیل کسب و کار بیشتر نیاز دارید و قابلیت های بهبودی که یک سیستم BPM به ارمغان می آورد، پس این گزینه، بسیار بهتر خواهد بود.

تفاوت بین مدیریت فرایند کسب و کار (BPM) و برنامه ریزی منابع سازمانی (ERP) می تواند کاملا گنگ باشد. به نظر می رسد که BPM :یک عمل است، در حالی که ERP تعدادی فن آوری است که می تواند این عمل را پشتیبانی کند. با این گفتار، BPMمی تواند با فناوری یا بدون فناوری به دست آید، هرچند داشتن یک سیستم مدیریت فرآیند کسب و کار (BPMS) و یا یک ERP توانایی شما برای بکارگیری عملکرد BPM را افزایش می دهد.

ERP، از سوی دیگر، تنها می تواند از طریق فناوری (و مهندسی مجدد فرآیند) به دست آید. و این موضوع به علت یکپارچه سازی اطلاعات در عملکردهای کسب و کاراست .باERP ، سازمان با استفاده از چند ماژول ، به یک منبع داده منحصر به فرد، دسترسی پیدا می کند. به عنوان مثال منابع انسانی، تدارکات و غیره. برخی از اپلیکیشن های سازمانی ممکن است قابلیت BPM را داشته باشند، اما دسترسی آنها معمولاً چیزی فراتر از نرم افزار (application)نیست.

برخلاف نرم افزار ERP که ممکن است دارای ماژول های یکپارچه باشد، اجزای BPMS معمولاً همیشه به صورت یک کل یکپارچه در دسترس نیستند.

BPMدر مرکز خود، بر روی بهینه سازی کارایی فرآیندهای موجود، تمرکز دارد؛ نظارت بر اثربخشی فرآیندها ، مدل سازی فرآیندها و شبیه سازی. بنابراین BPM می تواند به عنوان یک سطح "فرآیند_هوشمند" توصیف شود که می تواند با یک ERPسازمانی ادغام شود و یا نشود.

معماری سرویس گرا

واقعیت موجود در سازمان های IT این است که زیربنا در میان سیستم های عامل، برنامه های کاربردی، نرم افزارهای سیستمی، و زیربنای کاربردی به صورت ناهمگن است. برخی برنامه های کاربردی موجود برای اجرای فرایندهای فعلی تجارت مورد استفاده قرار گرفته اند، بنابراین آغاز از صفر برای ساختن زیربنای جدید یک رویکرد قابل انتخاب محسوب نمی گردد .سازمان ها باید به شکلی سریع به تغییرات تجاری واکنش نشان دهند؛ از سرمایه های موجود در برنامه های کاربردی و زیربنای کاربردی به منظور تمرکز بر روی نیازمندی های تجاری جدیدتر استفاده نمایند؛ کانال های جدید تعامل با مشتریان، شرکا، و تامین کنندگان را پشتیبانی کنند؛ و یک معماری که تجارت ارگانیک را پشتیبانی نماید به کار گیرند SOA .با طبیعت اتصال آزادانه خود به سازمان ها امکان بهره گیری از سرویس های جدید یا ارتقای سرویس های موجود را به شیوه ای قطعه قطعه به منظور تمرکز بر نیازمندی های تجاری فراهم می آورد، امکانی را برای قابل استفاده نمودن سرویس ها در کانال های متفاوت فراهم می سازد، و سازمان موجود و برنامه های کاربردی نسل قبل را به عنوان سرویس ها ارائه می کند، در نتیجه سرمایه های زیربنای IT موجود را حراست می نماید .

یک سازمان استفاده کننده از SOA می تواند یک برنامه کاربردی مرکب زنجیره تامین را با استفاده از مجموعه ای از برنامه های کاربردی موجود که کارکرد خود را از طریق رابط های استاندارد ارائه می دهند، ایجاد نماید .

معماری سرویس گرا به عنوان یکی از آخرین دستآوردها در تولید نرم افزار، به نظر می رسد، در سالهای آتی معماری غالب صنعت فناوری اطلاعات و ارتباطات باشد. علت بوجود آمدن این معماری، ایده ای بود که در ذهن تعدادی از معماران آن وجود داشت و آن نرم افزار به عنوان سرویس بود. در مدل نرم افزار به عنوان سرویس شما نرم افزار خود را بگونه ای طراحی می کنید که قابل استفاده توسط سیستم های دیگر باشد یعنی دیگران می توانند برای استفاده از سرویس شما ثبت نام کنند و هر موقع که لازم داشتند از خدمات آن بهره ببرند، همانند

حالتی که در مورد شبکه های تلویزیون کابلی وجود دارد. تا زمانی که شما به سرویس متصل هستید، شما می توانید هر لحظه که خواستید از سرویس استفاده کنید. برای مدتهای طولانی برنامه نویسان سعی می کردند تا، کدهای خود را بصورت modular بنویسند، تا بتوان از آن در تولید نرم افزارهای دیگر استفاده کرد. تفاوت نوشتن کد بصورت modular و بر اساس معماری سرویس گرا در حجم مخاطبان آن است. دوباره به همان مثال اول برمی گریم، وقتی شما کد خود را به منظور قابل استفاده بودن توسط نرم افزارهای دیگر، به شکل modular می نویسید مانند این است که، یک شبکه تلویزیون کابلی درون یک ساختمان خاص دارید و بنابراین فقط ساکنین آن ساختمان می توانند از آن بهره برداری کنند .در جهان امروز طیف مخاطبانی که بالقوه می توانند از سرویس شما استفاده کنند، کل کاربران روی شبکه اینترنت است. بنابراین باید مکانیزمی بوجود می آمد، که می توانست پاسخگوی این محیط جدید (اینترنت) و کاربران آن باشد و بنابراین معماری سرویس گرا بوجود آمد. این معماری توسط دو شرکت IBM, Microsoft بوجود آمد، که هر دو شرکت طی سالهای اخیر از حامیان اصلی سرویسهای وب و عامل بسیاری از ابداعات جدید در حیطه سرویس های وب، مانند UDDI ,WSEبوده اند. قابل ذکر است، که در آخرین معماری در حال توسعه، در تولید نرم افزار که هنوز هم در مرحله تحقیقاتی است (MDA) ، تدابیری جهت هماهنگی با معماری سرویس گرا در نظر گرفته شده است. از نمونه های استفاده از این معماری در کشور خودمان، سازمان ثبت احوال کشور است که موظف شده تا پایگاه های اطلاعاتی خود را بصورت سرویس وب و مبتنی بر این معماری به سایر نهادها مانند نیروی انتظامی و سایر دستگاه ها ارائه دهد.

در واقع معماری سرویس گرا مجموعه قوانین، سیاستها و چهارچوبهایی است که نرم‌افزارها را قادر می‌سازد تا عملکرد خود را از طریق مجموعه سرویس‌های مجزا و در عین حال مربوط به هم در اختیار سایر درخواست کنندگان قرار دهند تا بتوانند بدون اطلاع از نحوه پیاده‌سازی و تنها از طریق رابطه‌ای استاندارد و تعریف شده، این سرویس‌ها را پیدا کرده و فراخوانی نمایند و معماری سرویس‌گرا روشی برای ساخت سیستم‌های توزیع شده‌ای است که در آنها عملکرد سیستم به‌صورت سرویس در اختیار کاربران و یا سایر سرویس‌ها قرار می‌گیرد.

به بیان دیگر معماری سرویس‌گرا رهیافتی است برای ساخت سیستم‌های توزیع شده که فرایند کسب و کار و دیگر خدمات را در قالب سرویس ارائه می‌کند. این سرویس‌ها هم توسط دیگر نرم افزارها قابل فراخوانی هستند و هم برای ساخت سرویس‌های جدید مورد استفاده قرار می‌گیرند، این رهیافت برای یکپارچه سازی فناوری‌ها در محیطی که انواع مختلفی از پلتفرم‌های نرم افزاری و سخت افزاری وجود دارد ایده آل است .نهایتا، معماری سرویس‌گرا انعطاف‌پذیری بیشتری را برای سازمان‌ها در ساختن برنامه‌های کاربردی و فرایندهای تجاری ایجاد می‌کند تا در حالی که به استفاده از برنامه‌ی کاربردی موجودشان ادامه می‌دهند قادر به تولید سریع‌تر سرویس‌های جدید باشند.

سرویس

در دنیای گرداگرد ما نمودهای سرویس (خدمت) به وفور دیده می شود. در واقع عمر ارایه سرویس توسط بشر به اندازه عمر تمدن بشر است. هر فردی که وظیفه خاصی را در راستای خدمت رسانی به افراد دیگر انجام می دهد در واقع سرویس ارایه می دهد. به عنوان مثال راننده، نقاش، پزشک و ... . به علاوه مجموعه ای از افراد که هر یک وظیفه منفک از دیگری داشته اما در یک راستا کاری را به انجام می رسانند اقدام به ارایه سرویس می کنند. مانند افراد مختلف یک شرکت هواپیمایی که مشغول ارایه سرویس پرواز به مشتریان هستند. و به همین صورت است یک سازمان یا شرکت که خدمتی ارایه می کند

سرویس ها اجزاء مستقلی هستند که پیغام های XML با ساختار مشخص و خوش تعریف (Well-defined) را پردازش می کنند.

یک سرویس یک واحد مجزا و مستقل از یک وظیفه‌مندی است که می تواند به صورت مستقل و با کمترین وابستگی از دیگر بخش‌ها استفاده، بروزرسانی و عمل کند. ویژگی‌های یک سرویس عبارتند از:

- منطقاً نشان دهنده یک فعالیت کسب‌وکار با خروجی معیین است.
- خودمختار است.
- محتویات و جزئیات آن بر مشتریان سرویس پوشیده است.
- سرویس ممکن است شامل دیگر سرویس‌ها برای ارائه سرویس خود باشد.

سرویس کاری است که به وسیله یک سرویس دهنده انجام می شود که ممکن است انجام یک درخواست کوچک روی داده مانند دریافت یا ذخیره اطلاعات باشد یا مربوط به انجام کاری پیچیده تر مانند چاپ یک تصویر باشد.

انواع سرویس :

- فرایندی
- حرفه
- نرم افزاری

فرایندی	حرفه	نرم افزاری
انتخاب واحد دانشگاه	-تهیه لیست واحدهای قابل ارائه -تهیه فهرست اساتید برای ارائه واحدها -کنترل عدم تداخل واحدهای ارائه شده -بازگشایی فایل آموزشی دانشجویان -تغییر ظرفیت واحدهای ارائه شده	-ورود شناسه کاربری و رمز عبور به سیستم آموزش -درج مقادیر در جدول واحدهای ارائه شده درج مقادیر د جدول اساتید -مشاهده جدول زمانبندی واحدهای ارائه شده برای کنترل تداخل -اصلاح مقادیر فیلد ظرفیت واحدها
ثبت نام دانشجویان	-ثبت اطلاعات دانشجویان و تشکیل پرونده -صدور کارت دانشجویی و کارت غذا -ارسال معرفی نامه دانشجو به دانشکده مربوطه	-درج مقادیر در جدول مشخصات دانشجو -درج اطلاعات سوابق تحصیلی دانشجو در جدول سوابق آموزشی -اسکن و ذخیره تصویر دانشجو -اخذ مدارک مربوطه و بایگانی الکترونیکی آنها -چاپ کارت دانشجویی
صدور گواهی اشتغال به تحصیل	-کنترل وضعیت دانشجو -چاپ وضعیت تحصیلی دانشجو مهرنمودن آن	-مشاهده گزارش وضعیت تحصیلی دانشجو -چاپ گزارش وضعیت تحصیلی دانشجو

معماری سرویس

چندین مصرف کننده سرویس می توانند با ارسال پیام اقدام به فراخوانی سرویس ها نمایند. این پیام ها معمولا توسط یک گذرگاه سرویس تغییر شکل داده شده و به سوی یک پیاده سازی سرویس مناسب هدایت می گردند. این معماری سرویس می تواند یک موتور قواعد تجاری را فراهم سازد که امکان تلفیق قواعد تجاری در یک سرویس یا چندین سرویس را عملی سازد. معماری سرویس مزبور همچنین یک زیربنای مدیریت سرویس فراهم می آورد که سرویس ها و اعمالی از قبیل بازرسی، پرداخت صورتحساب، و واقعه نگاری (logging) را مدیریت می نماید. به علاوه، این معماری انعطاف پذیری ناشی از دارا بودن فرایندهای تجاری تغییر پذیر را به سازمان ها ارزانی می دارد، فرایندهایی که نیازمندی های تنظیمی همانند Sarbanes Oxley (SOX)i را مد نظر قرار می دهند، و سرویس های اختصاصی را بدون تحت تاثیر قرار دادن سایر سرویس ها تغییر می دهند .

اجزی تشکیل دهنده یک معماری سرویس‌گرای ساده عبارتند از

-درخواست‌کننده سرویس

-تهیه‌کننده سرویس

-فهرست سرویس‌ها

تبادل اطلاعات بین این اجزا به‌وسیله پیغام و رابط استاندارد صورت می‌گیرد. نحوه ارتباط این سه جزء شامل موارد ذیل است :

-منتشر کردن سرویس

-پیدا کردن سرویس

-متصل شدن به سرویس

واژه معماری سرویس گرا در دوم آوریل سال 1996 توسط گروه گارتنر به کار برده شد. معماری سرویس گرا یک پایه و اساس برای معماری است؛ یک رویکرد طراحی است؛ یک معماری سرویسگرا از یک مجموعه بی پایان از سرویسها ، ارائه کنندگان سرویس و مصرف کنندگان سرویس تشکیل شده است که آنها از طریق واسط ها ی استاندارد ارتباط برقرارمیکنند.

پروتکل های معماری سرویس گرا

- SOAP, WSDL, UDDI

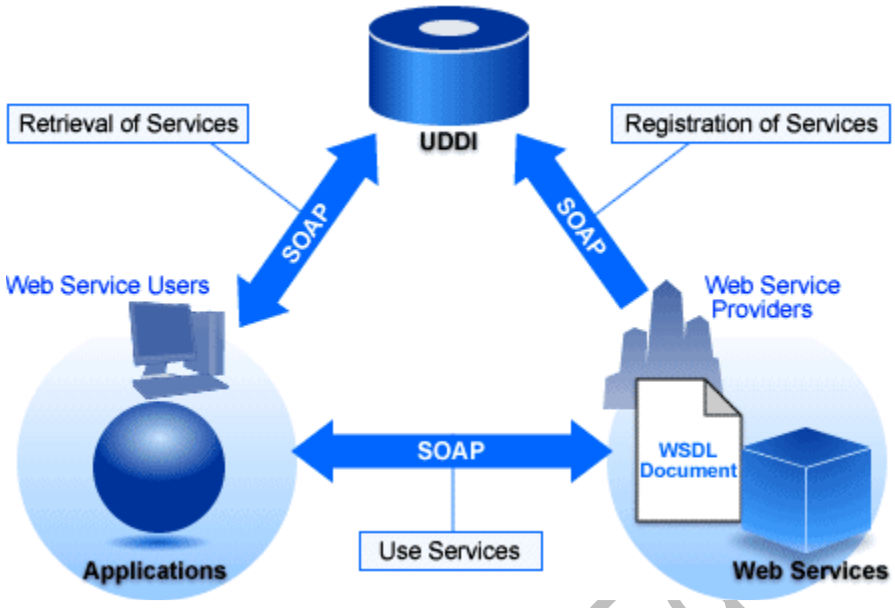
WSDL، UDDI، و SOAP قطعات اساسی زیربنای SOA هستند.

WSDL برای توصیف سرویس به کار برده شده است؛

UDDI، برای ثبت و جستجوی سرویس ها؛

وSOAP ، به عنوان یک لایه نقل و انتقال جهت ارسال پیام ها میان مصرف کننده سرویس و فراهم کننده سرویس است .

در حالی که SOAP ساز و کار پیش فرض برای سرویس های وب است، تکنولوژی های جایگزین، انواع دیگری از انقیادها (binding) را برای یک سرویس تحقق می بخشند. یک مصرف کننده می تواند به جستجوی یک سرویس در رجیستری UDDI بپردازد، WSDLرا برای سرویسی که دارای توصیف است تهیه نماید، و سرویس را از طریق SOAPفراخوانی کند .



الگوهای یکپارچه سازی از گذشته تاکنون

تاکنون از الگوهای یکپارچه سازی مختلفی استفاده شده است . ارتباط بین برنامه ها و یا یکپارچه سازی برنامه ها می تواند در دو سطح مبادله داده (اشتراک داده) و یا مبادله توانمندی (اشتراک قابلیت ها و یا توانمندی) صورت پذیرد .

به عنوان مثال ، فرض کنید سیستم "الف" ، سیستمی است که اطلاعات کارکنان یک سازمان را درخود نگهداری کرده باشد و سیستم "ب "، سیستمی است که توان و یا قابلیت محاسبه مالیات حقوق را به صورت یک پتانسیل دارا باشد. در چنین مواردی ،می توان حداقل دو احتمال را متصور دانست : احتمال اول این که سیستم "ب" درخواست اطلاعات یک کارمند بخصوص را از سیستم "الف" داشته باشد (اشتراک داده) و احتمال دوم این که، سیستم "الف " نیازمند استفاده از خدمات محاسبه مالیات حقوق تعبیه شده در سیستم "ب " باشد (اشتراک قابلیت).

اشتراک داده:

برنامه ها در ابتدا تمایل داشتند که داده بین خود را به اشتراک بگذارند . اشتراک داده مبتنی بر فایل ،استفاده از بانک های اطلاعاتی و در نهایت سوکت ، راه حل های ارایه شده جهت پاسخ به نیاز اشتراک داده بین برنامه ها از گذشته تاکنون می باشد . شکل ۱ ، مهمترین ویژگی هر یک از راه حل های فوق را نشان می دهد.

شکل ۱ : روش های مختلف اشتراک داده بین برنامه ها

سوکت :

یک کانال رابط برنامه‌نویسی نرم‌افزار (به انگلیسی: Socket Application Programming interface یا Socket API) ، که اجازه می‌دهد یک برنامه کاربردی کانالهای شبکه را استفاده و مدیریت نماید، معمولاً توسط

سیستم عامل ارائه می‌گردد. کانالهای اینترنت **رابط برنامه‌نویسی نرم‌افزار** معمولاً براساس استاندارد کانال برکلی (به انگلیسی: Berkeley sockets) هستند.

یک **نشانی کانال** ترکیبی از نشانی پروتکل اینترنت (IP) و شماره درگاه (Port Numebr) است، و بسیار شبیه به یک شماره تماس تلفنی است که ترکیبی از پیش شماره تماس و یک شماره داخلی که به پیش شماره افزوده گردیده است. بر پایه این نشانی، کانالهای اینترنت، بسته‌های اطلاعات دریافتی را به برنامه‌های کاربردی مرتبط یا پردازش و ریسه مناسب تحویل می‌دهند.

انواع سوکت‌ها

اگر بخواهیم از نظر اهمیت انواع سوکت را معرفی کنیم دو نوع سوکت بیشتر وجود ندارد.(انواع دیگری هم هستند ولی کم اهمیت ترند). این دو نوع سوکت عبارتند از :

- سوکت‌های نوع استریم که سوکت‌های اتصال گرا (connection oriented) نامیده می‌شود.
- سوکت‌های نوع دیتاگرام که سوکت‌های بدون اتصال (connectionless) نامیده می‌شوند.

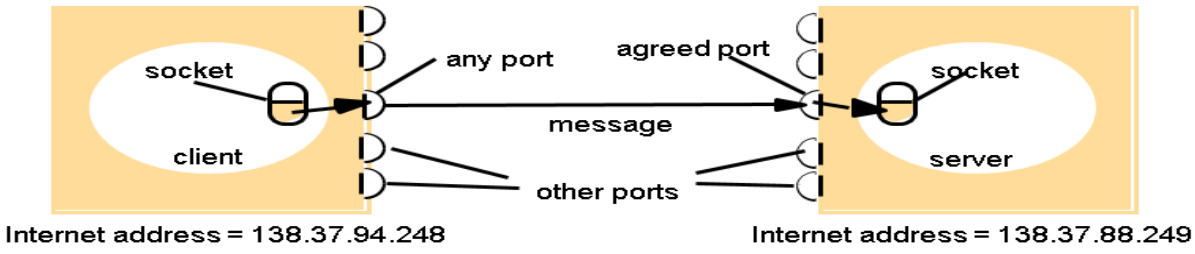
سوکت یک مفهوم انتزاعی از تعریف ارتباط در سطح برنامه‌نویسی است و برنامه‌نویس با تعریف سوکت عملا تمایل خود را برای مبادله داده‌ها به سیستم عامل اعلام کرده و بدون درگیر شدن با جزئیات پروتکل TCP یا UDP از سیستم عامل می‌خواهد تا فضا و منابع مورد نیاز را جهت برقراری یک ارتباط، ایجاد کند .منظور از این جمله که "**سوکت یک مفهوم انتزاعی است** "آن است که چیزی به نام سوکت وجود خارجی ندارد، بلکه سیستم عامل آن را تجسم بخشیده است !

سوکت‌های نوع استریم

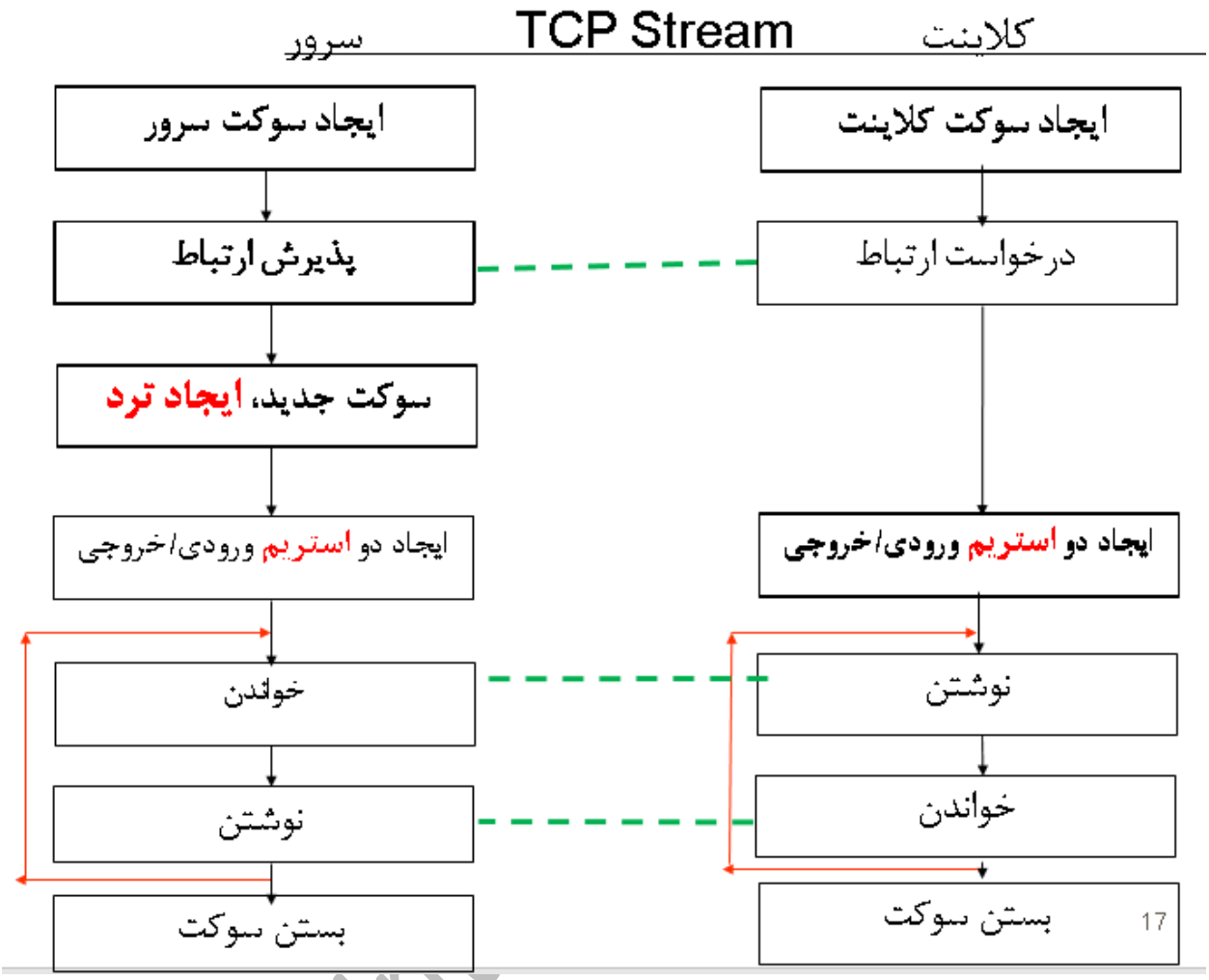
روش ارسال برای سوکت‌های نوع استریم همان روش TCPاست و بنابراین داده‌ها با رعایت ترتیب، با اطمینان صد در صد و با نظارت کافی بر خطاهای احتمالی مبادله می‌شوند. به عنوان مثال پروتکل انتقال فایل(FTP) ، پروتکل انتقال صفحات ابرمتن (HTTP) یا پروتکل انتقال نامه‌های الکترونیکی (SMTP)همگی نیازمند برقراری یک ارتباط مطمئن و عاری از خطا هستند و طبعا از سوکت‌های **نوع استریم** بهره می‌برند .سوکت‌های **نوع استریم** دقیقا بر روی پروتکل TCP بوده و طبیعتا قبل از مبادله داده‌ها باید یک اتصال به روش دست تکانی سه مرحله‌ای (Three Way Handshake) بین دو پروسهٔ نهایی برقرار بشود .

سوکت‌های نوع دیتاگرام

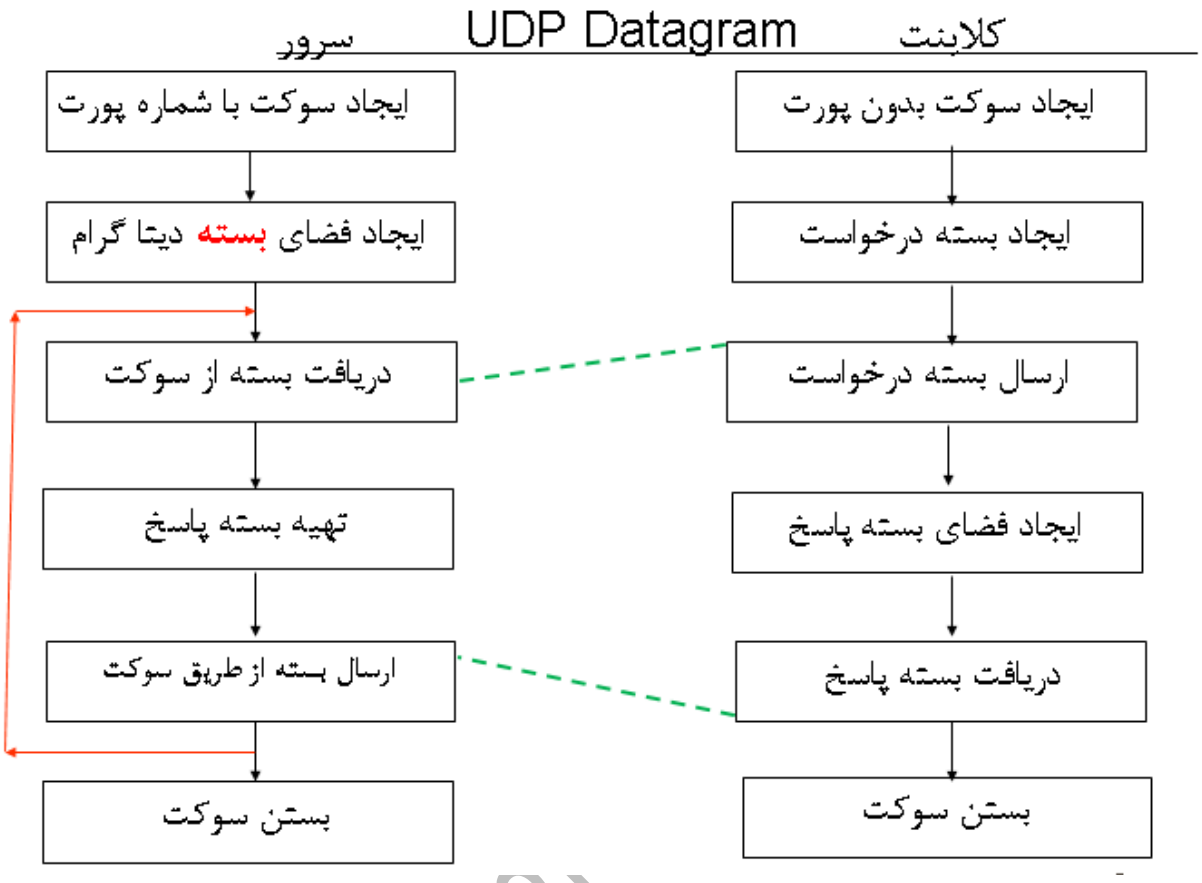
سوکت نوع دیتاگرام نامطمئن است و هیچگونه تضمینی در ترتیب جریان داده‌ها وجود ندارد. سوکت نوع دیتاگرام مبتنی بر پروتکل UDP است و بدون نیاز به برقراری هیچ ارتباط یا اتصال، داده‌ها مبادله می‌شوند و بنابراین تضمینی در رسیدن داده‌ها، صحت داده‌ها و حفظ ترتیب داده‌ها وجود ندارد ولی با تمام این مشکلات باز هم در برخی از کاربردها مثل انتقال صدا و تصویر مورد استفاده قرار می‌گیرد. تنها حُسن استفاده از سوکت‌های دیتاگرام، سرعت تحویل داده‌ها است.



چگونگی ارتباط در سوکت ها



نحوه ارتباط در سوکت های نوع TCP



نحوه ارتباط در سوکت های نوع UDP

یکی از آرزوهای دیرینه هر سیستم نرم افزاری زیستن در محیطی است که بتواند در صورت ضرورت با سایر سیستم ها تعامل برقرار نماید و جلوه ای از یک زندگی مدنی را به تصویر بکشد. تصویری که هم بتوان در صورت نیاز ، سخن و درخواست دیگران را شنید و هم بتوان سخن و درخواست خود را به گوش سایر سیستم ها رساند. این نوع تعاملات می تواند شامل درخواست یک داده بخصوص و یا به خدمت گرفتن یک پتانسیل خاص از یک سیستم دیگر باشد. تعاملاتی که ریشه در نیاز به یکپارچه سازی داده و یا برنامه ها را در یک سازمان از گذشته تاکنون نشان می دهد.

یکپارچه سازی تمامی برنامه های موجود در یک سازمان با هدف پایداری و یکنواختی داده و قابلیت ها ، مشکلات و مسائل مختص به خود را دارد . چراکه این کار شامل یکپارچگی گروه برنامه های مختلفی نظیر برنامه های سفارشی) برنامه های نوشته شده با زبان هایی نظیر (C/C++,Java/J2EE ، بسته های نرم افزاری)نظیر برنامه های CRM و (ERP و برنامه های قدیمی (برنامه های نوشته شده بر روی سیستم های mainframe و ...) است .

اشتراک توانمندی :

علاوه بر اشتراک داده ، اشتراک توانمندی بین برنامه ها یکی از نیازهای حیاتی در بحث یکپارچگی بین برنامه ها است . مهمترین نقطه ضعف سه روش اشاره شده دربخش قبل ، عدم توان آنها در به اشتراک گذاشتن قابلیت ها بین برنامه ها است . چگونه می توان برنامه های یک سازمان را به هم یکپارچه کرد تا امکان به اشتراک گذاشتن قابلیت ها و توانمندی فراهم گردد ؟ بخاطر داشته باشید که یکپارچه سازی برنامه های یک سازمان یک فرآیند مشکل است چراکه این کار مستلزم درگیر شدن با انواع مختلفی از برنامه های نوشته شده با زبان های برنامه نویسی متفاوت است که هر یک بر روی یک پلت فرم خاص اجراء می شوند .

روشهای مبتنی بر فایل و مبتنی بر سوکت، تنها محدود به تسهیم داده هستند، و به برنامه های کاربردی اجازه تسهیم عملکرد را نمی دهند.

سوکت مفهوم اتصال برنامه های کاربردی را معرفی کرد که برای تسهیم عملکرد، ضروری هستند.

به عبارت دیگر، هنگامی که برنامه های کاربردی عملکرد را به اشتراک می گذارند، بدون توجه به روش یکپارچه سازی، سوکت ها همیشه تقریبا در پس زمینه هستند.

اشتراک عملکرد :

توابع تعریف شده در یک برنامه کاربردی، توسط سایر برنامه های کاربردی سازمان فراخوانی شوند.

یکپارچه سازی کاربردهای سازمانی، فرآیند دشواری است چرا که انواع بسیار متفاوتی از برنامه های کاربردی را در بر میگیرد که به زبان های متفاوتی نوشته شده اند و روی انواع متفاوتی از پلت فرم ها که ممکن است از لحاظ جغرافیایی، توزیع شده باشند، اجرا می شوند.

انواع روشهای فراخوانی توابع

- فراخوانی همگام
- فراخوانی ناهمگام

توابع همگام

تابع فراخواننده، مسدود میشود و تا زمانی که کنترل از تابع فراخوانی شده، بازنگردد، قادر به انجام هیچ کاری نیست.

توابع ناهمگام

کد تابع فراخواننده می تواند برای انجام کارهای دیگر ادامه پیدا کند، بدون اینکه منتظر برگشت کنترل از تابع فراخونی شده باشد.

فراخوانی رویه راه دور : (RPC)

RPC برگرفته شده از(Remote Procedure Call)که به دلیل مطرح کردن برخی مفاهیم کلیدی نقطه عطفی در یکپارچه سازی برنامه ها است به همراه CORBA (برگرفته شد از Common Object Request

Broker Architecture) و در نهایت ESB (برگرفته شده از Enterprise Service Bus) از جمله تلاش های مهم جهت به اشتراک گذاشتن قابلیت ها و در نهایت یکپارچه سازی محسوب می گردند . ایده اصلی RPC، فراخوانی روتین موجود بر روی یک کامپیوتر راه دور توسط یک کامپیوتر محلی است که از یک معماری دو لایه تبعیت می کند RPC.با ارایه مفاهیمی همچون ارایه دهنده سرویس ، استفاده کننده سرویس ، تعریف اینترفیس از طریق استفاده از یک مشخصه فایل و نحوه ارسال پارامتر مورد نیاز یک برنامه از سمت سرویس گیرنده به سمت سرویس دهنده ، گام های بسیار مهمی را به منظور ایجاد زیرساخت مفهومی و اطلاعاتی لازم جهت یکپارچه سازی سازمانی برداشت RPC . مستقل از زبان برنامه نویسی نیست و می بایست استفاده کننده از سرویس و مصرف کننده سرویس از یک زبان برنامه نویسی مشابه استفاده نمایند . همچنین ، ارتباط بین سرویس گیرنده و سرویس دهنده به صورت نقطه به نقطه است .توجه داشته باشید که هدف ، یافتن راه حلی جامع در خصوص یکپارچه سازی است که بتواند بر روی سه محور اساسی تاکید نماید :

- قابلیت استفاده مجدد از کد
- مستقل از زبان های برنامه نویسی
- مستقل از پلت فرم

انواع توابع همگام :

RPC : فراخوانی رویه راه دور

RPCکنترل شده : کد تابع فراخواننده و فراخوانی شده دردو برنامه مختلف روی یک ماشین اجرا می شوند.این روش شبیه روش RPC است با این تفاوت که نیازی به اتصالات شبکه ای نیست.و از هسته سیستم عامل برای برقراری ارتباط استفاده میشود.ایجاد کتابخانه ها در سی شارپ نمونه ای از این نوع فراخوانی ها میباشد.

فراخوانی توابع محلی : در این نوع کد تابع فراخواننده و فراخوانی شده هردو بخشی از یک برنامه هستند و .هردو تابع روی یک ماشین هستند.مانند نوشتن یک تابع درون یک برنامه (یا ایجاد یک کلاس) و استفاده از آن در همان برنامه .

معایب روش RPC :

- قابلیت استفاده مجدد کد در این روش ضعیف است، زیرا کدهای تجمیع، تفکیک و ارتباطات شبکه در برنامه های کلاینت و سرور نوشته شده اند.
- این روش مستقل از زبان نیست و کلاینت و سرور باید از یک زبان برنامه نویسی یکسان استفاده کنند.
- نوع پیوند بین برنامه های کاربردی پیوند قوی است، زیرا فراخوانی ها همگام هستند.
- (Peer to Peer)نیست، یعنی در این روش نقش کلاینت و سرور ثابت است و رابطه به صورت همتا به همتا نیست .یعنی کلاینت به عملکرد تعبیه شده در سرور دسترسی دارد ولی عکس آن امکان پذیر نیست.
- از روش ارتباط نقطه به نقطه استفاده می کند، بنابراین هنگامی که تعداد زیادی برنامه نیاز به یکپارچه سازی دارند مناسب نیست.
- اگر تعداد زیادی از فراخوانی های راه دور درگیر باشند این روش مناسب نیست زیرا به دلیل ماهیت ذاتی فراخوانی همگام کلاینت نمی تواند قبل از اینکه سرور کارش را تمام کند کار خود را ادامه دهد. می توان این مشکل را با استفاده از برنامه نویسی چند رشته ای حل کرد اما این کار پیچیدگی برنامه را افزایش داده و ریسک هایی به همراه دارد.

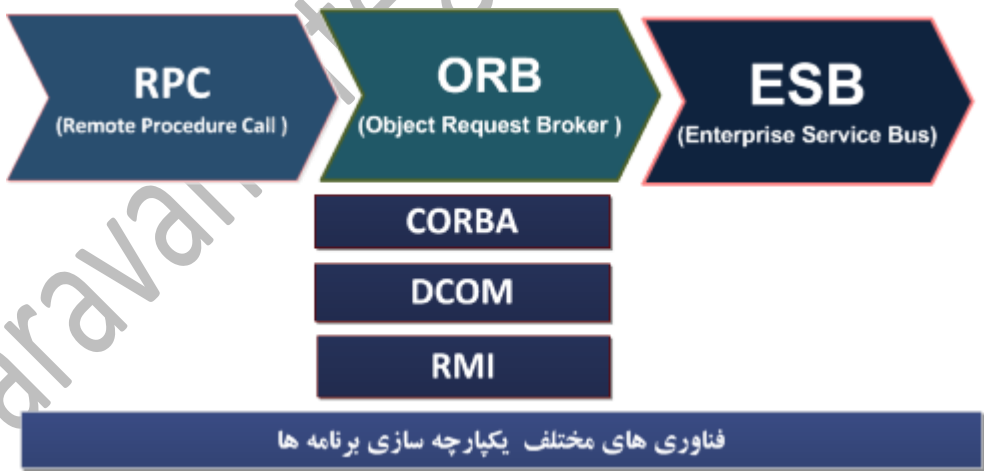
ORB(برگرفته شده از Object Request Broker)

یکی از راه حل های ارایه شده در راستای تحقق موارد اشاره شده است که سه مدل متفاوت از آن در دسترس می باشد :

- CORBAبرگرفته شده از (Common Object Request Broker Architecture)
- DCOMبرگرفته شده از(Distributed Component Object Model)
- RMIبرگرفته شده از(Remote Method Invocation)

CORBAرایج تر است DCOM بیشتر وابسته به یک پلت فرم است (سیستم عامل ویندوز) و RMI به پلت فرمی وابسته نیست ولی تنها می تواند توسط زبان برنامه نویسی جاوا استفاده شود .

شکل ۲ ، برخی فناوری های موجود به منظور یکپارچه سازی برنامه ها رانشان می دهد.

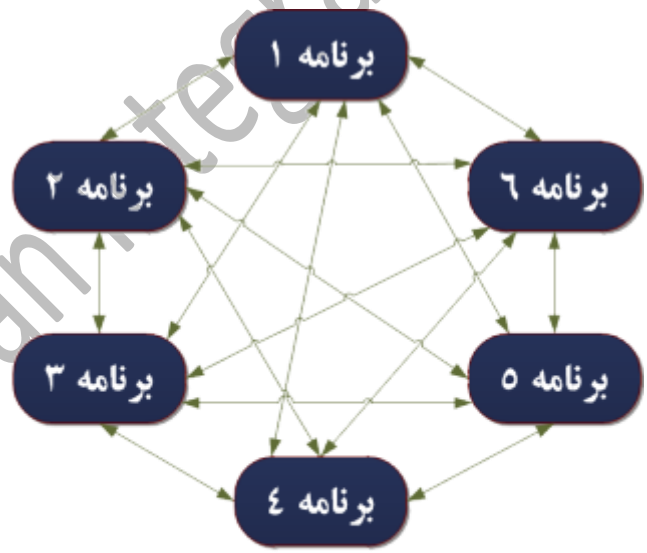


شکل ۲ : فناوری های مختلف یکپارچه سازی برنامه ها

گذرگاه سرویس سازمانی : ESB

سرویس های وب صرفا به یک بخش از راه حل یکپارچه سازی پاسخ شایسته ای می دهند و برخی از مشکلات ناهمگونی همچنان حل نشده باقی می ماند. به عنوان نمونه در اغلب موارد ما به مکانیزمی جهت پروتکل ارتباطی و روشی جهت تبدیل داده به منظور تامین نیازهای یک ارایه دهنده سرویس در جهت پاسخ به خواسته یک درخواست کننده سرویس ، نیاز خواهیم داشت .

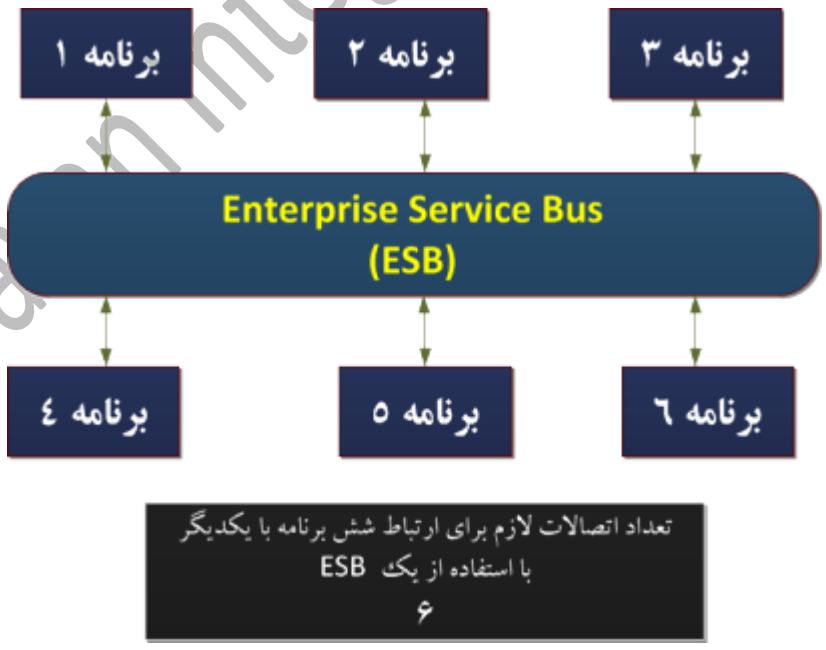
ESB، یک روش جامع و قابل توسعه به منظور ارتباط تعداد زیادی از برنامه ها بدون نیاز به برقراری یک ارتباط مستقیم بین هر زوج برنامه است . ارتباطات از نوع نقطه به نقطه برای سازمان های بزرگ که دارای تعداد زیادی برنامه می باشند ، مناسب نیست . شکل ۳ ، تعداد ارتباطات مورد نیاز جهت اتصال و یا یکپارچه سازی شش برنامه را نشان می دهد .



شکل ۳ : تعداد ارتباطات مورد نیاز جهت اتصال و یا یکپارچه سازی شش برنامه با روش نقطه به نقطه

فرض کنید بخواهیم ده برنامه را در یک سازمان به هم مرتبط نمائیم ، با بکارگیری روش نقطه به نقطه به ۴۵ اتصال نیاز خواهیم داشت .در صورتی که با بکارگیری یک ESB صرفا به ده اتصال نیاز خواهیم داشت . این وضعیت در خصوص ESB صدق نخواهد کرد چراکه ارتباطات به صورت نقطه به نقطه نخواهند بود . یکپارچگی مبتنی بر سرویس های وب همچنان به صورت نقطه به نقطه است . هر زوج برنامه موجود در سازمان لازم است دارای یک ارتباط جداگانه مختص به خود باشند . رویکرد فوق برای سازمان هایی با تعداد برنامه های کم ، می تواند مفید باشد ولی برای یک سازمان بزرگ نمی توان بر روی آن حساب باز کرد و از آن به عنوان یک راه حل مناسب یکپارچه سازی استفاده کرد .

ESB، بدون این که به حریم خصوصی برنامه ها تعرض نماید ،با هر برنامه با زبان خود برنامه ارتباط برقرار می نماید .شکل ۴ ، تعداد ارتباطات مورد نیاز جهت اتصال و یا یکپارچه سازی شش برنامه را به کمک یک ESB نشان می دهد .



شکل ۴ : تعداد ارتباطات مورد نیاز جهت اتصال و یا یکپارچه سازی شش برنامه توسط یکESB

گزینه های مختلف یکپارچه سازی برنامه ها

برای یکپارچه سازی برنامه ها از گزینه های مختلفی می توان استفاده کرد:

۱) نقطه به نقطه سفارشی :

یک ارتباط مستقیم بین برنامه ها برقرار می گردد.

- ویژگی
  - طراحی و پیاده سازی صرفا با هدف ارتباط مستیقم بین سیستم های موردنظر .
  - نوشتن کد سفارشی برای استخراج داده ، پردازش قواعد کسب و کار و بارگذاری داده
  - فرمت سفارشی داده
  - پروتکل انتقال ناسازگار
  - قابل استفاده برای یکپارچه سازی بلادرنگ و batch
- مزایا
  - عدم نیاز به سرمایه گذاری بالا بر روی ابزار
  - عدم نیاز به یادگیری مهارت های جدید توسط پیاده کنندگان
  - عدم اختصاص زمان زیاد برای ایجاد ، توسعه و بکارگیری استراتژی یکپارچه سازی سازمانی

۲) مبتنی بر پیام :

سیستم های مبداء پیام های سازمانی رابرای یک گذرگاه مشترک منتشر می کنند .برنامه ها به عصویت پیام های مرتبط درآمده و با استناد به آنها عمل می کنند .

اگرچه اشیای توزیع شده، بسیاری از معایب روشRPC رابرطرف کردند ولی دو عیب آن را از قلم انداختند:

۱) RPCو اشیای توزیع شده، هر دو از روش ارتباط همگام استفاده می کنند.

- پیوند بین سرور و کلاینت، یک پیوند قوی است.
- مقیاس پذیری پایین است.

۲) در روشهای RPC و اشیای توزیع شده، ارتباط بین برنامه ها کابردی، قابل اعتماد نیست .

- تضمینی وجود ندارد که پیام و مقدار بازگشتی به مقصد تحویل داده شود.
- ممکن است کلاینت در مواردی مانند توقف در اتصال شبکه یا زمانی که سرور فعال نیست، در عملیات خود دچار وقفه شوند.

راه حل این مشکل پیام رسانی ناهمگام است .

- ویژگی
  - به عنوان یک کارگزار بین برنامه ها عمل می کند .
  - ذخیره و فوروارد کردن پیام ها
  - ارایه ی تضمین شده و تقریبا بلادرنگ
  - ارایه محیطی برای تعریف قواعد
- مزایا
  - سیستم ها با یکدیگر یکپارچه می شوند ولی جفت نمی شوند
  - قواعدکسب و کار و موتور تبدیل در یک کارگزار پیام متمرکز می شود
  - امکان یکپارچه سازی بلادرنگ که کاهش تاخیر را به دنبال دارد
  - حل دو به توان n مساله . به موازات افزایش سیستم ها ، تلاش لازم جهت یکپارچه سازی به صورت خطی رشد خواهد کرد.

۳) سرویس های وب :

APIچیست؟(Application Programming Interface)

APIرابط کاربردی برنامه‌نویسی) مجموعه‌ای از تعاریف، دستورالعمل‌ها، پروتکل‌ها و ابزارها برای دسترسی از یک نرم‌افزار به نرم‌افزاری دیگر است که هر دو مبتنی بر ویندوز می‌باشد. رابط کاربردی برنامه‌نویسی مجموعه‌ای خاص از قوانین (کدها) است که به برنامه‌های نرم‌افزاری امکان برقراری ارتباط را می‌دهد API .به عنوان رابط بین برنامه‌های مختلف مانند یک مسیر، تعامل میان انسان و کامپیوتر را تسهیل می‌نماید.

APIها درحالی‌که از بخش‌های مختلف برنامه محافظت کرده و امکان برقراری ارتباط را فراهم می‌کنند به‌گونه‌ای ساخته‌شده‌اند که عملکرد اطلاعات انتخاب‌شده را نمایان می‌کنند. یک رابط برنامه‌نویسی کاربردی خوب باعث می‌شود که برنامه‌نویس راحت‌تر بتواند برنامه خود را توسعه بدهد.

به زبان ساده‌تر API قطعه کد واسطی است که نرم‌افزار سرویس‌دهنده را به نرم‌افزار سرویس‌گیرنده متصل کرده تا بتواند از سرویس‌های ارائه‌شده توسط آن استفاده نماید.

کاربردهای API چیست؟(API Usage)

قطعه کدها به همراه قوانین مشخصی برایAPI ها تولید می‌شوند که با جایگزین نمودن آن‌ها در نرم‌افزار سرویس‌گیرنده، ارتباط مابین دو نرم‌افزار برقرار می‌شود. این پل ارتباطی باعث می‌شود، سرویس‌گیرنده توسعه‌یافته و بتواند از خدمات ارائه‌شده نرم‌افزار سرویس‌دهنده استفاده نماید، همچنین از طریق API امکان دسترسی و ویرایش اطلاعات برای کاربران فراهم خواهد شد.

به‌عنوان‌مثال، زمانی که شما بخواهید متنی از یک نرم‌افزار را به نرم‌افزار دیگری کپی کرده و الصاق نمایید، API اینجا به کمک آمده و با برقراری ارتباط بین دو قطعه نرم‌افزار نقش خود را ایفا می‌کند. رابط‌های برنامه کاربردی امکان دسترسی یک نرم‌افزار به اطلاعات شما و تغییر آن‌ها را می‌دهد به این صورت که شما می‌توانید با بهره‌گیری از آن‌ها نرم‌افزار سایت خود را به سیستم یا نرم‌افزار سمت میزبان متصل کنید .پلتفرم‌هایی (platform)همچون ویندوز با فراهم آوردن قابلیت برقراری ارتباط بین قطعه کدهای مختلف توسط API ها این امکان را به وجود آورنده‌اند تا برنامه نویسان بتوانند برای ارتباط با سیستم‌عامل‌ها راهکاری داشته باشند .امروزه API در وب‌سایت‌ها هم استفاده می‌شوند.

نکات قابل‌توجه در استفاده از API ها

رابط‌های برنامه کاربردی، داده‌های ساخت‌یافته‌ای را ارائه می‌دهند که ارتباط بین نرم‌افزارها در قالب یک فرمت استاندارد را صورت می‌دهد.نرم‌افزار سرویس‌گیرنده با ارسال یک درخواست در قالب کدهای API نیاز خود را به نرم‌افزار سرویس‌دهنده ارسال می‌نمایند و پاسخ این درخواست از همین طریق استاندارد بازگشته و نیاز را مرتفع نموده و باعث توسعه آن می‌گردد.

رابط‌های برنامه کاربردی حاوی) API key شناسه‌ی (API هستندAPI key .ها ماهیت نرم‌افزار سرویس‌گیرنده را مشخص می‌نماید و نشان می‌دهد که چه نرم‌افزاری درخواست را ارسال نموده است و سپس بر اساس پروتکل‌های موجود پاسخ درخواست شما ارسال می‌گردد، همچنین در این بخش محدودیت زمانی و تعداد ارسال درخواست در بازه‌ی زمانی توسط نرم‌افزار سرویس‌دهنده به نرم‌افزار سرویس‌گیرنده اعلام می‌گردد ، و اگر تعداد درخواست بیش از استاندارد تعیین شده باشد از طرف نرم‌افزار سرویس‌دهنده نادیده (ignore)گرفته می‌شود.

از مزایای تولید API ها می‌توان این نکته را متذکر شد که اگر شما در حوزه‌ی کاری خود اولین مجموعه‌ای باشید که برای استفاده از خدماتی خاص API مخصوصی را تولید و در بازار عرضه نموده‌اید، رقبا و سایر مجموعه‌های ارائه‌دهنده خدمات برای توسعه کار خود مجبور به استفاده از استاندارد API شما می‌باشند؛ همچنین زمانی که به کاربران خود دسترسی استفاده از API ها را می‌دهید ، می‌توانید خلاقیت در استفاده ازAPI ها را در برنامه‌های مختلف بیابید که برای یافتن این راهکارهای نوین در ارتباط با نرم‌افزارهای مختلف زمانی را صرف نکرده‌اید.

APIتحت وب(Web API)

رابط‌های برنامه کاربردی اطلاعات را برای کاربران به شکلی خاص فرمت کرده و به‌صورت دینامیکی ارائه می‌دهد . APIضمن دریافت اطلاعات آن‌ها را ارسال کرده و اطلاعات را بر مبنای درخواست Client برگردانده و کارهای مختلفی انجام می‌دهد، تمامی این موارد در بستر وب و مبتنی بر اینترنت بدون محدودیت در استفاده از نرم‌افزارهای مبتنی بر وب نیز رخ می‌دهد، که توضیحات کامل Web Serviceدر بخشی مجزا ارائه گردیده است.

مسئله عدم تجانس بین کاربردهای یک سازمان:

- عدم تجانس میان افزار (سرورها)
- عدم تطابق پروتکل (... , http)
- تنوع فرمت داده‌ها
- تنوع معرفی واسطها (... ,CORBA, Java RMI)
- نداشتن مکان مشترک جهت جست و جوی سرویس‌ها

راه حل : استفاده از استانداردها

استاندارد: مجموعه‌ای از قوانین و مشخصات که توسعه پیدا می‌کنند و پذیرفته می‌شوند و به جزییات پیاده‌سازی وابسته نیستند.

مثال: XML, SOAP, WSDL, UDDI, WS-I Basic Profile

۱) XML(Extensible Markup Language) : زبان ارتباطات مشترک ، فرمتی مستقل از میان افزار برای تبادل داده

۲) SOAP(Simple Object Access Protocol) : فرمت مشترک برای تبادل پیام‌ها بین کاربردها (سرویس دهنده و گیرنده)

۳) WSDL(Web Services Description Language) : فرمت مشخصات سرویس مشترک، زبان توصیف وب سرویس‌ها و مستقل از زبان و پلت‌فرم جهت تعیین واسط یک سرویس

۴) UDDI(Universal Description, Discovery and Integration) : ابزاری مشترک برای جست و جوی سرویس، روشی استاندارد برای ثبت، حذف و جست و جوی سرویس‌ها

مثال خرید بلیط قطار

امکان یکپارچه سازی توانمندی ها از طریق XML بر روی یک پروتکل باز نظیر . SOAP سایر سیستم ها در صورت نیاز می توانند از این سرویس استفاده نمایند . ورودی و خروجی به سرویس های وب به صورت XML است .

- ویژگی
  - زبان مشترک برای مبادله بین سیستم ها ی ناهمگن
  - مبتنی بر فناوری های استاندارد اینترنت
  - خودتشریح
  - حمایت از کشف پویا و یکپارچه سازی
  - تطبیق سرویس ها درون یک مدل کلی معماری
  - حمایت توسط تولید کنندگان مهم و اصلی
- مزایا
  - حل مسائل به روشی مشابه با EAI
  - نیاز به ابزارهای ارزان قیمت یکپارچه سازی
  - استفاده از سیستم عامل یکپارچه سازی اختصاصی

زبان نشانه‌گذاری گسترش‌پذیرXML(eXtensible Markup Language)

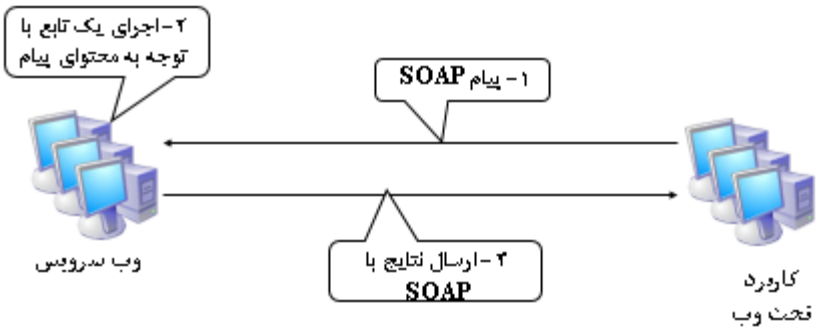
- در سال ۹۷ استاندارد شد
- به شرکت خاصی تعلق ندارد
- جبران ضعف HTML
- جدا بودن محتوا از ظاهر
- استفاده از تگ مشابه HTML
- تگ جهت تعیین نوع داده نه نمایش آن (مانند HTML)
- برخلاف HTML، تگ‌ها ثابت نیستند
- هر کاربرد خودش تصمیم می گیرد که چگونه اطلاعات را نمایش دهد
- XML parser برای خواندن اطلاعات
- هر تکنولوژی می‌تواند به استفاده از آن بپردازد
- مهم‌ترین استاندارد در وب‌سرویس‌ها
- اسناد XML ابزاری جهت انتقال اطلاعات بین سرویس دهنده و گیرنده
- مبنایی برای WSDL جهت معرفی واسط
- مبنای پروتکل SOAP جهت دسترسی به وب سرویس‌ها
- UDDI که برای انتشار و یافتن سرویس هست نیز بر اساس XML هست

یک مثال ساده از سند XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

عنصر فوقانی: note (هر عنصر شامل داده، ویژگی و تعدادی عنصر دیگر)

SOAP(ارتباط کاربردها و عناصر)



هدف : ارسال داده بین سیستم های روی شبکه

نکاتی پیرامون پیام SOAP :

- XML به تنهایی برای ارتباط کافی نیست.
- SOAP : ارائه فرمت پیام مشترک برای سرویس‌دهنده و سرویس‌گیرنده
- SOAP از XML استفاده می‌کند
- مستقل از زبان برنامه‌نویسی و پلت‌فرم
- پیام SOAP یک سند XML هست
  - هدف: تفسیر درست توسط گیرنده
- دریافت توسط سرورهای SOAP

عناصر پیام

مزیت های SOAP

زبان WSDL

- مبتنی بر فرمت XML
- توانایی تعریف واسط و توصیف مشخصات وب سرویس‌ها توسط خودشان
  - اطلاعات (پارامترهای) مورد نیاز وب سرویس
  - اطلاعات (پارامترهای) خروجی وب سرویس
- طراحی برای درک آن توسط ماشین
- در ابتدا هر یک از توسعه‌دهندگان، استاندارد خودشان را داشتند
- پس از یکسان شدن استاندارد IBM و مایکروسافت، WSDL به وجود آمد
- ارائه WSDL توسط این دو کمپانی و استاندارد شدن آن

هر وب سرویس روی اینترنت یک WSDL دارد.

این فایل توصیف کننده وب سرویس:

- مشخصات
- مکان
- نحوه استفاده (با استفاده از پروتکل‌های مختلف)

در تئوری، هر برنامه کاربردی تحت وب با استفاده از WSDL به جستجوی وب سرویس دلخواه خود گردیده و از آن استفاده می‌کند.

قسمت های سند :WSDL

UDDI:

UDDI می‌تواند:

- به طور خصوصی نگهداری شود
- در مکان‌های عمومی در اینترنت قرار داده شود

بزرگ‌ترین و مهم‌ترین پایگاه در اینترنت

UDDI Business Registry که توسط چهار شرکت نگهداری می‌شود:

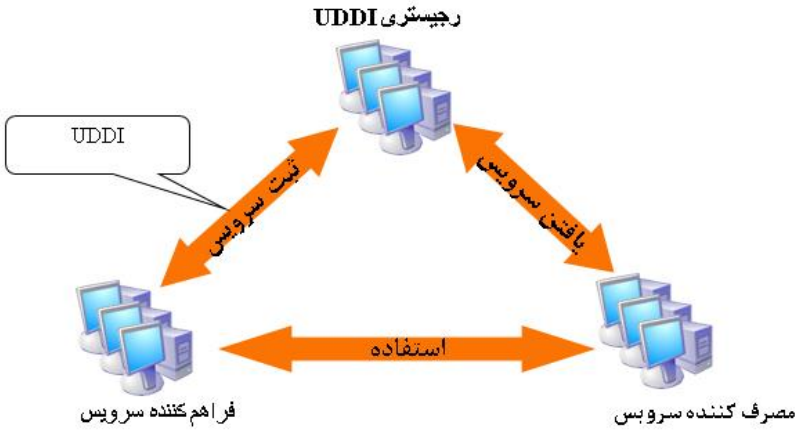
- مایکروسافت
- IBM
- SAP
- HP

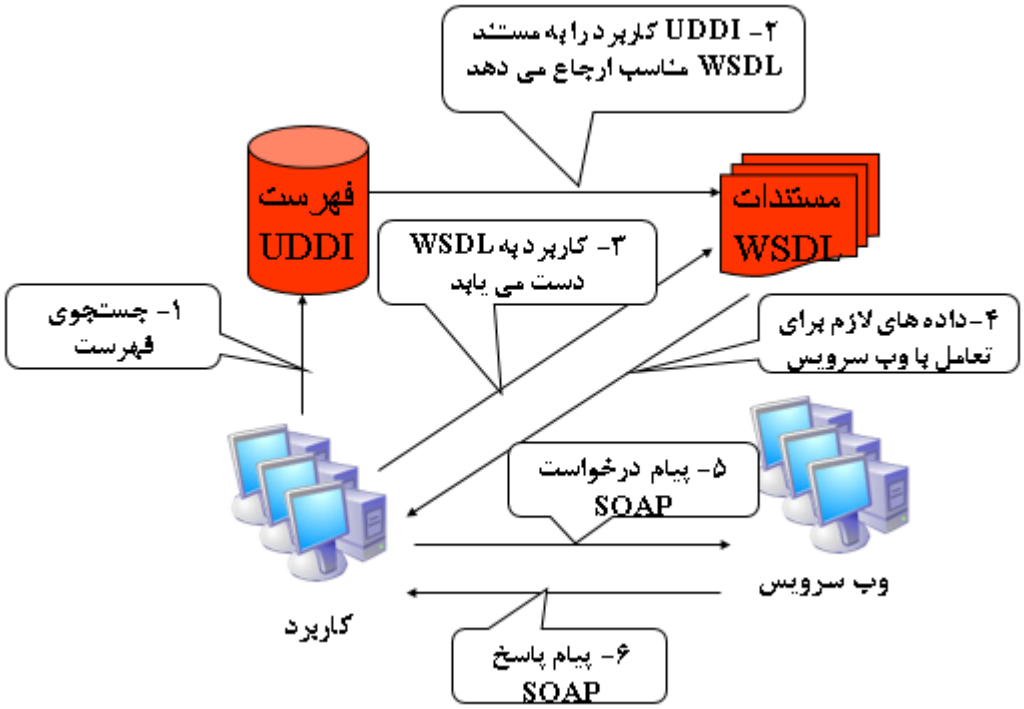
این اطلاعات شامل:

- اطلاعات تماس شرکت‌ها و توضیحات متنی آن شرکت‌ها

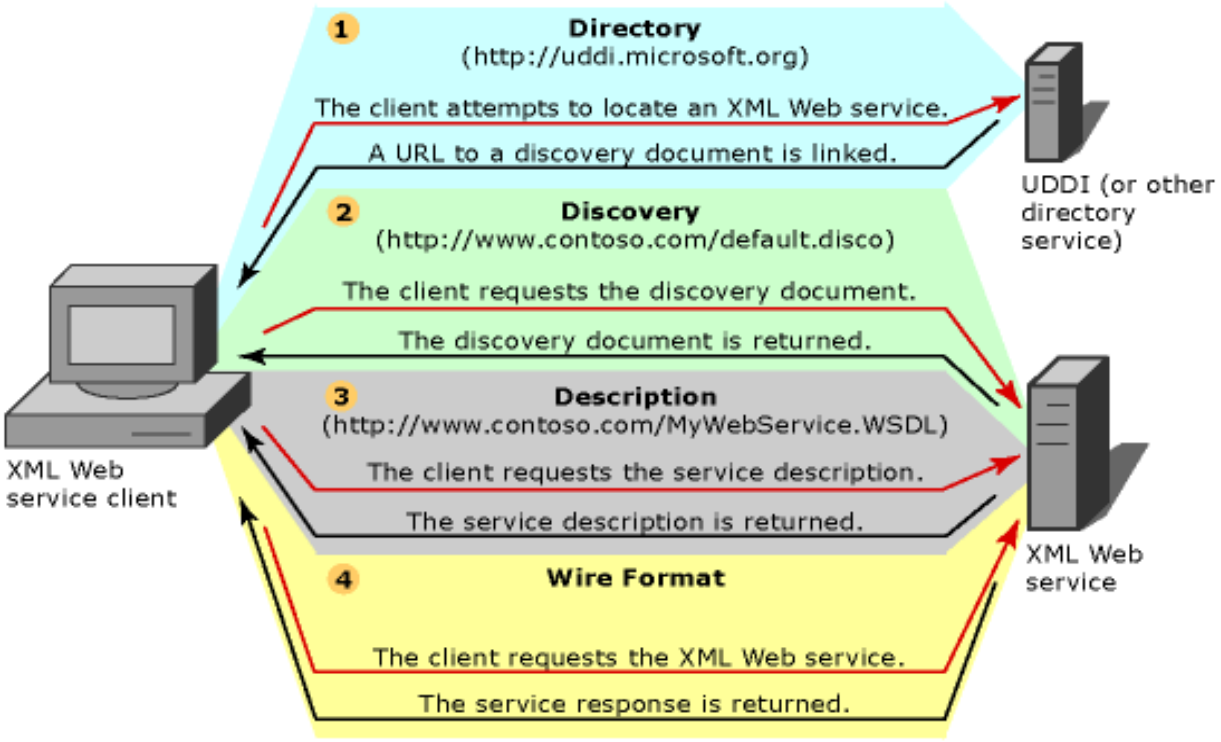
- اطلاعات طبقه‌بندی شده شرکت‌ها و اطلاعات درباره توانایی وب سرویس‌های آنها
- اطلاعات تکنیکی درباره سرویس‌های آنها و نحوه پردازش اطلاعات

رجیستری UDDI





وب سرویس در یک نگاه



مثال : یک وب سرویس

۴) ابزارهای ETL (برگرفته شده از Extract, Transform, Load) :

مجموعه ای استاندارد از ابزارها و فرآیندها که از آنها برای استخراج ، تبدیل و بارگذاری حجم بالائی از داده بین سیستم ها استفاده می گردد . استفاده از این نوع ابزارها در ایجاد یک انبار داده بسیار متداول است .

- ویژگی
  - ارایه ابزارهایی برای : تمیزکاری داده ، تصحیح اشتباهات تایپی ، حل موارد متناقض ، حذف عناصر ، تفسیر عناصر
  - قادر به ترکیب منابع داده : بر اساس مقادیر کلیدها ، تطبیق فازی براساس خصلت های غیرکلید ، مقایسه متنی با جداول مرجع
  - قادر به شناسایی و حذف موارد تکراری
  - قادر به ایجاد کلیدهای جایگزین : سیستم های عملیاتی و انبار داده دارای فرضیات و نیاز های داده یی مختلف می باشند . بنابراین ، انبارداده نیازمند مجموعه کلیدهای اولیه خود است .
  - قادر به ایجاد تجمیع به منظور افزایش کارآیی query در انبار داده و انبارک های داده می باشند .
  - بارگذاری و ایندکسینگ: برای انبار داده با حجم بالا نیاز به بارگذاری حجم بالائی از داده است .
- مزایا
  - دارای کارآیی لازم برای جابجایی حجم بالائی از داده در یک بازه زمانی کوتاه
  - بکارگیری یک روش یکدست برای تبدیل داده
  - قادر به ارایه و یا یکپارچگی با متادیتا برای مدل داده سازمانی

شکل ۵ مشخصات گزینه های مختلف یکپارچه سازی برنامه ها را نشان می دهد.

شکل ۳ : تعداد ارتباطات مورد نیاز جهت اتصال و یا یکپارچه سازی شش برنامه با روش نقطه به نقطه

در اوایل حضور بانک های اطلاعاتی برخی از کارشناسان این رویا را داشتند که بتوانند بانک های اطلاعاتی را در سازمان خود بگونه ای ایجاد نمایند که داده تکراری حذف گردد . "ضبط داده یک مرتبه ، ذخیره داده یک مرتبه و استفاده از داده به دفعات " شعاری بود که در گذشته بسیاری به دنبال تحقق آن بودند . پردازش های غیرمتمرکز ، پکیج های نرم افزاری و اعتقاد به بهره برداری از سیستم های قدیمی علی رغم بروز تغییرات عمده در سازمان و بالطبع آن فرآیندهای مربوطه ، پایانی بود بر این رویا .

فقدان یک رویکرد جامع به فرآیندهای یک سازمان باعث گردید که هر روز شاهد تولد یک سیستم جدید باشیم. سیستم هایی که هر یک از دید خود به دنبال اتوماتیک کردن بخشی از فرآیندهای یک سازمان بودند و برای حل این مساله ،مدل داده مختص به خود را ایجاد می کردند. همین موضوع باعث شد ، علی رغم این که تمامی سیستم ها زیر یک سقف زندگی می کردند ولی کاملا با هم غریبه بودند و امکان به اشتراک گذاشتن داده و یا برخی از قابلیت ها را بین خود نداشتند. از طرف دیگر ، رهبران و مدیران کسب و کار به منظور تصمیم گیری مناسب و به موقع نیاز به اطلاعاتی یکدست و منسجم در رابطه با موجودیت های اساسی کسب و کار داشتند که در خوشبینانه ترین حالت این اطلاعات در سیستم های مختلفی توزیع شده بود . در بسیاری از موارد این احتمال وجود داشت که دارای اطلاعات متناقضی در ارتباط با یک موجودیت نظیر مشتری باشیم که این موضوع باعث بروز مسائل و مشکلات زیادی در بخش های مختلف یک سازمان می گردید .

نیاز به یکپارچه سازی داده یکی از چالش های اساسی اکثر بنگاه های کسب و کار در عصر حاضر است . امروزه سازمان ها ده ها تا صدها میلیون دلار را صرف یکپارچه سازی داده برای سیستم های تراکنشی و یا سیستم های تحلیلی خود) نظیر سیستم هوش کسب و کار) می کنند . ما امروز به سازمان هایی برخورد می کنیم که دارای صدها تا هزاران ارتباط نقطه به نقطه غیرمستندی می باشند که جهت انجام وظایف خود علاوه بر امکانات سخت افزاری متعدد ، به نیروی انسانی لازم نیز جهت حمایت از این فرآیندها نیاز دارند . وجود ده ها و یا صدها برنامه در یک سازمان که هر یک به دنبال اجرای بخشی از فرآیندهای یک سازمان می باشند ولی بین آنها هیچگونه ارتباط و یا یکپارچگی وجود ندارد ، قدرت عملیاتی و تصمیم گیری یک سازمان را با چالش های جدی مواجه می سازد .چالش هایی که می تواند منجر به خارج شدن یک سازمان از مدار رقابت و در نهایت حذف از دایره وجود گردد.

سوکت نویسی :

مراحل ایجاد یک ارتباط :

۱) سرور سوکتی را تعریف می کند
۲) سرور سوکت را به یک IP که IPخودش است اختصاص می دهد و یک پورت Bind میکند .
۳) سرور به پورت گوش می دهد
۴) کلاینت سوکتی را تعریف میکندو IP و پورت سرور را به آن اختصاص می دهد .
۵) کلاینت درخواست اتصال یا کانکت شدن به سرور را می دهد
۶) سرور درخواست کلاینت را دریافت و آن را می پذیرد
۷) کلاینت اطلاعاتی را ارسال می کند
۸) سرور اطلاعات را می گیرد
۹) سرور اطلاعات را ارسال میکند و کلاینت آن را می گیرد
۱۰)سرور بسته می شود
۱۱) کلانیت بسته می شود

فضای نام های مورداستفاده در سوکت نویسی به زبان سی شارپ:

System.Net
System.Net.Socket
System.Text

توابع مهم مورد استفاده:

ایجاد سوکت :

```
Socket sktListener=new socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
```

- sktListener نامی است که برای سوکت در نظر گرفته ایم .(دلخواه)
- بخش اول به معنی این است که از شبکه ای استفاده میکنیم که دارای IP ورژن ۴ است.

- بخش دوم به این معنی است که میخواهیم داده ها را بهصورت استریم تبادل کنیم.
- بخش سوم نوع پروتکل مورد استفاده را نشان میدهد.

نحوه اختصاص آی پی و پورت به سوکت :

```
IPEndPoint ipLocal = new IPEndPoint(IPAddress.Any, 1800);
sktListener.Bind(ipLocal);
```

تابع IPEndPoint آدرس آی پی و پورت را نگهداری کرده و تابع Bind این آدرس ها را به سوکت sktListener که قبلا تعریف کرده ایم اختصاص میدهد.

نکته : در دستور فوق چون برای سرور نوشته ایم نیازی به اختصاص آی پی نبود و فقط پورت ۱۸۰۰ را باز کردیم.

جهت گوش دادن به پورت از دستور زیر استفاده میشود :

```
sktListener.Listen(100);
```

اکنون ماشین سرور درحال گوش دادن به پورت ۱۸۰۰ میباشد و عدد ۱۰۰ نمایانگر تعداد اتصالاتی است که میتوانند در صف قرار گیرند..

حال باید درخواست کانکت شدن کلاینت را بپذیریم:

```
sktListener = sktListener.Accept();
```

حال باید داده ها را دریافت کنیم :

نکته : در سوکت پروگرمینگ ، داده ها به صورت آرایه ای از بایت ها منتقل می شوند .برای ارسال رشته های یونیکد و .... بایست آنها را کد گذاری کنیم . برای کد گذاری یا کد گشایی از کلاس System.Text استفاده میکنیم.

مثال کد گشایی و کدگذاری :

```
byte[] byt = Encoding.ASCII.GetBytes("salam");
```

این دستور رشته ی salam را با فرمت اسکی به آرایه ای از بایت ها تبدیل میکند.

```
string str = Encoding.ASCII.GetString(byt);
```

این دستور آرایه ی byt را رمزگشایی میکند.

نکته : عمل رمزگذاری هنگام ارسال داده ها و عمل رمزگشایی هنگام دریافت داده ها انجام میشود.

اکنون میخواهیم داده ها را دریافت کنیم پس عمل رمز گشایی را انجام میدهیم:

```
byte[] buffer = new byte[500];
sktListener.Receive(buffer);
string Data = Encoding.ASCII.GetString(buffer);
```

دستور Receive(buffer) داده های ارسال شده را دریافت کرده و آنها را در متغیرbuffer ذخیره میکند.
دستور آخر عملیات رمزگشایی را انجام میدهد.
حال داده ها دریافت شده اند و میتوانیم انها را پردازش کنیم .(بعنوان مثال در صفحه نمایش نشان دهیم.) :

```
Console.WriteLine(Data);
```

کدهایی که تا به اینجا دیدیم برای ایجاد سوکت های همگام یا سنکرون بوده است. این سوکت ها در برنامه های ویندوز و کلا سیستم های مالتی تسک کاربردی ندارند. چرا که زمانی که از متد accept استفاده نموده ایم، در این حالت برنامه تا رسیدن یک سوکت به آن قفل شده و قادر به انجام کاری نیست.

در سوکت های آسنکرون از متدهای آسنکرون برای گوش دادن ، ارسال ، دریافت و ... استفاده می کنیم. در این آموزش ، یک برنامه سمت سرور به صورت آسنکرون طراحی می کنیم که قادر به گوش دادن به یک کلاینت است.

برای پیاده سازی سوکت های ناهمگام باید از delegate ها استفاده کنیم .delegate یک اشاره گر به تابع است ، در سوکت های آسنکرون از delegateی به نام AsyncCallback استفاده میکنیم.

نکته : متدهای آسنکرون با پیشوند Beginو End شروع میشوند.

برای نوشتن یک سوکت آسنکرون مانند مراحل سنکرون باید عمل کنیم با این تفاوت که شکل بعضی دستورات تغییر میکند :

```
Socket Mainlistener = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

IPEndPoint server = new IPEndPoint(IPAddress.Any, 1800);
Mainlistener.Bind(server);

AsyncCallback callBackMethod = new AsyncCallback(AcceptCallback);

Mainlistener.Listen(4);
Mainlistener.BeginAccept(AcceptCallback,Mainlistener);
```

در این مثال ، مشخص کرده ایم که سوکت شروع به عمل گوش دادن و انتظار کند و سپس به محض کانکت شدن یک کلاینت به کامپیوتر ما ، تابع AcceptCallback اجرا گردد و به اموری که تعیین می کنیم رسیدگی کند.

تابع AcceptCallback بایستی به اینصورت نوشته شود :

```
private void AcceptCallback(IAsyncResult ar)
{

Socket temp = ((Socket)ar.AsyncState);
Socket worker = temp.EndAccept(ar);
AsyncCallback ReceiveMethod = new AsyncCallback(ReceiveCallBack);
worker.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None,new
ReceiveMethod, worker);

}
```

براي دريافت اطلاعات به صورت آسنكرون، از متد ()BeginRecieve استفاده كرديم.

اکنون باید متد ()RecieveCallBack را بنویسیم :

```
private void ReceiveCallBack(IAsyncResult ar)
{
    Socket worker = ((Socket)ar.AsyncState);
    int bytesReceived = worker.EndReceive(ar);
     string str = System.Text.UTF8Encoding.UTF8.GetString(buffer);
}
```

ابتدا اطلاعات وضعیت را بدست آوردیم . سپس به گوش دادن موقت خاتمه میدهیم.و بعد از آن باید اطلاعات دریافت شده را کدگشایی کنیم.

یک مثال ساده : در این مثال به پیام های رسیده از پورت ۱۳۰۰۰ از آدرس LocalHost گوش میکند و ارتباط با سایر کلاینت ها را برقرار میکند :

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;
class MyTcpListener
{
public static void Main()
{
TcpListener server=null;
try
{
// Set the TcpListener on port 13000.
Int32 port = 13000;
IPAddress localAddr = IPAddress.Parse("127.0.0.1");
// TcpListener server = new TcpListener(port);
server = new TcpListener(localAddr, port);
// Start listening for client requests.
server.Start();
// Buffer for reading data
Byte[] bytes = new Byte[256];
String data = null;
// Enter the listening loop.
while(true)
{
Console.Write("Waiting for a connection... ");
// Perform a blocking call to accept requests.
// You could also user server.AcceptSocket() here.
TcpClient client = server.AcceptTcpClient();
```

```
Console.WriteLine("Connected!");
data = null;
// Get a stream object for reading and writing
NetworkStream stream = client.GetStream();
int i;
// Loop to receive all the data sent by the client.
while((i = stream.Read(bytes, 0, bytes.Length))!=0)
{
// Translate data bytes to a ASCII string.
data = System.Text.Encoding.ASCII.GetString(bytes, 0, i);
Console.WriteLine("Received: {0}", data);
// Process the data sent by the client.
data = data.ToUpper();
byte[] msg = System.Text.Encoding.ASCII.GetBytes(data);
// Send back a response.
stream.Write(msg, 0, msg.Length);
Console.WriteLine("Sent: {0}", data);
}
// Shutdown and end connection
client.Close(

}
}
catch(SocketException e)
{
Console.WriteLine("SocketException: {0}", e);
}
finally
{
// Stop listening for new clients.
server.Stop();
}
Console.WriteLine("\nHit enter to continue...");
Console.Read();
}
```

کدهای کامل سمت سرور -سنکرون :

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
static Socket sktListener;
 static void Main(string[] args)
{
sktListener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        IPEndPoint ipLocal = new IPEndPoint(IPAddress.Any, 1800);
        sktListener.Bind(ipLocal);
        sktListener.Listen(100);
        sktListener = sktListener.Accept();
        byte[] buffer = new byte[500];
sktListener.Receive(buffer);
sktListener.close();
}
```

کدهای کامل سمت کلاینت -سنکرون:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
```

```
static void Main(string[] args)

{

Socket sktClient = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

        sktClient.Connect("127.0.0.1″, 1800);

        string str = "Hello Server….";

        byte[] buffer = Encoding.ASCII.GetBytes(str);

        sktClient.Send(buffer);

        sktClient.Close();

}
```

سمت سرور – آسنکرون:

```
byte[] buffer = new byte[1024];

 public Form1()

{

        InitializeComponent();

 Socket Mainlistener = new Socket(AddressFamily.InterNetwork,

        SocketType.Stream, ProtocolType.Tcp);

  IPEndPoint server = new IPEndPoint(IPAddress.Any, 1800);

 Mainlistener.Bind(server);

  AsyncCallback callBackMethod = new AsyncCallback(AcceptCallback);

  Mainlistener.Listen(4);

 Mainlistener.BeginAccept(AcceptCallback,Mainlistener);

}
```

```
 private void AcceptCallback(IAsyncResult ar)
{
        Socket temp = ((Socket)ar.AsyncState);
        Socket worker = temp.EndAccept(ar);
        AsyncCallback ReceiveMethod = new AsyncCallback(ReceiveCallBack);
worker.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, ReceiveMethod, worker);
}
 private void ReceiveCallBack(IAsyncResult ar)
{
        Socket worker = ((Socket)ar.AsyncState);
        int bytesReceived = worker.EndReceive(ar);
        string str = System.Text.UTF8Encoding.UTF8.GetString(buffer);
        ShowInfo(str);
        AsyncCallback ReceiveMethod = new  AsyncCallback(ReceiveCallBack);
        worker.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, ReceiveMethod,
worker);
}
 delegate void ShowInfoCallback(string text);
private void ShowInfo(string text)
{
        if (this.txtMain.InvokeRequired)
        {
            ShowInfoCallback d = new ShowInfoCallback(ShowInfo);
```

```
            this.Invoke(d, new object[] { text });
        }
        else
        {
            txtMain.Text+=text+"\n";
        }
}
```

منابع :

1) www.sbu.ac.ir (آزمایشگاه مرجع معماری سازمانی)

2) www.zachman.blogfa.com(معماری سازمانی)

3) https://ieaf.ir/ (چارچوب معماری سازمانی ایران)

۴) اصول، مبانی و روش های معماری سازمانی سروپس گرا ، امیر مهجوریان – فریدون شمس