**Web-ICONIX:**

**روشي جديد براي مدل سازي**

**برنامه هاي**

**كاربردي وب**

**بوسيله**

**UML**

**فهرست مطالب**

**1- مقدمه 4**

**2- برنامه های کاربردی وب** 5

2-1- تعريف برنامه های کاربردی وب 5

2-2- ويژگيها 5

2-3- تفاوت برنامه های کاربردی وب و سايت وب 6

2-4- معماری برنامه های کاربردی وب 6

2-5- اجزای برنامه های کاربردی وب 7

2-6- پروتکل ارتباطی وب (HTTP) 7

2-7- ساختار دهی به داده های وب با XML 10

2-8- ارتباطات توزيع شده در وب 12

2-8-1-RMI (Remote Method Invocation) 12

2-8-2- CORBA (Common Object Request Broker Architecture Protocol) 14

2-8-3- SOAP (Simple Object Access) 14

2-8-4- مقايسه روش‌هاي توزيع اشياء روي وب 16

2-9- ارتباط بين برنامه های کاربردی وب 16

2-9-1- WSDL (Web Services Description Language) 16

2-9-2- UDDI (Universal Description , Discovery and Integration) 17

2-9-3- مقايسه و ارزيابي 27

**3- مرور برخی از پژوهش های پيشين 19**

3-1- HDM (Hypermedia Design Model) 19

3-2- RMM (Relationship Management Methodology) 19

3-3- OOHDM (Object Oriented Hypermedia Design Model) 20

3-4- مقايسه و ارزيابي 21

**4- معرفی مدل فرآيند جديد Web-ICONIX 21**

4-1- فرآيند RUP 21

4-1-1- فاز آغاز 22

4-1-2- فاز جزييات 23

4-1-3- فاز ساخت 24

4-1-4- فاز انتقال 25

4-1-5- تحليل و ارزيابي 25

4-2- ملحقات وبUML 26

4-2-1- تعريف UML 26

4-2-2- انواع دياگرام های UML 27

4-2-3- استفاده از UML برای مدل سازی برنامه های کاربردی وب 27

4-2-4- ملحقات وب کنالن 29

4-3- معرفی روش ICONIX 30

4-3-1- نقشه کلی ICONIX 30

4-4- گام های Web- ICONIX 31

4-4-1- فاز آغاز 31

4-4-2- فاز جزييات 32

4-4-3- فاز ساخت 33  
4-4-4- فاز انتقال 34

**5- ساخت يک نمونه برنامه کاربردی وب با روش جديد 35**

5-1- فاز آغاز 35

5-2- فاز جزييات 39

5-3- فاز ساخت 42

5-4- فاز انتقال 47

**6- ارزيابی** 47

منابع 49

# مقدمه

در چند سال اخير اصطلاح جديدي به فرهنگ لغات تكنولوژي اطلاعات اضافه شده: برنامه كاربردي‌ وب. بسياري از سيستم‌هاي نرم‌افزاري تجاري به دليل الزامات تازه يا براي بهره گيري از فرصت هاي جديد، در حال تبديل كامل يا بخشي به برنامه هاي كاربردي‌ وب هستند، از جمله در زمينه تجارت الكترونيك، كتابخانه هاي ديجيتالي، آموزش از راه دور، ادارات مجازي و.... رشد نمايي كاربردهاي وب، نياز به روشهايي براي مدل سازي و مستندسازي مراحل ساخت آنها را بيشتر مي كند. اين نياز در ابتدا محدود به داشتن روشي براي نشان دادن ارتباطات پيچيده بين صفحات وب بود، ولي با ظهور تكنولوژيهاي جديدتر و گسترش كاربرد معماري هاي چند لايه نرم افزارها، و نيز گسترش دامنه كاربردهاي برنامه هاي تحت وب، اين روشهاي اوليه ايجاد شده براي مدلسازي برنامه هاي كاربردي وب به سرعت ناكارامد شدند. پس از آن فرآيندهايي براي ساخت برنامه‌هاي كاربردي وب ارائه گرديد. مشكل اصلي استفاده از اين فرآيندها در اين است كه آنها هركدام چرخه حياتي متفاوت با نرم افزارهاي غيروبي براي برنامه هاي كاربردي وب ارائه مي‌كنند، كه علاوه برنياز به آموزش يك فرآيندجديد به تيم توليد نرم افزار، كار در پروژه هاي تركيبي (كه برنامه كاربردي وب جزيي از يك پروژه نرم افزاري كلان‌تر است) را با مشكل مواجه مي سازد.

در اين مقاله مدل فرآيندي جديد بنام Web-ICONIXبراي ساخت برنامه هاي كاربردي وب در ابعاد متوسط درچارچوب كلي تبيين شده در فرآيند ساخت نرم‌افزار RUP و بر پايه روش كلاسيك ICONIX ارائه مي گردد. اين مدل فرآيند مانند RUP تكراري[[1]](#footnote-2) و افزايشي[[2]](#footnote-3) است. محور كار در تحليل نيازها و طراحي نرم افزار در آن، مدلموردكاربرد[[3]](#footnote-4) است و براي مدل سازي از نمادهاي موجود در UML و نيز ملحقات پيش‌نهادي توسط جيم كنالن براي ابزار Rational Rose استفاده مي شود. به دليل حفظ مفاهيم و مراحل كلي شناخته شده RUP و ICONIX و ابزارهاي شناخته شده‌ي آنها، به كارگيري اين مدل فرآيند نسبت به روش‌هاي ديگري كه مراحل و نمودارها و ابزارهاي كاملا جديدي را براي ساخت برنامه هاي كاربردي وب به كار مي گيرند ساده تر است و آموزش آن و نيز سازگار كردن آن با روش هاي مورد استفاده براي ساخت بخش هاي غير وبي سيستم با هزينه و زمان كمتري انجام مي گيرد.

ابتدا در فصل دوم برخي از تعاريف برنامه هاي كاربردي وب ذكر شده ويژگيهاي آنها و تفاوت‌هاي يك برنامه كاربردي وب با يك وب سايت پويا بيان مي گردند. همچنين در اين بخش معماري عمومي برنامه‌هاي كاربردي وب، شيوه‌ي اصلي انتشار اشيا در برنامه‌هاي كاربردي وب و روش‌هاي ايجاد سيستم‌هاي توزيع شده تحت وب مورد بررسي كلي قرار مي‌گيرد. در فصل سوم برخي از پژوهش‌هاي پيشين در اين حيطه كه رويكردي مهندسي به ساخت برنامه‌هاي كاربردي وب دارند به اختصار معرفي مي‌شوند و به دلايل ايجاد آنها و اشكالات اصلي هريك اشاره مي‌گردد. فصل چهارم به معرفي مدل فرآيند جديدي با عنوان Web-ICONIX مي پردازد. در اين بخش ابتدا چارچوب كلي مدل فرآيند RUP و مدل هاي اصلي و نيز نقشه كلي مسير روش ICONIX و ملحقات وب كنالن براي رسم مدل تحليل/طراحي برنامه كاربردي وب معرفي مي‌شوند. سپس با تلفيق اين سه موضوع و برخي تغييرات لازم در آنها براي هماهنگ شدن با مراحل ساخت برنامه هاي كاربردي وب، فازهاي مدل فرآيند جديد به شكل گام به گام بيان مي‌گردند. در فصل پنجم مراحل ساخت يك نمونه برنامه كاربردي وب با مدف فرآيند جديد نشان داده مي‌شود. در پايان در فصل ششم ارزيابي و خطوط كلي كارهاي آينده مي‌آيد و در جدولي نتايج مقايسه مدل فرآيند Web-ICONIX با ساير روشها نشان داده مي شود.

1. **برنامه‌هاي كاربردي وب**

# 2-1- تعريف برنامه ‌هاي كاربردي وب

پس از شيوع “اپلت‌هاي جاوا” در صفحات وب، اين نظر ابراز شد كه برنامه كاربردي‌ وب به هر چيزي گفته مي‌شود كه از جاوا استفاده كند. ولي با گسترش تكنولوژي‌هاي استفاده شده در ايجاد نرم‌افزارهاي مبتني بر وب، به تدريج اين تعريف عموميت يافت كه به هر چيزي كه از سرويس‌دهنده وب استفاده كند يك برنامه كاربردي‌ وب مي‌گويند. در RUP يك برنامه كاربردي وب اينچنين تعريف شده: «هر برنامه كاربردي كه بخشي از كارايي‌اش با واسطه يك مرورگر وب حاصل شود. يك برنامه كاربردي وب، وب سايتي است كه ورودي كاربر را مي‌گيرد (يا به طور مستقيم و يا بر اساس حركت كاربر بين صفحات وب) و حالت داخلي يك سيستم تجاري را تغيير مي‌دهد» (مثلا در ساده‌ترين حالت چيزي را در يك پايگاه داده تغيير مي‌دهد). بر اساس اين تعريف، يك برنامه كاربردي‌ وب سيستمي نرم‌افزاري است كه حالت تجاري دارد، و كاربران نهايي آن در سطح وسيعي (بر روي وب) پراكنده‌اند.

#### 2-2- ويژگيها

بطور كلي برنامه هاي كاربردي وب سه خصلت اصلي دارند: داشتن كاربران پراكنده، استفاده از داده هاي پراكنده، و ساخته شدن از تركيب تكنولوژيهاي گوناگون. بر اساس اين خصلت هاي اصلي، مي توان برخي از ويژگي هاي برنامه هاي كاربردي وب را چنين بر شمرد:

* اهميت بيشتر طراحي گرافيكي: ظاهر برنامه هاي تحت وب در پذيرش آنها نقش مهمي دارد. در ايجاد يك واسط كاربر مناسب مجموعه اي از افراد شامل بازارياب ها، طراحان خلاق و مديران تجاري نقش دارند.
* داشتن كاربران ناشناس، كه ممكن است هيچ آموزشي براي استفاده از برنامه نديده باشند.
* حساسيت بيشتر تهديدات امنيتي:با قرار گرفتن برنامه روي وب، تهديدات ناشناخته بيشتري امنيت سرويس ها و اطلاعات را تهديد مي كند.
* دسترسي به شيوه هاي مختلف: يك سايت وب ممكن است به شيوه هاي مختلف مورد بازديد قرار گيرد (مانند دسترسي فقط-متن[[4]](#footnote-5)، دسترسي از طريق گوشي موبايل، دسترسي صوتي، دسترسي از طريق ابزارهاي ترجمه برخط[[5]](#footnote-6) و يا از روي سكوهاي مختلف كامپيوترهاي شخصي مانند ويندوز، لينوكس، اپل و...)
* كاربر هميشه از وجود سايت مطلع نيست: سايت بايد به شكلي ساخته شود كه براي كاربر يافتن آن از طريق اسم دامنه يا موتور جستجو آسان باشد.
* بار غير قابل پيش بيني: توجه بيشتر نسبت به مقياس پذيري[[6]](#footnote-7) سيستم را مي طلبد
* نرم افزار ذاتا توزيع شده است، به طوري كه كنترل كمي روي تنظيمات محيط اجرا در طرف كاربر داريم.
* ارتباطات ناگزير با ساير برنامه هاي تحت وب: بسياري اوقات داده هاي مورد نياز برنامه هاي تحت وب از ساير سايت ها تامين مي شوند. همچنين گاهي ارتباط كاركردي بين اين نرم افزارها وجود دارد

# 2-3- تفاوت برنامه ‌هاي كاربردي وب و سايت وب

تفاوت يك برنامه كاربردي‌ وب و يك وب سايت در كاربرد آنهاست. برنامه كاربردي‌ وب يك لايه منطق تجاري را پياده سازي مي‌كند و استفاده از آن، حالت دروني اين لايه را تغيير مي‌دهد. اين مساله از آن جهت مهم است كه نقطه تمركز مدل‌سازي را مشخص مي‌كند. چون برنامه كاربردي‌ وب داراي يك لايه‌ي منطق تجاري است، مهم‌ترين مدل‌هاي سيستم بر روي اين لايه و حالت آن متمركز مي‌شوند، نه بر روي جزييات نمايش صفحات. نمايش در جاي خود مهم است (اگر به آن توجه نشود هيچكس نمي‌تواند با سيستم درست كار كند) ولي بايد بين روابط تجاري و جزييات نمايش در هنگام مدل‌سازي فاصله روشني را حفظ كرد. اگر مسائل مربوط به نمايش مهم يا پيچيده‌اند، بايد آنها را هم به شكلي مدل‌سازي كرد، ولي نه به عنوان جزيي از مدل منطق تجاري سيستم. بعلاوه منابعي كه روي نمايش كار مي‌كنند، بيشتر هنري هستند و كمتر با پياده‌سازي قوانين تجاري درگيرند. همچنين يك برنامه كاربردي وب با يك وب سايت معمولي سه تفاوت مهم دارد :

* كاربران فقط بين صفحات حركت نمي كنند، بلكه علاوه بر آن تراكنش هايي نيز انجام مي دهند.
* همچنان كه برنامه نمو مي كند، ساختار ابررسانه[[7]](#footnote-8) هم دچار تغيير مي شود و صفحات از پيش آماده نيستند.
* كاربران مختلف ممكن است ميدان ديدهاي مختلفي نسبت به داده ها و نيز توانايي اجراي عمليات مختلفي را داشته باشند.

# 2-4- معماري برنامه ‌هاي كاربردي وب

معماري عمومي صفحات وب از نوع سرويس‌دهنده/سرويس‌گيرنده است، با برخي تفاوت‌هاي قابل ذكر. يكي از برتري‌هاي مهم برنامه كاربردي‌ وب اين است كه اجزايش[[8]](#footnote-9) تنها روي سرويس‌دهنده قرار مي‌گيرند و هيچ نرم‌افزار يا تنظيم ويژه‌اي در طرف سرويس‌گيرنده لازم نيست. تفاوت مهم ديگر بين يك برنامه كاربردي‌ وب و يك سيستم سرويس‌دهنده/سرويس‌گيرنده در طبيعت ارتباط سرويس‌دهنده و سرويس‌گيرنده است. پروتكل اصلي ارتباطات در برنامه كاربردي‌ وب پروتكل HTTP است، كه پروتكلي Connectionless است و براي ارتباطات سنگين و عاري از خطا طراحي نشده است. ارتباطات بين سرويس‌دهنده و سرويس‌گيرنده در يك برنامه كاربردي‌ وب معمولا شامل حركت بين صفحات وب است، نه ارتباط مستقيم بين اشياء طرف سرويس‌دهنده و سرويس‌گيرنده. به شكل انتزاعي ميتوان گفت همه پيامها در يك برنامه كاربردي‌ وب مي‌توانند به عنوان در**خواست** و **دريافت** صفحات وب در نظر گرفته شوند.

به دليل خصلت تركيبي تكنولوژي هاي وب، استفاده از مدل هاي چند لايه [[9]](#footnote-10) در برنامه هاي كاربردي وب اجتناب ناپذير است. در حالت كلي مي توان معماري اين برنامه ها را با مدل سه لايه نشان داد كه در آن صفحات HTML نهايي لايه *نمايش* را مي سازند، اسكريپت هاي برنامه (كه عمليات اصلي را انجام مي دهند و به زبان هايي چون ASP يا JSP يا PHP و... نوشته شده اند) لايه منطق تجاري را تشكيل مي دهند و پايگاه داده ها (يا ساير منابع داده اي، مانند منابع ذخيره چندرسانه اي و فايل سيستم) در سومين لايه جاي مي گيرند. شكل يك شماي كلي معماري يك برنامه كاربردي وب را نشان مي دهد.

نمايش

(صفحات HTML)

مرور گر

منطق تجاري

پايگاه داده ها

HTTP

HTTP

###### لايه اول لايه دوم لايه سوم

**سرويس گيرنده (Client) سرويس دهنده (Server)**

اينترنت

**شكل 1 -**  شماي كلي معماري يك برنامه كاربردي وب ‎[13]

UDDI

# 2-5- اجزاي صفحات وب

هر صفحه مخلوطي از محتويات و دستورات مربوط به نمايش است كه توسط HTML بيان شده‌اند. بعضي صفحات داراي اسكريپت‌هايي هستند كه توسط مرورگر تفسير و اجرا مي‌شوند. اين اسكريپت‌ها رفتار پوياي صفحه را هنگام نمايش داده‌شدن روي صفحه نشان مي‌دهند و با اجزاي ديگر (مرورگر، محتويات صفحه و اجزاي ديگر؛ از قبيل اپلت‌ها، ActiveXها و Plug-inها) درارتباط هستند. بعضي وقتها كاربر اطلاعاتي را در فيلدهاي صفحه واردكرده آنها را براي پردازش شدن به سرويس‌دهنده مي‌فرستد. كاربر همچنين مي‌تواند از طريق لينك‌ها بين صفحات حركت كند. در هر حال،‌كاربر مي‌تواند ورودي‌هايي به سيستم بدهد كه ممكن است **حالتتجاري** سيستم را تغيير دهند.

از ديد سرويس‌گيرنده، صفحه‌ي وب هميشه يك فايل HTML است. ولي در طرف سرويس‌دهنده، يك صفحه مي‌تواند خودش را در شكل‌هاي مختلفي نشان بدهد. در برنامه ‌هاي كاربردي اوليه، صفحات وب توسط CGIها ساخته مي‌شدند. در سيستم‌هاي مبتني بر CGI ،‌معمولا يك دايركتوري ويژه براي قرار گرفتن CGIها روي سرويس‌دهنده ايجاد مي‌شد. وقتي يك CGI درخواست مي‌شد،‌ سرويس‌دهنده به جاي آنكه متن آنرا به طور معمولي براي درخواست كننده بفرستد (كاري كه در مورد صفحات HTML انجام مي‌شود)،‌آنرا با مفسر مناسب (مثلا Perl) اجرا كرده نتيجه را براي درخواست‌كننده صفحه پس مي‌فرستد. خروجي اين فرآيند يك فايل HTML است كه توسط مرورگر ارسال كننده‌ي درخواست دريافت مي‌شود. منطق تجاري در زمان اجراي فايل روي سيستم انجام مي‌شود. در آن زمان، امكان برقراري ارتباط با ساير منابع سرويس‌دهنده (مانند پايگاه داده و اجزاي لايه‌هاي مياني) وجود دارد.

سرويس‌دهنده‌هاي امروزي نسبت به اين روش قديمي بهتر عمل مي‌كنند. بطورکلی صفحات وب را مي‌توان به سه دسته تقسيم كرد: صفحه‌هاي اسكريپت دار، صفحه‌هاي كامپايل شده، صفحات مخلوط (هر دو) در دسته اول، صفحاتي كه كاربر مي‌تواند درخواست كند، به عنوان فايل‌هاي اسكریپت در سيستم‌عامل سرويس‌دهنده شناخته مي‌شوند. وقتي اين صفحه درخواست شود، يك مفسر آنرا اجرا كرده خروجي را به شكل يك فايل HTML براي درخواست كننده مي‌فرستد. مثال اين روش، ASP از مايكروسافت و Cold Fusion ست. در دسته دوم، صفحات كامپايل شده، سرويس‌دهنده وب يك فايل باينري را بار كرده اجرا مي‌كند. كد كامپايل شده از مشخصات درخواست رسيده استفاده مي‌كند و به منابع سيستم مورد نيازش براي ساختن جريان خروجي HTML دسترسي دارد. معمولا صفحات كامپايل شده كارايي بهتري نسبت به اسكريپت‌ها دارند. با فرستادن پارامترهايي به صفحات كامپايل شده، او مي‌تواند كارهاي مختلفي انجام دهد. يك فايل كامپايل شده ممكن است همه كارهايي را كه يك دايركتوري از اسكريپت‌ها انجام مي‌دادند، يك تنه انجام دهد. تكنولوژي‌هايي كه از اين روش استفاده مي‌كنند ISAPI از مايكروسافت و NSAPI از Netscape هستند. دسته سوم صفحاتي هستند كه به صورت اسكريپت نوشته مي‌شوند،‌ و بعد در اولين نوبتي كه درخواستي برايشان برسد، كامپايل مي‌شوند و از آن پس اين نسخه كامپايل شده براي درخواست‌هاي بعدي مورد استفاده قرار خواهد گرفت. فقط وقتي محتويات صفحه اصلي تغيير كنند، صفحه احتياج به كامپايل شده دوباره خواهد داشت. اين روش تركيبي از انعطاف‌پذيري اسكريپت‌ها و كارايي كدهاي كامپايل‌شده است. JSP (مخفف Java Server Pages) نمونه‌اي از اين روش است.

# 2-6- پروتكل ارتباطي وب (HTTP)

مرورگرها و وب سرور ها از يك پروتكل خاص به نام HTTP [[10]](#footnote-11) استفاده مي كنند ، كه دقيقا چگونگي فورمت يك مرورگر و چگونگي ارسال درخواست به يك وب سرور را تعريف مي كند. مرورگر كاربر درخواستي به شكل يك خط از كاراكتر هاي ASCII كه به وسيله يك CR[[11]](#footnote-12) ويك Line-Feed Pair تمام مي شود مي فرستد. يك سرور خوب احتياجي به CR نخواهد داشت. اين درخواست شامل كلمه GET، يك فضاي خالي ومكان سند مورد نظر مي باشد.

وقتي كه يك وب سرور/سايت، تنظيم مي‌شود، معمولا براي استفاده يك دايركتوري مشخص در فايل سيستم محلي يك ماشين، مانند دايركتوري ريشه يك وب سايت، نصب مي شود. اسناد وب در اين دايركتوري يافت مي شوند.

مشخصات كامل اسناد URL[[12]](#footnote-13) ناميده مي‌شود. پروتكل HTTP نام ماشين ميزبان و شماره پورت انتخابي ونام/ مكان اسناد را مشخص مي‌كند. URL يك كلمه واحد بدون فاصله است. اگر هر كلمه ديگري در خط در خواست ديده شود، از آن ها چشم پوشي مي‌شود يا مانند يك HTTP كامل با آن ها رفتار مي شود.URL راهي براي تعيين يك موضوع يا منبع در شبكه است و مي توان گفت معادل نام يك فايل در يك فايل سيستم است. URL براي درخواست انواع مختلف موضوعات با پروتكل هاي مختلف كاربرد دارد.

علاوه بر اينHTTP در پروتكل هاي اينترنتي ديگر مانند news، ftp،gofer و... هم كاربرد دارد.هر پروتكل نوع اطلاعات يا نوع منابع ارائه شده را مشخص مي كند. امكان تنظيم كردن وب سرور، براي پورت هاي بالاي 80 نيز وجود دارد، كه اغلب براي ايجاد يك سايت خصوصي به كار مي رود. بعضي ازوب سرور ها براي تنظيمات سرور وب از پورت اضافه استفاده مي كنند كه به اداره كنندگان وب (كساني كه مسووليت اداره يك سرور وب و سايت را دارند) اجازه مي‌دهد كه از راه دور، تنها به وسيله يك مرورگر، تنظيم كردن يك سرور وب را مديريت كنند.

يكي از مهمترين اهداف طراحي يك سيستم تحت وب (به طور عمومي) افزايش كارايي وكاهش ميزان خطا مي باشد. در اولين سيستم تحت وب در CERN، صفحات وب تنها شامل لينك‌هايي به اسناد وكامپيوتر هاي ميزبان بودند. در حال حاضر HTML با توجه به خصوصياتش مي تواند اين ويژگي را با اضافه كردن عناصر جديد تغيير دهد.

مرور گرها و وب سرور هاي يك سيستم بايد داراي اين شرايط باشند. تمام اينها باعث مي‌شود كه از يك پروتكل غيراتصال‌گرا مانند HTTP (به عنوان يك پروتكل مبنا) براي مديريت در خواست ها استفاده كنيم.HTTP به عنوان يك پروتكل غيراتصال‌گر معرفي مي‌شود، زيرا به محض شكل گرفتن يك تقاضا، ارتباط بين سرور و ايستگاه كاري برقرار مي‌شود. زماني اين ارتباط به وسيله سرور قطع مي شود كه تمام اسناد فرستاده شده باشد. كاربر ممكن است قبل از اتمام كامل اسناد از ادامه ارتباط صرف نظر كند، در اين حالت سرور هيچ خطاي ارتباطي را نمي تواند ثبت كند. سرور نيازي ندارد كه اطلاعاتي را درباره درخواست‌هاي بعد از قطع ارتباط نگهداري كند كه اين امكان را براي ميزبان و ايستگاه كاري به وجود مي آورد كه به صورتمستقل از هم عمل كنند و در مقابل قطع موقتي شبكه ايمن باشند.

HTTP روي TCP[[13]](#footnote-14) اجرا مي شود ولي مي تواند روي هر سرويس غير اتصال گرا نيز اجرا شود. TCP يك پروتكل شبكه (لايه پايين)است كه در اينترنت استفاده مي شود و كامپيوتر ها را قادر مي سازد كه شبكه شوند وتبادل اطلاعات داشته باشند. TCP (كه معمولا به صورت تركيبي با پروتكل اينترنت ياIP به كار مي‌رود) در پياده سازي يكي از لايه هاي OSI[[14]](#footnote-15) كه مدل مرجع براي شبكه است كاربرد دارد.

پروتكل مرتبط ديگر، HTTPS يا HTTP با SSL [[15]](#footnote-16)، به HTTP ذكر شده شبيه است. اين پروتكل در اينترنت براي مبادله اطلاعات حساس(مانند اطلاعات شخصي و اطلاعات مالي) مورد استفاده قرار مي گيرد. اين پروتكل ايمني اطلاعات از رمزنگاري استفاده مي كند.

در حال حاضر نسخه 1.1 از HTTPدر اينترنت مورد استفاده قرار مي گيرد. يك نسخه جديد ، HTTP-NG به وسيله W3C تعريف شده است كه هدف از آن رفع كمبود هاي فعلي به وسيله فعاليت هاي مهندسي مانند پيمانه‌اي بودن و سادگي و چندلايه‌اي بودن وافزايش توانايي ، انعطاف و كارايي بيشترمي باشد. HTTP-NG ، مطمئنا براي ارتباط اينترنتي كاراتر اما پيچيده تر مورد توجه قرار خواهد گرفت. اميد مي‌رود كه HTTP-NG مفهوم شيء در اينترنت را توسعه دهد و ارتباطات توزيع شده بين اشياء را آسان كند.

**2-7- ساختاردهي به داده‌هاي وب باXML**

XML[[16]](#footnote-17) تکنولوژی است براي نظام‌مند كردن ساختاردهي به داده ها در وب، که به شکل گسترده از آن پشتيبانی می شود. اين تکنولوژی باز است، به اين معنی که متعلق به شرکت خاصی نيست و اولين بار در کنسرسيوم W3C در سال ۱۹۹۶ برای ساده کردن انتقال ديتا ايجاد شده است.

با گسترده شدن استفاده از وب در دهه ۹۰ کم کم محدوديت های HTML مشخص شد. ضعف HTML در توسعه پذيری (قابليت اضافه و کم کردن خواص) و ضعف آن در توصيف ديتاهايی که درون خود نگهداری می کند برنامه نويسان را از آن نااميد کرد. همچنين مبهم بودن تعاريف آن باعث شد از توسعه يافتن باز بماند. در پاسخ به اين اشکالات W3C يک سری امکانات را در جهت توسعه HTML به آن افزود که امکان تغيير ساختار متنهای HTML مهم ترين آن است. اين امکان را CSS [[17]](#footnote-18) می نامند. باكمك اين تكنولوژي، مسائل مربوط به نمايش از مسائل مربوط به ساختار صفحه و اطلاعات جدا مي‌شوند و صفحات ترتيب منطقي‌تر و منظم‌تري پيدا مي‌كنند.

اين توسعه تنها يک راه موقتی بود. بايد يک روش استاندارد شده، توسعه پذير و دارای ساختار قوی ايجاد می شد. در نتيجه W3C استاندارد XML را ساخت.

XML دارای قدرت و توسعه پذيریSGML [[18]](#footnote-19) و سادگی که در ارتباط در وب به آن نياز دارد است. استقلال اطلاعات يا جدا بودن محتوا از ظاهر يک مشخصه برای XML به حساب می آيد. متنهای XML فقط يک ديتا را توصيف می کنند و برنامه ای که XML برای آن قابل درک است بدون توجه به زبان و سيستم عامل قادر است به اطلاعات درون فايل XML هر گونه شکلی که مايل است بدهد. متنهای XML حاوی ديتا هستند بدون شکل خاص، بنابراين برنامه ای که از آن می خواهد استفاده کند بايد بداند که چگونه می خواهد آن اطلاعات را نمايش دهد. بنابراين نحوه نمايش يک فايلXMLدر يک كامپيوتر شخصي با PDA و تلفن همراه می تواند متفاوت باشد.

وقتی يک برنامه با متن XML مواجه می شود بايد مطمئن باشد که آن متن حاوی ديتای مورد نظر خود است. اين اطمينان توسط برنامه هايی با نام تجريه‌گر[[19]](#footnote-20) حاصل می‌شود. تجزيه‌گرها دستورات متن XML را بررسی می کنند. همچنين آنها به برنامه کمک می‌کنند تا متن های XML را تفسير کند. به صورت اختياری هر متن XML می تواند به متن ديگری اشاره کند که حاوی ساختار فايل XML اصلی باشد. به آن متن XML دوم DTD [[20]](#footnote-21)گفته می شود.

وقتی فايل XML به DTD اشاره می کند برنامه تجزيه‌گر فايل اصلی را با DTD بررسی می کند که آيا به همان ساختاری که در DTD توصيف شده شکل گرفته است يا خير. اگر يک تجزيه‌گر XML بتواند يک متن را به درستی پردازش کند متن XML نيز به شکل صحيحی فرمت شده است.

وقتی که اکثر نرم افزارها امکانات وبی خود را افزايش دادند اين طور به نظر می رسد که XML به عنوان يک تکنولوژی جهانی برای فرستادن اطلاعات بين برنامه ها انتخاب شود. تمامی برنامه هايی که از XML استفاده می کنند قادر خواهند بود که XML همديگر را بفهمند. اين سطح بالای تطابق بين برنامه ها باعث می شود که XML يک تکنولوژی مناسب برای وب سرويس باشد. چون بدون اينکه احتياج به سيستم عامل و سخت افزار يکسان باشد می تواند اطلاعات را جابجا کند.

2-8- **ارتباطات توزيع شده در وب**

## با گسترش استفاده از معماري برنامه‌هاي كاربردي وب بعنوان معماري پايه براي نرم‌افزارهاي كاربردي گوناگون، پروتكل ارتباطي اصلي وب، يعني HTTP كمبودهاي خود را نشان داد. مهمترين نقطه ضعف اين پروتكل از يك نگاه شيء گرا، طبيعت غير اتصال گراي آن است. در برنامه نويسي شيء گرا، استفاده از اشياء توزيع شده سابقه‌ي زيادي دارد. هرچند مي‌توان يك سيستم تحت وب را نوعي نرم افزار توزيع شده در نظر گرفت كه اشياء آن (صفحات وب) در سطح وب توزيع شده‌اند، ولي به دليل غير ارتباطگرا بودن پروتكل، تفاوت‌هاي زيادي بين سيستم‌هايي كه تنها از اين پروتكل استفاده كنند با الگوهاي طراحي پيشين نرم‌افزارهاي توزيع شده وجود خواهد داشت. چرا كه بيشتر سيستم‌هاي توزيع شده به وجود ارتباط پايدار در شبكه تكيه مي‌كنند.

## علاوه بر آن، ايجاد بسته‌هاي حجيم HTTP براي انتقال هر درخواست كوچك بين اشياء طرف سرويس گيرنده و سرويس دهنده، كارايي برنامه را به طور قابل ملاحظه‌اي كاهش مي‌دهد. از سوي ديگر، در هر نوع گسترشي كه براي حل اين مشكلات پيشنهاد شود، بايد حفظ مزاياي فعلي برنامه‌هاي تحت وب منظور شود و گسترش نبايد به قيمت از دست رفتن اين مزايا گردد. از جمله اين مزايا مي‌توان به سادگي توسعه و نگهداري آن، و نيز توانايي كار با سرويس‌گيرنده‌هاي متنوع، با امكانات حداقل و كم‌توان‌تر اشاره كرد، همچنين توانايي آن براي كار روي شبكه‌هايي با ارتباطات ضعيف.

## در حال حاضر سه روش توزيع اشياء براي برنامه‌هاي كاربردي وب بكار مي‌روند: RMI[[21]](#footnote-22) از شركت سان، SOAP[[22]](#footnote-23) از W3C، و CORBA[[23]](#footnote-24) از OMG. هر كدام از اين سه روش واسط‌هايي براي برقراري ارتباط بين اشياء توزيع شده و اشيائي كه منطق تجاري را پياده سازي مي‌كنند استفاده مي‌نمايند. همگي سرويس‌هاي نام و دايركتوري براي مشخص كردن جاي شيء در سيستم ارائه مي‌كنند، و همگي مجموعه‌اي از سرويس‌هاي امنيتي دارند. ضمنا مي‌توان براي ارتباط بين اشياي ايجاد شده در يك روش با اشياي ايجاد شده در روش‌هاي ديگر، پل‌هايي ايجاد كرد. در اينجا به توضيح مختصر هر يك از اين سه روش مي‌پردازيم.

## 2-8-1- RMI

## RMI استاندارد جاوا براي اشياي توزيع شده است. RMI به كلاس‌هاي جاوا اجازه مي‌دهد با ديگر كلاس‌هاي جاوا كه به طور بالقوه مي‌توانند در ماشين ديگري باشند ارتباط داشته باشند. بعنوان يك مجموعه API و يك مدل براي اشياي توزيع شده، RMI به برنامه نويسان اجازه مي‌دهد به آساني به ايجاد سيستم‌هاي توزيع شده مبادرت نمايند. نسخه‌هاي اوليه RMI از سريال‌سازي[[24]](#footnote-25) جاوا و نيز از JRMP[[25]](#footnote-26) براي فراخواني متدها از روي شبكه استفاده مي‌كردند و JRMP پروتكل انتقال[[26]](#footnote-27) فراخواني متدها از راه دور بود. در نسخه‌هاي جديدتر RMI، پشتيباني از پروتكل انتقال IIOP[[27]](#footnote-28) (كه توسط CORBA ايجاد شده) نيز افزوده شده است. هرچند شايد به نظر بيايد از نقطه نظر طراح نرم‌افزار، لايه انتقال مخفي است و نقشي در فرآيند طراحي ندارد، ولي به دليل برخي ناسازگاري‌هاي بين دو استاندارد فوق، توجه به تكنولوژي لايه انتقال در هنگام طراحي ضروري است.

## RMI دو نوع شيء جديد معرفي مي‌كند: ريشه[[28]](#footnote-29) و اسكلت[[29]](#footnote-30). ريشه در طرف سرويس گيرنده قرار مي گيرد و نقش شيء بيروني را ايفا مي‌نمايد. اسكلت روي سرويس دهنده مي نشيند و همه جزييات مربوط به ارتباطات را مديريت مي‌كند. براي ايجاد اين اشيا نيازي به كد نويسي نيست و آنها به طور خودكار از روي كلاس‌هاي منطق تجاري برنامه توسط كامپايلر مخصوصي بنام rmic توليد مي‌شوند. هدف اين است كه تا حد امكان طراح و برنامه نويس درگير جزييات مربوط به ارتباطات راه دور نباشند. شكل دو لايه‌هاي RMI را نشان مي‌دهد.

اشياي سرويس دهنده اشياي سرويس گيرنده

اسكلت ريشه

زيربناي RMI

لايه انتقال (JRMP يا IIOP)

**شكل 2 –** لايه هاي معماري RMI

## در برنامه‌هاي كاربردي وب، RMI اغلب به عنوان يك رسانه ارتباطي بين يك اپلت و يك Application Server عمل مي‌كند. خود اپلت به عنوان بخشي از صفحه وب براي مرورگر كاربر فرستاده مي‌شود. تنها چيزي كه ضروري است اين است كه مرورگر كاربر از جاوا پشتيباني كند. بقيه چيزها و كلاس هاي مورد نياز، به طور خودكار براي مرورگر ارسال خواهند شد. وقتي اپلت اجرا شد، با سرويس دهنده ارتباط برقرار مي‌كند، درخواست ارجاعي به شيء مورد نظرش مي‌كند، و سپس شروع به فراخواني متدهاي آن مي‌نمايد، چنانچه انگار يك شيء محلي است.

## 2-8-2- CORBA

## CORBA يك محصول نيست، بلكه استانداردي است كه توسط OGM عرضه شده است. CORBA تعيين مي‌كند اشيا چگونه بايد از طريق شبكه با هم ارتباط و همكاري داشته باشند و متدهاي يكديگر را صدا بزنند. CORBA اجازه مي‌دهد اشيا به هر زباني نوشته شوند، و تعاريف آن وابسته به زبان خاصي نيست. اين كار بوسيله تعريف واسطي بنام IDL[[30]](#footnote-31) انجام شده است، كه در تئوري، اشيا را مستقل از زبان برنامه نويسي آنها تعريف مي‌كند. هرچند در واقعيت زبان‌هايي كه توان پشتيباني از IDL را دارند محدودند.

## ميان‌افزاري كه CORBA براي ارتباطات بين ماشين‌هاي مختلف تعريف مي‌كند ORB[[31]](#footnote-32) نام دارد. ORBها روي سرويس دهنده و سرويس گيرنده نصب مي‌شوند و برقراري ارتباط بين اشيا را بر عهده مي‌گيرند. ارتباطات بين خود ORBها در شبكه‌هايي كه برمبناي TCP/IP ساخته شده‌اند توسط IIOP انجام مي‌شود. شكل سه نحوه ارتباط بين اشيا را در اين مدل نشان مي‌دهد.

ORB

**IIOP**

سرويس گيرنده

پياده سازي شيء

ORB

**شكل 3 –** مدل اشيا در CORBA

**2-8-3- SOAP**

SOAP[[32]](#footnote-33) يکی از عمومی ترين استاندارد هايی است که در وب سرويس ها استفاده می‌شود. اين استاندارد اولين بار توسط DeveloperMentor، شرکت UserLand و مايکروسافت در سال ۱۹۹۸ ساخته شده و نسخه اول آن در سال ۱۹۹۹ ارايه شده است. آخرين نسخه SOAP، نسخه 1.2 بود که در دسامبر سال ۲۰۰۱ در W3C ارايه شد. نسخه 1.2 نشان دهنده کار زياد بر روی آن و نمايانگر اشتياق زياد صنعت IT برای استفاده از SOAP و وب سرويس است.

هدف اصلی SOAP ايجاد روشی جهت فرستادن ديتا بين سيستم هايی است که بر روی شبکه پخش شده اند. وقتی يک برنامه شروع به ارتباط با وب سرويس می کند، پيغام های SOAP وسيله ای برای ارتباط و انتقال ديتا بين آن دو هستند. يک پيغام SOAP به وب سرويس فرستاده می شود و يک تابع يا سابروتين را در آن به اجرا در می آورد به اين معنی که اين پيغام از وب سرويس تقاضای انجام کاری را دارد. وب سرويس نيز از محتوای پيغام SOAP استفاده کرده و عمليات خود را آغاز می کند. در انتها نيز نتايج را با يک پيغام SOAP ديگر به برنامه اصلی می فرستد.

به عنوان يک پروتکل مبتنی بر XML، پروتکل SOAP تشکيل شده از يک سری الگوهای XML است. اين الگوها شکل پيغام های XML را که بر روی شبکه منتقل می‌شود را مشخص می کند، مانند نوع ديتاها و اطلاعاتی که برای طرف مقابل تفسير کردن متن را آسان کند. در اصل SOAP برای انتقال ديتا بر روی اينترنت و از طريق پروتکل HTTP طراحی شده است ولی از آن در ديگر مدلها مانند LAN نيز می توان استفاده کرد. وقتی که وب سرويس ها از HTTP استفاده می کنند به راحتی می توانند از ديوار آتش عبور کنند.

يک پيغام SOAP از سه بخش مهم تشکيل شده است: پوشش[[33]](#footnote-34)، سربرگ[[34]](#footnote-35)، و بدنه. قسمت پوشش برای بسته بندی کردن کل پيغام به کار می رود. اين بخش محتوای پيغام را توصيف و گيرنده آن را مشخص می کند. سربرگ يک بخش اختياری می باشد و مطالبی مانند امنيت و مسيريابی را توضيح می دهد. بدنه پيغام بخشی است که ديتاهای مورد نظر در آن جای می گيرند. ديتاها بر مبنای XML هستند و از يک مدل خاص که الگوها آن را توضيح می دهند تبعيت می کنند. اين الگو ها به گيرنده کمک می کنند تا متن را به درستی تفسير کند. پيغام های SOAP توسط سرورهای SOAP گرفته و تفسير می شود تا در نتيجه آن، وب سرويس ها فعال شوند و کار خود را انجام دهند.

برای اينکه از SOAP در وب سرويس استفاده نکنيم از تعداد زيادی پروتکل بايد استفاده شود. برای مثال XML-RPC تکنولوژی قديمی تری بود که همين امکانات را ايجاد می کرد. به هر حال، خيلی از سازندگان بزرگ نرم افزار SOAP را بر تکنولوژی‌های ديگر ترجيح دادند. دلايل زيادی برای انتخاب SOAP وجود دارد که خيلی از آنها درباره پروتکل آن است که فراتر از اين متن می باشد. دو برتری مهم SOAP نسبت به تکنولوژی های ديگر عبارتند از قابليت توسعه و سادگی.

پيغام های SOAP معمولا ً کدهای زيادی ندارند و برای فرستادن و گرفتن آن به نرم افزارهای پيچيده نياز نيست. SOAP اين امکان را به برنامه نويس می دهد تا بنا به نياز خود آن را تغيير دهد. در آخر بدليل اينکه SOAP از XMLاستفاده می کند می تواند بوسيله HTTP اطلاعات را انتقال بدهد بدون اينکه زبان برنامه نويسی، سيستم عامل و سخت افزار برای آن مهم باشد.

**2-8-4- مقايسه روش‌هاي توزيع اشياء روي وب**

## از بين سه روش فوق، استفاده از RMI هم براي برنامه‌نويس ساده‌تر است، و هم امكانات كافي در اختيار وي قرار مي‌دهد. در حالي كه استفاده از CORBA با وجود قابليت‌هاي كامل آن، مشكلات بيشتري دارد. در عين حال CORBA به دليل آنكه سعي كرده وابسته به يك زبان خاص نباشد، برخي از قابليت‌هاي RMI را ندارد، كه مهمترين آن توان انتقال پارامترهايي غير از انواع اصلي (مثلا يك شيء) را ندارد. نقطه ضعف اصلي RMI نسبت به CORBA اين است كه تنها ارتباط بين اشياء نوشته شده به زبان جاوا را بر قرار مي‌كند. در حالي كه CORBA (از نظر تئوريك) مي‌تواند بين اشيايي كه در همه زبان‌هاي برنامه‌نويسي ايجاد شده‌اند ارتباط برقرار نمايد. SOAP همانطور كه از نامش برمي‌آيد، سعي در ساده كردن ارتباطات بين اشياء توزيع شده دارد، و از اين جهت استاندارد راحتي است. SOAP براي تعريف داده‌ها از XML استفاده مي‌كند و بستر ارتباطي آن هم HTTP است. در عوض SOAP نسبت به دو روش قبل بسياري از جزييات را پشتيباني نمي‌كند، و براي مقاصد پيچيده مناسب نيست. بيشترين كاربرد آن در ارتباطات داده‌اي محدود بين اشياي پراكنده روي اينترنت است.

## 2-9- ارتباط بين سرويس‌هاي وب

چشم‌انداز آتي برنامه‌هاي كاربردي وب، فضايي را تصوير مي‌كند كه در آن برنامه‌هاي گوناگون با جستجو در دايركتوري‌ها ، سرويس‌دهنده‌هايي را مي‌يابند كه سرويس مورد نظر آنها را ارائه مي‌دهد، و از سرويس آن استفاده مي‌كنند. در زمينه‌ي ارتباطاتB2B[[35]](#footnote-36) بين برنامه‌ها، استانداردهايي تدوين شده است كه در اينجا به آنها پرداخته مي‌شود.

**2-9-1- WSDL**

استاندارد ديگری که نقش اساسی در وب سرويس بازی می کند WSDL[[36]](#footnote-37) است. يکی از مشخصات وب سرويس ها توصيف خود آنهاست به اين معنی که وب سرويس دارای اطلاعاتی است که نحوه استفاده از آن را توضيح مي‌دهد. اين توضيحات در WSDL نوشته می شود، متنی به XML که به برنامه ها می گويد اين وب سرويس چه اطلاعاتی لازم دارد و چه اطلاعاتی را بر می گرداند.

وقتی که سازندگان نرم افزار برای اولين بار SOAP و ديگر تکنولوژی های وب سرويس را ساختند دريافتند که برنامه ها قبل از اينکه شروع به استفاده از يک وب سرويس بکنند بايد اطلاعاتی درباره آن را داشته باشند. اما هر کدام از آن سازندگان برای خودشان روشی برای ايجاد اين توضيحات ابداع کردند و باعث شد که وب سرويس ها با هم هماهنگ نباشد. وقتی IBM و مايکروسافت تصميم گرفتند تا استاندارد های خود را يکسان کنند WSDL بوجود آمد. در ماه مارس سال ۲۰۰۱ مايکروسافت، IBMو Ariba نسخه 1.1 را به W3C ارائه کردند. گروهی از W3C بر روی اين استاندارد کار کردند و آن را پذيرفتند. هم اکنون اين تکنولوژی در دست ساخت است و هنوز کامل نشده. ولی هم اکنون اکثر سازندگان وب سرويس از آن استفاده می کنند.

هر وب سرويسی که بر روی اينترنت قرار می گيرد دارای يک فايل WSDL است که مشخصات، مکان و نحوه استفاده از وب سرويس را توضيح می دهد. يک فايل WSDL نوع پيغام هايی که وب سرويس می فرستد و می گيرد را توضيح می‌دهد، مانند پارامترهايی که برنامه صدا زننده برای کار با وب سرويس بايد به آن بفرستد. در تئوری يک برنامه در وب برای يافتن وب سرويس مورد نظر خود از روی توضيحات WSDL ها جستجو می کند. در WSDL اطلاعات مربوط به چگونگی ارتباط با وب سرويس بر روی HTTP يا هر پروتکل ديگر نيز وجود دارد.

اين مهم است که بدانيم WSDL برای برنامه ها طراحی شده است نه برای خواندن آن توسط انسان. شکل فايلهای WSDL پيچيده به نظر می آيد ولی کامپيوترها می توانند آن را بخوانند و تجزيه و تحليل بکند. خيلی از نرم افزارهايی که وب سرويس می سازند فايل WSDL مورد نياز وب سرويس را نيز توليد می کنند بنابراين وقتی برنامه نويس وب سرويس خود را ساخت به شکل خودکار WSDL مورد نياز با آن نيز ساخته می‌شود و احتياجی به آموزش دستورات WSDL برای ساختن و استفاده از وب سرويس نيست.

**2-9-2- UDDI**

UDDI[[37]](#footnote-38)، به شرکتها و برنامه نويسان اجازه می دهد تا وب سرويس های خود را بر روی اينترنت معرفی کنند. اين استاندارد در اصل بوسيله مايکروسافت، IBM و Ariba و پنجاه شرکت بزرگ ديگر ساخته شده است. با استفاده از UDDI شرکتها می‌توانند اطلاعات خود را در اختيار شرکت های ديگر قرار بدهند و مدل B2B ايجاد کنند. همان طور که از نام آن مشخص است شرکت ها می توانند وب سرويس خود را معرفی کنند، با وب سرويس ديگران آشنا شوند و از آن در سيستم های خود استفاده کنند. اين استاندارد جديدی است و در سال ۲۰۰۰ ساخته شده است و کنسرسيومی از شرکتهای صنعتی در حال کار بر روی آن هستند. نسخه دوم UDDI در ماه ژوئن سال ۲۰۰۱ ارائه شد و نسخه سوم آن در دست ساخت است.

UDDIيک متن مبتنی بر XML را تعريف می کند که در آن شرکت ها توضيحاتی درباره چگونگی کار وب سرويس شرکتشان و امکانات خود می دهند. برای تعريف اين اطلاعات از شکل خاصی که در UDDI توضيح داده شده استفاده می شود. شرکت ها می توانند اين اطلاعات را در UDDI شرکت خود نگهداری کنند و تنها به شرکت های مورد نظرشان اجازه دستيابی به آنها را بدهند يا آنها را در مکان عمومی و در اينترنت قرار دهند.

بزرگترين و مهمترين پايگاه UDDI پايگاه UDDI Business Registry يا UBR نام دارد و توسط کميته UDDI طراحی و اجرا شده است. اطلاعات اين پايگاه در چهار نقطه نگهداری می شود: مايکروسافت، IBM ،SAP و HP. اطلاعاتی که در يکی از چهار پايگاه تغيير کند در سه تای ديگر نيز اعمال می شود.

اطلاعات درون اين پايگاه ها شبيه دفترچه تلفن است: صفحات سفيد که در آنها اطلاعات تماس شرکت ها و توضيحات متنی آنهاست.،صفحات زرد كه حاوی اطلاعات طبقه بندی شده شرکتها و اطلاعات درباره توانايی های الکترونيکی آنها می باشد، و صفحات سبز كه حاوی اطلاعات تکنيکی درباره سرويس های آنها و نحوه پردازش اطلاعات شرکت آنها می باشد.

اطلاعات تجاری و سرويس های شرکت ها کاملا ً طبقه بندی شده است و اجازه می‌دهد که به راحتی در آنها جستجو کرد. سپس متخصصان IT می توانند از اين اطلاعات استفاده کرده و شرکت ها را برای خدمات بهتر به هم متصل کنند. با اين شرح UDDI امکان پياده سازی مدل B2B را ايجاد می کند و شرکتها می توانند از سرويس‌های يکديگر استفاده کنند.

شرکت هايی که به UDDI علاقه نشان داده اند قدرتمند هستند و خيلی از آنها از وب سرويس و استانداردهای آن در محصولات خود استفاده می‌کنند.

NTT Communications of Tokyo يکی از شرکت هايی است که در حال اضافه کردن توضيحاتی به ساختار UDDI است. در هر حال حاضر شرکت ها هنوز کمی درباره وارد کردن خود در پايگاه های عمومی محتاط هستند. اين چيز عجيبی نيست. شرکتها ابتدا اين امکانات را فقط برای شرکای خود ايجاد می کنند. شرکتهای بزرگ نيز برای مديريت بر سرويس های خود و اشتراک آنها بين قسمت های مختلف از اين استاندارد استفاده می کنند. وقتی اين استاندارد به حد بلوغ خود برسد و کاربران با آن احساس راحتی بکنند استفاده از آن نيز در مکان های عمومی فراگير خواهد بود.

اين تغيير رويه برای شرکت های بزرگی که B2B را به روش های قديمی اجرا کرده بودند مشکل است. بعضی نيز اشکال امنيتی بر اين روش می گيرند و مايل نيستند اطلاعاتشان را بدهند. اما با گذشت زمان و کامل شدن اين تکنولوژی و درک لزوم استفاده از آن شرکت ها چاره ای جز استفاده از آن ندارند.

**2-9-3- مقايسه و ارزيابي**

نقش‌هايي كه UDDI و WSDL ايفا مي‌كنند از برخي جهات متفاوت است. WSDL سعي دارد جزييات يك سرويس مشخص را تدوين كند، در حالي كه UDDI بيشتر روي ايجاد بستري ارتباطي بين سرويس‌هاي وب و فراهم كردن فهرست‌هايي از سرويس‌هاي موجود متمركز است. از اين رو شايد مدلي تركيبي از UDDI و WSDL (و نيز SOAP) براي تامين نيازهاي ارتباطي سرويس‌هاي وب مناسب باشد. در اين مدل UDDI بستر ارتباطي را فراهم مي‌كند و فهرستي قابل جستجو از سرويس‌هاي ارائه شده را در اختيار برنامه‌هاي كاربردي متقاضي قرار مي‌دهد. پس از يافته شدن سرويس مورد نظر، با رجوع به آدرس WSDL مربوط به به آن سرويس، جزييات فني ارتباط براي متقاضي روشن مي‌شود. سرانجام با استفاده از SOAP (يا ديگر پروتكل‌هاي ارتباط بين اشياء) ارتباط واقعي بين اشياء برقرار مي‌گردد.

## 3- مرور برخي از پژوهش هاي پيشين

# تلاش‌هايي كه براي مدون و روش‌مند كردن ساخت برنامه‌هاي كاربردي وب صورت گرفته‌اند داراي دو رويكرد كلي هستند. يك دسته به توسعه روش‌هاي كلاسيك مهندسي نرم‌افزار براي مراحل ساخت برنامه‌هاي كاربردي وب مي‌پردازند (مانند OOHDM) و دسته ديگر تنها به ايجاد مدل‌هايي گرافيكي براي نمايش ارتباطات صفحات وب و توصيف اين مدل‌ها با زبان‌هايي چون XML بسنده مي‌كنند (مانند WebML). در اين مقاله سعي بر اين است كه يك مدل فرآيند كامل براي همه مراحل ساخت برنامه‌هاي كاربردي وب ارائه شود. از اين رو خانواده‌اي از روش‌هاي دسته اول در اينجا مورد بررسي قرار مي‌گيرند.

# 3-1- HDM

HDM [[38]](#footnote-39) جد بزرگ گروهي از روش هاي ساخت برنامه هاي كاربردي وب است. در اين روش طراحي ابررسانه در دولايه انجام مي گيرد: لايه Hyperbase و لايه دسترسي. لايه نخست اطلاعات پايه درباره اشياء و روابط دوطرفه آنها و مسيرهاي حركت بين آنها را تعريف مي كند. دومين لايه گروهبنديها و نحوه سازماندهي عناصر اطلاعات پايه را معرفي مي كند و گامهاي كاربر در فضاي اطلاعات را ساماندهي مي نمايد. اين دو لايه خود در دو مرحله ايجاد ميشوند: طراحي كليات (كه ساختار اطلاعات و مسيرهاي حركت را بيان مي كند) و طراحي جزييات (كه جزييات فراموش شده مانند چينش و اجزا را مي پوشاند). ولي HDM يك چرخه حيات رسمي براي توسعه نرم افزار ارائه نمي كند، به همين دليل روش هاي ديگري مانند RMM ايجاد شدند تا اين نياز را بر طرف نمايند.

# 3-2- RMM

RMM[[39]](#footnote-40) روشي براي طراحي، ساخت و نگهداري سيستم‌هاي وبي است. هدف اصلي آن كاهش هزينه نگهداري سايتهاي وب مبتني بر پايگاه داده است. در اين روش از يك مدل بصري[[40]](#footnote-41) سيستم براي تسهيل مباحث طراحي استفاده مي‌شود. اين روش يك فرآيند تكرار شونده است كه در آن جزييات عناصر بصري در صفحات وب، و وابستگي‌هاي آنها با عناصر پايگاه داده نشان داده مي‌شود.

RMM داراي اين هفت مرحله است: طراحي ERD ، طراحي برش‌ها[[41]](#footnote-42)، طراحي مسيرياب، طراحي پروتكل ارتباطي، طراحي واسط كاربر، طراحي رفتار، و آخرين مرحله هم پياده سازي و تست است. سه مرحله اول تصوري از دامنه برنامه كاربردي ابررسانه در قالب موجوديت ها و برش ها و مسيرهاي بين آنها ارائه مي دهند. مرحله سوم فعاليتي فني است كه تبديلات لازم براي نگاشت شماي مفهومي به ساختار پياده سازي را تعريف مي كند.

اما RMM براي ساختن برنامه ‌هاي كاربردي وب كم مي‌آورد. برنامه ‌هاي كاربردي وب شامل جزييات فني خاصي هستند كه به طور كامل توسط RMM پوشانده نمي‌شوند. از جمله آنها، اسكريپت‌نويسي در طرف سرويس‌گيرنده، اپلت‌ها، كنترل‌هاي ActiveX هستند. بعلاوه، برنامه ‌هاي كاربردي وب مي‌توانند به عنوان مكانيزمي براي توزيع در يك سيستم اشياء توزيع شده به كار روند. اپلت‌ها و ActiveXها مي‌توانند محتوي اجزايي[[42]](#footnote-43) باشند كه به صورت آسنكرون با اجزاي طرف سرويس‌دهنده از طريق RMI يا DCOM (مستقل از سرويس‌دهنده وب) ارتباط برقرار كنند. برنامه ‌هاي كاربردي پيچيده‌تر همچنين ممكن است همزمان از چند پنجره مرورگر در طرف سرويس‌گيرنده، يا از چند فريم (كه با هم ارتباط دارند) استفاده كنند

# 3-2- OOHDM

اين روش براي طراحي برنامه‌هاي كاربردي‌ وب در مقياس بزرگ، و حل مسائل مربوط به الگوهاي پيچيده مسيريابي در صفحات وب همراه با رفتار پيچيده محاسباتي ايجاد شده است. با اين فرض كه روش شئ گرا براي طراحي نرم‌افزارهاي مقياس بالا مزاياي زيادي دارد، OOHDM سعي در افزودن ملحقاتي به آن براي پوشش دادن به موارد خاص نرم‌افزارهاي تحت وب (برنامه‌‌هاي كاربردي وب) دارد.

OOHDM چرخه‌حيات هفت مرحله اي RMM را در چهار مرحله ساده كرده است:

1. طراحي مفهومي
2. طراحي مسير حركت بين صفحات[[43]](#footnote-44)
3. طراحي انتزاعي[[44]](#footnote-45) واسط
4. پياده سازي

در مرحله اول، جزييات مربوط به دامنه براساس روشهاي معمول شيء گرايي (بجاي استفاده از ERD) طراحي مي‌شوند. در مرحله دوم، اشياي مربوط به مسير حركت شناسايي و طراحي مي‌گردند. اين اشيا از طريق واسط‌ها، كه در مرحله سوم مدل سازي مي‌شوند، در دسترس كاربر قرار مي‌گيرند. سرانجام در مرحله آخر، اشياي طراحي شده در سه مرحله قبلي با امكانات محيط اجرا (مانند صفحات HTML، اسكريپت‌ها، پرس‌و جوهاي پايگاه داده و…) مطابقت مي‌يابند. در تمام اين مراحل،‌ از الگوهاي طراحي[[45]](#footnote-46) براي كاهش پيچيدگي طراحي و بازيافت راه‌حل‌هاي قبلا پيدا شده براي برخي مسائل و مشكلات متداول استفاده مي‌شود.

مزيت عمده‌ي روش OOHDM، استفاده از متدولوژي شيءگرا براي مدل سازي برنامه‌‌هاي كاربردي وب است، كه با خصلت دروني چنين برنامه كاربردي‌‌هايي همخواني‌هايي دارد. ليكن به دليل آنكه بسياري از مسائل مربوط به طراحي يك برنامه كاربردي‌ وب در مرحله چهارم اين متد اتفاق مي‌افتد، جدا سازي آن از مراحل طراحي باعث ناكارآمدي‌هايي در سه مرحله پيشين مي‌گردد. به طوري كه شايد لازم باشد در مرحله چهارم بخش عمده‌اي از طراحي صورت گرفته در مراحل قبلي بر اساس شرايط پياده سازي مورد بازبيني قرار گيرند. مدلي كه براي طراحي برنامه‌‌هاي كاربردي وب مورد استفاده قرار مي‌گيرد، بايد انعطاف لازم براي برخورد با درهم پيچيدگي مراحل طراحي و پياده سازي (كه غالبا در روند ساخت برنامه‌‌هاي كاربردي وب ناگزير است) را داشته باشد.

**3-4- مقايسه و ارزيابي**

روش HDM به عنوان پايه اصلي دو روش ديگر، پايه‌هاي اصلي مدل‌سازي ابرمتن را تعريف كرد، ولي هنوز بسياري از جزييات ديگر برنامه‌هاي كاربردي وب را نمي‌پوشاند. از سوي ديگر، HDM فاقد يك چرخه حيات رسمي است. اين چيزي است كه RMM سعي در جبران آن كرده است. در اين روش، يك چرخه حيات هفت مرحله‌اي معرفي شده است. RMM به پيروي از HDM از متدلوژي ساختيافته استفاده مي‌كند. تغيير اصلي OOHDM نسبت به اين دو، استفاده از متدولوژي شيءگرا است. در اين روش، هفت مرحله RMM در چهار مرحله خلاصه شده است. نقص مهم هر سه اين روش‌ها اين است كه براي مدل‌سازي پيچيدگيهاي معماري غالبا توزيع شده برنامه‌هاي كاربردي وب، راه حل خاصي پيشنهاد نمي‌كنند. اين پيچيدگيها بيشتر ناشي از رفتار دوگانه يك موجوديت (شيء) در دو سوي سرويس گيرنده و سرويس دهنده است، كه اگر درست مدل نشود، طراحي برنامه و نيز ارتباط دو مرحله طراحي و پياده سازي را دچار مشكل مي‌كند.

**4- معرفي مدل فرآيند جديد Web-ICONIX :**

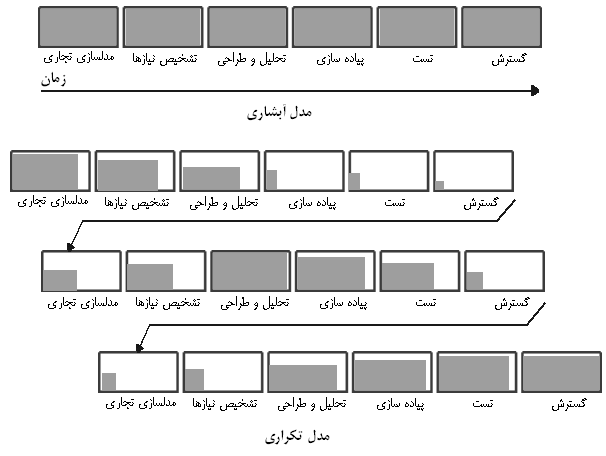
**فرآيند جديد** Web-ICONIX تركيبي از سه عنوان شناخته شده ديگر است، كه براي هماهنگ شدن با ويژگيهاي مراحل توليد يك برنامه كاربردي وب در ابعاد متوسط در نظر گرفته شده اند. چارچوب كلي فرآيد توليد نرم افزار از چرخه حيات ارائه شده در فرآيند RUP شركت Rational گرفته شده است. همچنين راهنمايي ها و توصيه هاي ارائه شده در نقشه مسير RUP براي برنامه هاي كاربردي وب نيز در اين روش لحاظ شده اند. به دليل پيچيدگي زياد پروسه توليد و هزينه سربار بالاي فرآيند RUP، در جزييات مراحل توليد (فاز ساخت) از روشApplied UseCase Driven Object Model (يا ICONIX) استفاده شده است. در اين روش محور عمليات نمودار مدل مورد كاربرد است و بر اساس آن طي فرآيندي مشخص، مدل كلاس برنامه و متدها و خصوصيات هريك از كلاس ها مشخص مي گردد. همچنين براي رسم مدل هاي خاص تحليل/طراحي صفحات وب در اين روش از ملحقات ارائه شده توسط آقاي جيم كنالن براي UML استفاده مي شود. رسم مدل داده اي سيستم نيز به مراحل ICONIX اضافه شده است. در اين فصل ابتدا سه جزء اصلي تشكيل دهنده اين روش معرفي مي‌گردند و سپس مراحل آن بيان مي‌شوند.

**4-1- فرآيند RUP [[46]](#footnote-47)**

RUP چارچوبي است كلي براي تشريح فرآيند ساخت نرم افزار. پس از آنكه تيم سه نفره شركت Rational ساخت UML را به عنوان يك شيوه ي نمادگذاري[[47]](#footnote-48) يكتا براي تشريح مدل شيء‌ به آخر رساند، تلاش خود را متوجه فرآيند توليد نرم افزار نمود.اساس RUP بر تكرار[[48]](#footnote-49) است و اساس تكرار اين است كه هر تكرار به يك محصول اجرايي ختم شود. هر تكرار شامل هر هفت مرحله چرخه حيات نرم افزار در مدل سنتي آبشاري است، يعني: مدل سازي تجاري، تخمين نيازها، تحليل و طراحي، پياده سازي، تست، نگهداري و توسعه. شكل چهار مدل تكراري و آبشاري را مقايسه مي كند.

در RUP كل فرآيند توليد نرم‌افزار به چهار فاز اصلي تقسيم مي‌شود، كه هر فاز مي‌تواند شامل يك يا چند تكرار باشد. هر فاز شامل مسيري است كه بين دو گردنه[[49]](#footnote-50) قرار دارد. اين چهار فاز عبارتند از:

* آغاز (inception)
* جزييات (elaboration)
* ساخت (construction)
* انتقال (transition)



**شكل 4 –** مقايسه مدل آبشاري با مدل تكراري

## 4-1-1- فاز آغاز

در اين فاز مدل تجاري سيستم رسم و دورنماي پروژه ترسيم مي‌شود. براي اين كار بايد همه‌ي موجوديت‌هايي كه سيستم با آنها تعامل دارد (بازيگران) شناخته شوند و نحوه تعاملشان با سيستم مشخص گردد. اين شامل تعيين همه‌ي موارد كاربرد و توضيح برخي از موارد مهم است. مدل تجاري شامل شرايط موفقيت، برآورد ريسك‌ها و تخمين منابع مورد نياز است. همچنين يك برنامه‌ي كلي كه زمانبندي مراحل انجام پروژه را نشان دهد.

### اهداف فاز آغاز

در انتهاي اين فاز اهداف چرخه‌ي حيات پروژه را تعيين مي‌كنيد و تصميم مي‌گيريد كه آنرا ادامه بدهيد يا نه. هدف اصلي اين فاز بدست آوردن هماهنگي بين تمام افراد درگير با پروژه درباره‌ي اهداف پروژه است. اهداف فرعي ديگر اين‌هايند:

* تعيين حدود و دورنماي پروژه
* تعيين موارد كاربرد حيياتي سيستم و سناريور اوليه آنها
* تعيين سناريوهاي جايگزين
* تخمين هزينه و زمان پروژه
* تخمين ريسك‌هاي احتمالي

#### خروجيهاي فاز آغاز

* سند نماي كلي [[50]](#footnote-51) : ديد كلي به هسته‌ي اصلي نيازهاي پروژه، قابليت‌هاي كليدي و حدود اصلي
* مدل مورد كاربرد: مشخص كردن همه‌ي موارد كاربردي كه در اين مرحله‌ي مقدماتي مي‌توانند شناسايي شوند)
* لغتنامه‌ي اوليه
* مدل تجاري اوليه، شامل زمينه‌ي تجارت، شرايط موفقيت، الزامات تجاري
* تخمين خطاي اوليه
* برنامه‌ي اوليه كه فازها و تكرارها را نشان مي‌دهد.

##### 4-1-2- فاز جزييات

در اين فاز تعريف محصول تصحيح مي‌شود و معماري آن مشخص مي‌گردد. همچنين خلاصه برنامه‌ي توسعه و گسترش محصول تدوين مي‌گردد. در انتهاي اين فاز اهداف و نماي پروژه دقيقتر تعيين مي‌شوند و انتخاب يك معماري و تبيين ريسك‌هاي عمده انجام مي‌گيرد. اين فاز مهمترين در بين چهار فاز ذكر شده است. فشار كار «مهندسي» نرم‌افزار در اين فاز است.

در اين فاز، در يك يا چند تكرار، يك پيش نمونه قابل اجرا از معماري سيستم ساخته مي‌شود. تعداد تكرارها بستگي به نما، اندازه، ريسك و جديدبودن پروژه دارد.

##### اهداف فاز جزييات

هدف اين فاز آناليز دامنه‌ي مساله و ايجاد يك پايه‌ي درست براي معماري، توسعه‌ي برنامه‌ي پروژه و برطرف كردن عمده‌ترين ريسك‌هاي پروژه است. تصميم‌ها در مورد معماري بايد بر اساس دركي از كل سيستم اتخاذ شوند. اين متضمن آن است كه بيشتر موارد كاربرد توصيف شوند. اهداف اوليه اين فاز عبارتند از:

* تعريف و تصحيح معماري، با بيشترين سرعت ممكن
* به دست آوردن يك برنامه‌ي درست براي فاز ساخت
* اثبات اينكه معماري مورد نظر، نياز پروژه را با هزينه و زمان معقولي تامين مي‌كند.

###### خروجيهاي فاز جزييات

* مدل مورد كاربرد
* ديدن نيازهاي اضافي و غيركاركردي كه به مورد كاربرد خاصي وابسته نيستند.
* يك نسخه‌ي قابل اجرا از معماري و مستندات پيوست (سند معماري نرم‌افزار) شامل طراحي زير مجموعه‌اي از موارد كاربرد و لغتنامه‌ي اصلاح شده
* مدل تجاري تجديد نظر شده
* فهرست ريسك‌هاي تجديد نظر شده
* برنامه‌ي توسعه‌ي پروژه
* راهنماي اوليه‌ي كاربر

4-1-3- فاز ساخت

در طول فاز ساخت، به صورت تكراري و افزايشي محصول نهايي كامل توليد مي‌شود، كه براي انتقال به كاربر آماده است. اين متضمن تبيين موارد كاربرد باقي‌مانده و ساير نيازمندي‌ها، تبيين طراحي، تكميل پياده سازي، و تست نرم‌افزار است. در انتهاي اين مرحله تصميم مي‌گيريد كه آيا نرم‌افزار، سايت و كاربران براي شروع بكار سيستم آماده‌اند يا نه.

در طول اين مرحله همه‌ي اجزاي باقي‌مانده ساخته مي‌شوند و قابليت‌هاي كاربردي توسعه يافته به محصول افزوده و تست مي‌شوند. در اين مرحله تاكيد و توجه بر مديريت منابع و كنترل عمليات براي بهينه كردن هزينه‌ها، زمانبندي و كيفيت است. اغلب پروژه‌ها آنقدر بزرگ هستند كه كارهاي اين مرحله بصورت موازي انجام گيرند. اين فعاليت‌هاي موازي باعث تسريع در توليد مي‌شوند، همچنانكه مي‌توانند پيچيدگي مديريت منابع را زياد كنند. يكي از مهمترين ويژگي‌هاي كيفي يك معماري خوب، سادگي ساخت آن است. اين يكي از دلايل آن است كه تعادل ميان توسعه‌ي معماري و برنامه‌ريزي در فاز جزييات مورد توجه بود.

اهداف فاز ساخت

* كمينه‌كردن هزينه‌هاي توسعه با بهينه كردن منابع و اجتناب از ضايعات و دوباره‌كاري‌هاي غيرلازم
* بدست آوردن كيفيت كافي در كمترين زمان ممكن
* بدست آوردن نسخه‌هاي مورد نياز (آلفا، بتا و ساير نسخه‌هاي تستي) در كمترين زمان ممكن

خروجيهاي فاز ساخت

خروجي فاز ساخت محصولي است كه براي قرار گرفتن در دست‌هاي كاربر نهايي‌اش آماده است. اين شامل موارد زير است:

* محصول نرم‌افزاري، سوار شده بر سكو
* راهنماي كاربر و توضيحي درباره‌ي نسخه‌ي فعلي

**4-1-4- فاز انتقال**

در اين مرحله محصول در دست‌هاي كاربر نهايي قرار مي‌گيرد. معمولا در اين مرحله مواردي مطرح مي‌شوند كه نياز به كار بيشتر براي ميزان كردن سيستم و تصحيح مشكلات پيش‌بيني نشده دارد. اين مرحله معمولا با يك نسخه‌ي بتا آغاز مي‌شود. در پايان اين مرحله تصميم مي‌گيريد كه آيا اهداف چرخه توليد محقق شده‌اند يا نه، و اينكه آيا بايد چرخه‌ي ديگري آغاز شود؟ اين فاز خود مي‌تواند شامل چند تكرار (مانند نسخه‌ي بتا، نسخه‌ي نهايي و رفع خطاهاي كوچك و ريزه‌كاري‌ها) باشد. همچنين مسائلي چون آموزش كاربران و تبديل داده از پايگاه داده‌ي فعلي به جديد مورد توجه قرار مي‌گيرند.

فعاليت‌هايي كه در تكرار‌هاي اين فاز انجام مي‌شوند، بستگي به هدف آن دارند: براي رفع خطاهاي كوچك، كار كدنويسي زيادي لازم نيست، ولي اگر قابليت جديدي بايد به محصول اضافه شود، تكرار شبيه فاز ساخت خواهد بود.

**اهداف فاز انتقال**

* كسب توانايي خود-پشتيباني توسط كاربر
* بدست آوردن و آغاز بكار محصول نهايي با بيشترين سرعت و كمترين هزينه

**4-1-5- تحليل و ارزيابي**

چارچوب فرآيند RUP ضمن ارائه يك نقشه مسير كلي براي هر فرآيند ساخت نرم‌افزار، بسياري از جزيياتي را كه يك مدل فرآيند بايد در نظر بگيرد ذكر كرده و تشريح مي‌كند. ضمن آنكه خصلت افزايشي و تكراري اين چارچوب فرآيند، انعطاف‌پذيري لازم را براي پوشش دادن به فرآيند ساخت انواع مختلف نرم‌افزار در اختيار قرار مي‌دهد. ولي RUP خود نمي‌تواند به عنوان يك فرآيند ساخت نرم‌افزار در نظر گرفته شود. چرا كه جزييات بسيار زياد آن، در صورت پايبندي كامل به طي كردن همه مراحل ذكر شده، هزينه سربار بالايي را بر پروژه تحميل مي‌كند، كه براي اغلب برنامه‌هاي كاربردي وب هم از نظر طولاني شدن مدت اجراي پروژه و هم از نظر بزرگ شدن تيم و افزايش هزينه‌هاي مستندسازي، به صرفه نيست. از اين رو در مدل فرآيندي كه در اينجا معرفي مي‌شود، ضمن باقي ماندن در چارچوب RUP، سعي شده با ساده سازي مراحل و كاهش مستندات توليدي، هزينه مالي و زماني پروژه كاهش يابد. در عين حال در پروژه‌هايي كه برنامه كاربردي وب گسترده است، امكان استخراج راهنمايي‌هاي دقيق‌تر از RUP وجود دارد.

**4-2- ملحقات وب UML**

**4-2-1- تعريف UML**

UML زباني استاندارد براي مشخص كردن، رسم مدل‌ها و مستند سازي اجزاي سيستم‌هاي نرم‌افزاري، يا مدل‌سازي تجاري سيستم‌هاي غير نرم‌افزاري است.

در فاصله دهه‌هاي 70 تا 80 متدولوژي‌هاي مختلفي با روش‌هاي متفاوتي براي تحليل و طراحي بر اساس شي‌ء گرايي پديدار شدند، طوري كه تعداد زبان‌هاي مدل سازي در طول سال‌هاي 89 تا 94 از 10 تا به 50 تا رسيد. اين باعث مي‌شد بسياري از كاربران براي پيدا كردن روش مناسب دچار مشكل شوند و جنگ متدها در بگيرد. توسعه‌ي UML از اواخر 1994 توسط GradyBooch و Jim Rumbaugh در شركت Rational شروع شد و در ابتدا سه متد Booch، OMT، و OOSE را با هم يكي مي‌كرد و در قالب UML 0.9 ارائه شد. سپس براي نسخة اصلي UML از شركت‌هاي مختلف نظرخواهي شد، تا آنكه در آغاز سال 1997 نسخه 1.0 آن عرضه گرديد. در پايان همان سال نسخه 1.1 عرضه شد و از سوي OMG (Object Management Group) به عنوان يك استاندارد پذيرفته شد.

**اهداف ايجاد شيوه نماد گذاري UML چنين بيان شده است.**

1. فراهم كردن يك زبان مدل‌سازي بصري رسا و قابل فهم تا بشود با آن مدل‌هايي با معني ايجاد و مبادله كرد
2. فراهم كردن مكانيسم هايي توسعه پذير و تخصصي براي رساندن مفاهيم اصلي
3. مستقل بودن از زبان برنامه نويسي يا فرآيند توسعه
4. ايجاد يك پايه رسمي براي فهم زبان‌ مدل سازي
5. كمك به رشد بازار ابزارهاي شي‌ء گرا
6. پشتيباني از مفاهيم توسعه سطح بالا مانند همكاري[[51]](#footnote-52)، چارچوب[[52]](#footnote-53)، الگوها و اجزا
7. تركيب بهترين تلاش‌ها

**4-2-2- انواع دياگرام‌هاي UML**

1. **مدل مورد كاربرد:** رابطه‌ي بازيگران و موارد كاربرد را نشان مي‌دهد
2. **مدل كلاس:** ساختار و محتويات كلاس‌هاي سيستم را مدل مي‌كند و روابط بين كلاس‌ها (از جمله ارث بري، شامل بودن و مرتبط بودن) را نشان مي‌دهد
3. **مدل‌هاي اثر متقابل (Interaction):** رفتار موارد كاربرد را نشان مي‌دهند و دو دسته هستند:
4. مدل توالي (sequence) كه ترتيب زماني اعمال اشياء را نشان مي‌دهد.
5. مدل همكاري (collaboration) اثر متقابل بين اشياء را نشان مي‌دهد.
6. **مدل حالت:** ترتيب حالت‌هايي را كه يك شيء در طول عمرش به آنها دچار مي‌شود نشان مي‌دهد.
7. **مدل فعاليت:** رفتار سيستم‌ را نشان مي‌دهد.
8. **مدل‌هاي فيزيكي:** رابطه بين اجزاي فيزيكي سيستم را نشان مي‌دهند.

# 4-2-3- استفاده ازUML براي مدل‌سازي برنامه ‌هاي كاربردي وب

مسائل مربوط به معماري صفحات وب چون در منطق تجاري سيستم نقش دارند، بايد همگي مدل شوند. بعلاوه، از آنجا كه آنها فقط بخشي از منطق تجاري هستند، بايد با بقيه مدل يكپارچه باشند. در بسياري از موارد بخش اصلي منطق تجاري پشت سرويس دهنده وب و در يكي از لايه‌ها پاييني اجرا مي‌شوند (و نه در مسير درخواست و ارسال صفحات وب توسط سرويس‌دهنده/سرويس‌گيرنده وب). انتخاب زبان مدل سازي معمولا با توجه به نيازهاي اين بخش از برنامه كاربردي‌ انجام مي‌گيرد. با پذيرفته شدن UML توسط OMG به عنوان زبان رسمي مدل‌سازي اشياء، سيستم‌هاي بيشتر و بيشتري با اين زبان مدل‌سازي مي‌شوند. بنابراين، سوال اين است كه چگونه منطق تجاري درون بخش وبي برنامه كاربردي‌ را هماهنگ با بقيه‌ي برنامه كاربردي‌ مدل سازي كنيم؟

مدل‌ها با ساده كردن برخي جزييات،‌ به فهم سيستم كمك مي‌كنند. انتخاب آنچه بايد مدل شود اثر بزرگي در يافتن مشكل و تجسم راه‌حل دارد. برنامه ‌هاي كاربردي وب، مانند بقيه سيستم‌هاي نرم‌افزاري، به‌وسيله‌‌ي مدل‌هايي از قبيل مدل مورد كاربرد، مدل پياده‌سازي، مدل توسعه، مدل امنيتي و. . . نشان داده مي‌شوند. يك مدل ديگر كه منحصراً توسط سيستم‌هاي وبي مورد استفاده قرار مي‌گيرد، نقشه‌ي سايت است، كه صفحات وب و مسيرهاي حركت بين آنها را نشان مي‌دهد.

بيشتر تكنيك‌هاي مدل‌سازي كه امروزه مورد استفاده قرار مي‌گيرند براي مدل‌سازي مدل‌هاي اصلي يك برنامه كاربردي‌ وب مناسب هستند، و به بحث بيشتري احتياج ندارند. ولي يك مدل خيلي مهم، يعني مدل تحليل/ طراحي[[53]](#footnote-54) وقتي بخواهيم صفحات وب و كدهاي اجرايي مربوط به آنها را در مدل (در كنار ساير اجزاء مدل) بگنجانيم، مشكلاتي پديد مي‌آورد.

وقتي تصميم گرفتيم كه چگونه چيزي را مدل كنيم، تعيين درجه انتزاع و جزييات بسيار مهم است، تا چيزي كه تهيه مي‌شود به درد استفاده كننده‌ي مدل بخورد. معمولا گفته مي‌شود كه بهترين كار، مدل كردن مصنوعات سيستم است؛ چيزهايي كه براي توليد محصولات نهايي ايجاد و نگهداري مي‌شوند. مدل كردن كارهاي داخلي سرويس‌دهنده وب يا مرورگر، به طراحان برنامه كاربردي‌ وب كمكي نمي‌كند. مدل كردن صفحات، لينك‌هاي آنها به هم، محتويات پويا (Dynamic) كه صفحات را مي‌سازند،‌ و محتويات پويا در طرف سرويس گيرنده بسيار مهم هستند. اينها چيزهايي هستند كه طراح طراحي مي‌كند و كدنويس كدشان را مي‌نويسد. صفحات، لينك‌ها و محتويات پويا در طرف سرويس گيرنده و سرويس‌دهنده بايد مدل شوند.

گام بعدي، متناظر كردن اين چيزها با عناصر مدل‌سازي است. مثلا لينك‌ها با روابط Association (وابستگي) در مدل متناظرند. يك لينك، مسيري براي حركت از يك صفحه به ديگري ايجاد مي‌كند. به همين ترتيب، صفحات مي‌توانند در تناظر با كلاس‌ها در مدل منطقي سيستم قرار گيرند. اگر يك صفحه يك كلاس سيستم باشد، آنوقت اسكريپت‌هاي داخل صفحه متدهاي آن كلاس و متغيرهاي تعريف شده در آن صفحه (Scoped variables) صفات آن كلاس (Attributes) خواهند بود. مشكل وقتي پيدا مي‌شود كه در يك صفحه، تعدادي اسكريپت داشته باشيم كه در طرف سرويس‌دهنده عمل مي‌كنند و محتواي پوياي صفحه را مي‌سازند(ASP)، و يك مجموعه كاملا مجزا از اسكريپت‌ها هم داشته باشيم كه در طرف سرويس‌گيرنده كار مي‌كنند (JavaScript). در چنين حالتي، وقتي به كلاس يك صفحه در مدل نگاه مي‌كنيم، معلوم نيست كدام متدها و صفات (و حتي ورابط) روي سرويس‌دهنده فعال مي‌شوند (موقعي كه صفحه دارد ساخته مي‌شود) و كدام‌ها روي سرويس‌گيرنده، در زماني كه كاربر دارد با صفحات روي مرورگر خودش كار مي‌كند. بعلاوه يك صفحه وب، وقتي در يك برنامه كاربردي‌ وب قرار مي‌گيرد مي‌تواند به شكل بهتري به عنوان يك جزء سيستم مدل شود، و صرفا متناظر قرار دادن آنها با كلاس‌ها، به ما در فهم بهتر سيستم كمك نمي‌كند.

طراحان UML مي‌دانستند كه هميشه ممكن است حالتهايي پيش بيايد كه UML براي نشان دادن محتواي يك معماري يا دامنه به شكل مناسب كافي نباشد. براي حل اين مشكل، روشي براي افزودن ملحقات به UML توسط استفاده كنندگان از آن، براي گسترش دامنه معنايي آن پيش‌بيني شده‌است. اين روش به ما اجازه تعريف stereotype ، taged values و constraintهاي جديد را مي‌دهد.

Stereotype به ما امكان مي‌دهد يك نوع مفهومي جديد تعريف كنيم (مثلا براي تعريف كردن يك صفحه وب). Taged Valueها زوج‌هاي كليد-مقداري هستند كه به يك عنصر مدل مربوط مي‌شوند و به ما امكان مي‌دهند يك مقدار را به عنصري از مدل بچسبانيم. Constraints (محدوديت‌ها) قوانيني هستند كه درستي مدل را تعريف مي‌كنند. آنها مي‌توانند توسط متن معمولي، يا به‌وسيله‌ي زبان OLC[[54]](#footnote-55) تعريف شوند. روشي كه در اين پروژه درباره‌اش صحبت مي‌شود، مجموعه ملحقاتي به UML براي برنامه ‌هاي كاربردي وب را معرفي مي‌كند.

در آخر، يك نكته ديگر درباره مدل‌سازي: بايد بين منطق تجاري و منطق نمايش به وضوح فرق گذاشت. براي بسياري از برنامه ‌هاي كاربردي تجاري، تنها منطق تجاري بايد در مدل ADM بيايد. جزييات مربوط به نمايش، مانند دكمه‌هاي متحرك، اجزاء پرنده و ساير اجزاي واسط كاربر به طور معمول متعلق به ADM نيستند. اگر احتياج به مدل كردن آنها بود، بايد مدل مجزايي براي واسط كاربر درنظر گرفته شود.

# 4-2-4- ملحقات وب كنالن

صفحات وب، چه اسكريپتي و چه كامپايل شده، در تناظر يك به يك با اجزاي UML هستند. يك جزء، يك جزء فيزيكي و قابل تعويض سيستم است. مدل پياده سازي سيستم اين اجزا و روابط آنها با يكديگر را نشان مي‌دهد. در يك برنامه كاربردي‌ وب، اين مدل نشان‌دهنده تمام صفحات وب سيستم و رابطه آنها با يكديگر است. در يك سطح، اين مدل شبيه يك نقشه سايت است.

از آنجا كه اجزا فقط نمايش دهنده اجزاي فيزيكي سيستم هستند، استفاده از آنها براي مدل‌كردن همكاري‌هاي درون صفحات مناسب نيست. اين سطح از انتزاع (كه مورد نياز طراح و كدنويس سيستم هست) بايد بخشي از مدل باشد. براي شروع مي‌توانيم بگوييم هر صفحه وب يك كلاس UML در مدل منطقي سيستم است، و روابط و پيوستگي‌هاي آن با ساير كلاس‌ها لينك‌ها را نشان مي‌دهند. ولي اين انتزاع به مشكل برمي‌خورد، وقتي به ياد بياوريم كه يك صفحه وب مي‌تواند شامل مجموعه‌اي از توابع قابل اجرا روي سرويس‌دهنده،‌ و مجموعه‌ي كاملا مجزايي از توابع قابل اجرا روي سرويس‌گيرنده باشد. صفحاتي مانند ASPها كه از اسكريت براي ساختن بخش‌هاي پوياي صفحه (اسكريپت‌هاي طرف سرويس‌گيرنده) به عنوان خروجي ASP استفاده مي‌كنند،‌از اين گونه‌اند. عكس‌العملي كه در برابر اين مشكل مي‌توانيم صورت ‌دهيم،‌ تعريف Stereotypeهايي براي مشخص‌كردن اينكه هركدام از اعمال درون صفحه بر روي سرويس‌دهنده معتبرند يا سرويس‌گيرنده، مي‌باشد. ولي اين روش پيچيدگي مدل را افزايش مي‌دهد، همان چيزي كه قصد داشتيم با مدل‌سازي از آن جلوگيري كنيم.

يك راه‌حل بهتر، جداكردن تكه‌هاي مختلف است. در نگاه منطقي، رفتار صفحه در طرف سرويس‌دهنده كاملا با رفتار او در طرف سرويس‌گيرنده متفاوت است. صفحه در هنگام اجرا بر روي سرويس‌دهنده، به منابعي نظير پايگاه داده، فايل‌ها و . . .) دسترسي دارد، در حالي كه وقتي همان صفحه به سرويس‌گيرنده مي‌رسد، رفتاري كاملا متفاوت و روابطي ديگر با مرورگر (از طريق مدل DOM) و ساير منابع طرف سرويس‌گيرنده دارد.

صفحات طرف سرويس‌دهنده را در يك كلاس و صفحات طرف سرويس‌گيرنده را در يك كلاس ديگر نشان داده مي‌شوند، و براي هركدام يك Stereotype اختصاص مي‌يابد. در ملحقاتي كه در اين پروژه استفاده شده‌اند، نمادهاي گرافيكي براي اين Stereotypeها ايجاد شده است. به اين ترتيب، براي يك صفحه وب كه در هر دو طرف سرويس‌گيرنده و سرويس‌دهنده كارهايي انجام مي‌دهد، دو كلاس تعريف مي‌شوند كه با هم ارتباط دارند: يكي ديگري را مي‌سازد. براي اين نوع رابطه بين كلاس‌ها هم يك Stereotype به نام Build تعريف مي‌شود.

**4-3- معرفي روش ICONIX**

در اين روش به عنوان يك راه ميانه بين روش‌هاي حداقلي (مانند XP) و روش‌هاي مبسوط (مانند RUP) توسط راجر روزنبرگ و كندال اسكات طراحي شده است. اين روش كه عنوان كامل آن Applied Use C*ase* DrivenObject Modeling است، با استفاده از مجموعه‌اي از نمودارهاي پيشنهادي UML و با محور قرار دادن مدل مورد كاربرد (UseCase) به تبيين مراحل توسعه نرم‌افزار مي‌پردازد.

**4-2-1- نقشه كلي ICONIX**

مراحل اين روش در شكل پنج نشان داده شده‌اند.

**شكل 5 -** نقشه كلي مراحل ICONIX ‎[12]



**ايستا**

(Static)

**پويا (**Dynamic)

مدل دامنه

مدل كلاس

نمودار ترتيب

نمودار تقويت

مدل مورد كاربرد

پيش نمايش

در اين روش كار با گفت و گو با كاربر و ثبت اظهارات وي، و نيز بررسي سيستم‌هاي نرم‌افزاري جاري آغاز مي‌شود. بر پايه اين مشاهدات، مدل‌هاي مورد كاربرد تهيه مي‌شوند و سناريوهاي آنها نوشته مي‌شود.

نمودارهايي كه در اين روش بايد رسم شوند در دو دسته كلي ديناميك و استاتيك قرار مي‌گيرند. نمودارهاي ديناميك (مدل مورد كاربرد، robustness diagram و نمودار توالي) گردش داده‌ها و ارتباطات اشياء با هم را نشان مي‌دهند (معادل DFDها در روش ساختيافته). در حالي كه نمودارهاي استاتيك (مدل دامنه و مدل كلاس) خصوصيات هر يك از اشيا و ارتباطات آنها با هم در حال سكون (مانند توارث) را نشان مي‌دهند. اين نمودارها را مي‌توان معادل ERDها در مدل ساختيافته دانست (هرچند چيزهاي بيشتري را در خود ثبت مي‌كنند).

پس از آنكه سناريوها نوشته شد، بايد مدل توالي اشياء براي هر سناريو رسم شود. براي پركردن فاصله موجود بين اين دو مرحله، از نمودارهاي تقويت استفاده مي‌شود. همزمان با اين مراحل، مدل دامنه سيستم رسم مي‌شود. اين مدل بر اساس دانش كسب شده در طي رسم نمودارهاي تقويت درباره اشياء حاضر در سيستم، اصلاح مي‌شود و به روز مي‌شود. سرانجام بر اساس مدل دامنه و نمودارهاي توالي، مدل كلاس سيستم رسم شده در اختيار تيم برنامه‌نويسي قرار مي‌گيرد.

**4-4- گام هاي Web-ICONIX**

در اين بخش فازهاي كار ساخت برنامه كاربردي وب در مدل فرآيند Web-ICONIX و قدم هاي لازم در هر فاز ارائه مي‌گردد. در اينجا از تشريح جزييات سه روش مورد استفاده (RUP، ICONIX، UML Web Extensions) خودداري شده و تنها ترتيب مراحل و تغييرات آنها نسبت به روند استاندارد مورد توجه قرار گرفته است. منابع كامل تر در باره هريك از سه موضوع فوق در بخش مراجع معرفي شده است.

4-4-1- فاز آغاز

تهيه گزارش نماي كلي، شامل:

* تعريف پروژه و اهداف آن،
* فهرست بازيگران سيستم ،
* رسم مدل مورد كاربرد،
* نوشتن متن سناريوهاي موارد كاربرد اصلي،
* طرح اوليه فرآيند تجاري[[55]](#footnote-56) برنامه (بوسيله نمودارهاي فعاليت استاندارد UML)،
* طرح اوليه صفحات اصلي برنامه (پيش نمايش روي كاغذ يا به شكل صفحات HTML)،
* تخمين هزينه و زمان انجام پروژه (در اينجا فرض بر اين است كه آشنايي قبلي با روش‌هاي تخمين هزينه و زمان پروژه وجود دارد، و بنابراين به آن پرداخته نمي‌شود)،
* تخمين اوليه ريسك ها (در اين زمينه نيز جزييات به روش‌هاي تخمين ريسك احاله مي‌گردد و در اينجا مورد بحث نيست)،
* برنامه زمان بندي اوليه كه فازها و تكرار ها را نشان مي دهد.

اين گزارش براي دست يافتن به توافق اوليه درباره حدود سيستم يا انعقاد قرار داد مورد استفاده قرار مي گيرد. در شكل شش مراحل لازم در اين بخش مشخص شده‌اند.

**تحليل**

- تعريف

- اهداف

- مدل مورد كاربرد

- سناريو اوليه

**طراحي**

- فرآيند تجاري

**پياده سازي**

- پيش نمايش (روي كاغذ يا روي صفحه مرورگر)

**توافق**

- هزينه

- زمان

- برنامه

**شكل 6 -** مدل آبشاري تكرارهاي فاز آغاز

4-4-2- فاز جزييات

**تهيه گزارش معماري سيستم شامل:**

* موارد كاربرد تكميل شده
* معماري برنامه :
  + لايه ها در هر دو طرف سرويس گيرنده و سرويس دهنده، از جمله: لايه مرورگر كاربر، لايه نمايش، لايه منطق تجاري، پايگاه داده، منابع چندرسانه‌اي سرويس دهنده و... .
  + اجزاي هر لايه (بدون نياز به رسم دقيق مدل هاي جزيي، تنها اجزاي اصلي هر لايه و ارتباطات آنها با هم مشخص مي شود) .
  + تكنولوژي‌هاي انتخابي براي هر لايه.
  + معماري مسيرياب[[56]](#footnote-57) سايت
* بستر سازي:
  + ارتباطات سيستم با ساير سيستم‌هاي تحت وب: اطلاعاتي كه بايد به شكل برخط با ساير سيستم ها مبادله شوند (مانند اطلاعات مالي، اطلاعات آب و هوا، فهرست پرواز هواپيماها و...) و ارتباطات B2B به‌همراه الزامات فني هرمورد دراين بخش مشخص مي‌گردند.
  + نيازهاي مربوط به ميزباني وب: بر اساس نيازهاي برنامه، نحوه ميزباني آن مشخص مي گردد. ممكن است نياز يك برنامه با داشتن فضايي محدود در يك سرويس دهنده مشترك برآورده شود، ولي برنامه ديگر به دليل ترافيك بالا، يا به دليل پردازش زياد، احتياج به آرايه اي از سرويس دهنده ها داشته باشد كه بار درخواست ها توسط يك توزيع كننده[[57]](#footnote-58) بين آنها پخش مي شود.
  + نيازهاي غيركاركردي ديگر (امنيت – دسترس پذيري و ...)

**تحليل**

- مدل مورد كاربرد

**طراحي**

- معماري لايه‌ها

- معماري مسيرياب

**بستر سازي**

- ارتباطات B2B

- معماري ميزبان روي وب

**شكل 7 -** مدل آبشاري تكرارهاي فاز جزييات

4-4-3- فاز ساخت

غالبا بيش از نيمي از زمان پروژه در اين فاز صرف مي شود. به دليل در هم تنيدگي مراحل تحليل و طراحي و حتي پياده سازي صفحات وب، بار اصلي اين مراحل در تكرارهاي فاز ساخت متمركز شده است. براي اين فاز (براي برنامه هاي كاربردي وب در گستره متوسط) غالبا دو يا سه تكرار مناسب است. مراحل كار در اين فاز در هر تكرار براساس نقشه مسير ICONIX پيش مي رود.

* طراحي برنامه طي سه مرحله كه بايد به طور موازي انجام شوند تكميل مي گردد:
  + مدل هاي ديناميك سيستم:
    - تكميل مدل مورد كاربرد و نوشتن دقيق همه سناريوها
    - رسم نمودارهاي تقويت. در اين نمودارها صفحات HTML موجوديت هاي مرزي و صفحات اسكريپت دار (ASP يا JSP يا ...) موجوديت هاي كنترلي را تشكيل مي دهند. جداول پايگاه داده يا منابع چندرسانه اي با موجوديت هاي داده اي نشان داده مي شوند.
    - رسم نمودار هاي توالي هر كدام از موارد كاربرد
  + مدل هاي استاتيك سيستم:
    - مدل پايگاه داده سيستم: براي رسم اين مدل (كه جزو مدل هاي توصيه شده در ICONIX نيست) از مدل هاي فيزيكي ابزار Rational Rose استفاده كرد . همچنين مي توان براي سرعت كار از ابزارهاي گزارش گيري داخلي پايگاه داده استفاده نمود.
    - مدل كلاس سيستم: تنها شامل كلاس‌هايي است كه لايه منطق تجاري سيستم را مي سازند و بر اساس نمودارهاي تقويت و توالي، با نمودارهاي استاندارد UML رسم مي شود.
    - مدل تحليل/طراحي سيستم (ADM): اين مدل با استفاده از ملحقات وب آقاي كنالن رسم مي شود. همه جزييات مربوط به ارتباط صفحات وب (چه در طرف سرويس دهنده و چه در طرف سرويس گيرنده) در اين مدل نشان داده مي شوند.
  + طراحي گرافيكي شماي سايت: بر اساس معماري مسيرياب سايت، طراحي شماي سايت در برنامه هاي گرافيكي انجام مي شود و پيش نمايش آن به صورت تصاويري يكپارچه ارائه مي گردد.
* آماده سازي سكوي محلي: بر اساس معماري ميزبان تعيين شده در فاز جزييات، سكوي محلي براي مرحله پياده سازي آماده مي‌شود. اين مرحله شامل تهيه و نصب سخت افزار و نرم افزارهاي مورد نياز و تنظيم سرويس دهنده هاي مورد نياز مي گردد.
* پياده سازي:
  + كدنويسي صفحات: بر اساس تكنولوژي هاي انتخاب شده در فاز جزييات صورت مي گيرد.
  + ايجاد صفحات واسط كاربر: طرح گرافيكي تاييد شده توسط ابزارهاي خودكار به كد HTML تبديل مي شود و پس از آن كدهاي توليد شده توسط طراح صفحات بهينه مي گردند.
  + تلفيق: كدهاي نوشته شده با صفحات واسط كاربر تركيب و روي سكوي محلي نصب مي شوند.
* تست محلي

**تحليل**

- مدل مورد كاربرد

- نمودار تقويت

- نمودار ترتيب

**طراحي**

- مدل پايگاه داده

- مدل كلاس

- مدل تحليل/طراحي

- طراحي گرافيكي

**بستر سازي**

- آماده سازي سكوي محلي

**پياده سازي**

- كدنويسي منطق تجاري

- ساخت واسط كاربر

- تلفيق

**شكل 8 -** مدل آبشاري تكرارهاي فاز ساخت

**تست محلي**

4-4- فاز انتقال

در اين مرحله سيستم به سكوي اصلي بر روي ميزبان اينترنت منتقل مي گردد و پس از تست نهايي به صاحب سيستم تحويل داده مي شود. غالبا پس از قرار گرفتن برنامه در بستر اصلي، نياز به يك يا چند تكرار ديگر براي بهينه سازي برنامه وجود دارد، كه مراحل آن مانند تكرارهاي فاز ساخت (با تاكيد بيشتر بر مراحل پياده سازي و تست) خواهد بود.

5- ساخت يك نمونه برنامه كاربردي وب با روش جديد

در اين فصل مراحل ساخت يك نمونه برنامه كاربردي وب با استفاده از مدل فرآيند ارائه شده در اين پروژه تشريح شده است. برنامه كاربردي مورد نظر، سيستمي براي ايجاد و مديريت صفحات شخصي كاربر است، كه قرار است در يك دانشگاه براي ايجاد صفحات شخصي وب اساتيد به كار رود. اين سيستم داراي دو بخش اصلي است:

* صفحات شخصي اساتيد، كه در سايت وب دانشگاه قرار مي‌گيرد،
* صفحات مديريتي، كه از طريق آن ويرايش و به روزرساني و ساير عمليات مديريتي سايت انجام مي‌شود.

**5-1- فاز آغاز**

## در اين فاز با تهيه گزارش نماي كلي پروژه، حدود آن تعيين مي‌شود. با استفاده از اين گزارش توافق اوليه بين سفارش دهنده سيستم (در اينجا مركز شبكه و انفورماتيك دانشگاه) و مجري پروژه درباره ابعاد سيستم صورت مي‌گيرد. همچنين در پايان اين سند برنامه زمان‌بندي و نيز تخمين هزينه و زمان اوليه پروژه ارائه مي‌شود.

# گزارش نماي كلي

# سيستم مديريت صفحات شخصي كاربر

# تعريف

اين سيستم امكان ايجاد صفحات شخصي وب به دو زبان فارسي و انگليسي براي اساتيد دانشگاه و به روزرساني آن توسط خود ايشان را فراهم مي‌سازد. صفحات شخصي شامل مشخصات و نشاني تماس، تصوير چهره، زمينه‌هاي پژوهشي، دروس ارائه شده، اسامي دانشجويان و فهرست سمينارها مي‌باشند.

# اهداف

* انتشار اطلاعات پژوهشي اساتيد دانشگاه به صورت دوزبانه
* فراهم شدن امكان به روزرساني اطلاعات پژوهشي

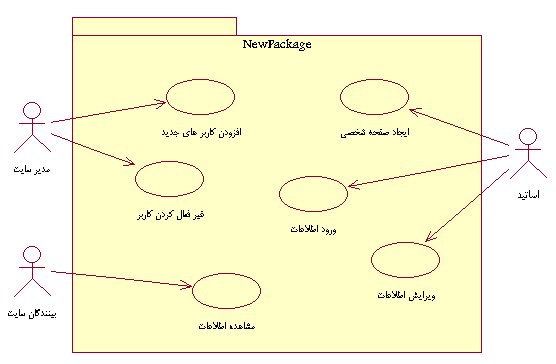
# بازيگران سيستم

* اساتيد: مي‌توانند براي خود صفحه شخصي ايجاد كنند، اطلاعات را وارد آن سازند و پس از آن نيز توانايي ويرايش اطلاعات و افزودن داده‌هاي جديد را داشته باشند.
* بينندگان سايت: از طريق فهرستي از صفحات شخصي ايجاد شده، به محتويات صفحات دسترسي خواهند داشت.
* مدير سايت: امكان ايجاد كدكاربري براي اساتيد و يا غيرفعال كردن كدهاي كاربري موجود را خواهد داشت.

**مدل مورد كاربرد سيستم**

اين مدل بر اساس اظهارات صاحبان سيستم، و نيز بر اساس برنامه فعلي (در صورت وجود) رسم مي‌شود. در اين مرحله فعاليت‌هاي اصلي هر بازيگر مشخص مي‌گردد.

**شكل 9 -** مدل مورد كاربرد اوليه



1-1

## 1-2

1-3

3-1

3-2

2-1

**سنار**يو**ی موارد کاربرد**

1-1- ايجاد صفحه شخصي

* + كاربر كدكاربري و كلمه عبور خود را وارد مي كند.
  + كاربر صفحه شخصي خود را فعال مي‌كند

**استثنا:**

* + در صورت غلط بودن كد كاربري، به كاربر پيغام خطا داده مي‌شود.

1-2- ورود اطلاعات:

* + كاربر كد كاربري و كلمه عبور خود را وارد مي كند
  + كاربر بخش مورد نظر را انتخاب مي كند
  + كاربر اطلاعات مورد نظر را وارد مي كند و دكمه ثبت را مي زند

**استثنا:**

* + در صورت غلط بودن كد كاربري، به كاربر پيغام خطا داده مي‌شود.
  + در صورت معتبر نبودن اطلاعات وارده به كاربر پيغام خطا داده مي‌شود.

1-3- ويرايش اطلاعات

* + كاربر كد كاربري و كلمه عبور خود را وارد مي كند
  + كاربر بخش مورد نظر را انتخاب مي كند
  + كاربر اطلاعات مورد نظر را ويرايش مي كند و دكمه ثبت را مي زند

**استثنا:**

* + در صورت غلط بودن كد كاربري، به كاربر پيغام خطا داده مي‌شود.
  + در صورت معتبر نبودن اطلاعات وارده به كاربر پيغام خطا داده مي‌شود.

2-1- مشاهده سايت:

* + بيننده سايت وارد صفحه اصلي سايت دانشگاه مي شود
  + بيننده سايت از فهرست صثفحات شخصي، صفحه مورد نظرش را انتخاب مي‌كند.
  + بيننده سايت صفحه شخصي مورد نظرش را مشاهده مي‌كند.

3-1- افزودن كاربر جديد

* + مدير سايت كد كاربري و كلمه عبور خود را وارد مي‌كند.
  + مدير سايت كدكاربري، كلمه عبور و نام كاربر جديد را وارد مي‌كند.
  + مدير سايت دكمه ايجاد كدكاربري را فشار مي دهد.

3-2- غيرفعال كردن كاربر

* + مدير سايت كد كاربري و كلمه عبور خود را وارد مي‌كند.
  + مدير سايت كاربر مورد نظر را از فهرست كاربران انتخاب مي‌كند.
  + مدير سايت دكمه غيرفعال كردن كاربر را فشار مي‌دهد.

**طرح اوليه صفحات اصلي**

سيستم صفحات شخصي كاربر

#### فهرست صفحات شخصي

* احمد احمدي
* رضا رضايي
* هاشم هاشمي

**مديريت صفحات**

**مديريت سايت**

صفحه شخصي احمد احمدي

#### استاديار

آدرس – ايميل

تصوير

زمينه‌هاي پژوهشي

......

......

اسامي دانشجويان

**منو**

اطلاعات شخصي

زمينه‌هاي پژوهشي

دروس

دانشجويان

سمينارها

احراز هويت

كدكاربري

كلمه عبور

مديريت صفحات

#### ويرايش اطلاعات شخصي

* نام:
* عنوان
* آدرس
* تلفن
* پست الكترونيك

**منو**

ويرايش اطلاعات شخصي

ويرايش زمينه‌هاي پژوهشي

افزودن زمينه‌هاي پژوهشي

ويرايش دروس

افزودن دروس

ويرايش دانشجويان

افزودن دانشجويان

ويرايش سمينارها

افزودن سمينارها

احراز هويت

كدكاربري

كلمه عبور

مديريت سايت

#### فهرست صفحات شخصي

* احمد احمدي
* رضا رضايي
* هاشم هاشمي

**فعال/غيرفعال**

#### افزودن كاربر جديد

نام

كد كاربر

كلمه عبور

افزودن

**شكل 10 -** طرح اوليه صفحات

**برنامه زماني**

براي ساخت اين سيستم دوماه زمان براي يك تيم سه نفره تخمين زده مي‌شود. اين تيم شامل يك نفر تحليلگر و طراح، يك نفر برنامه نويس وب و يك گرافيست است. بر اين اساس، برنامه زماني زير براي انجام پروژه پيشنهاد مي‌شود.

###### فاز انتقال

###### فاز ساخت

فاز

جزييات

فاز

آغاز

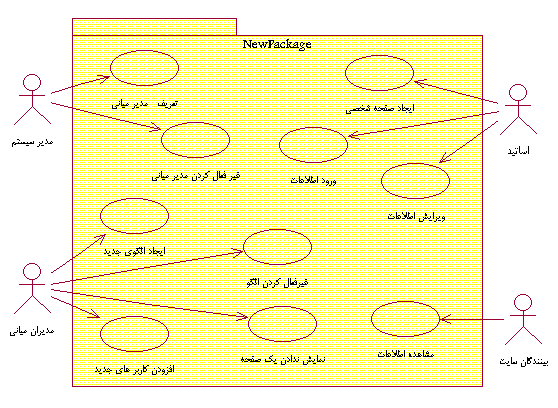
20 روز (يك تكرار) يك ماه (دو تكرار) يك هفته 3 روز

**شكل 11 -** برنامه زمان‌بندي پروژه

**5-2- فاز جزييات**

### مدل مورد كاربرد

در اين فاز در مرحله نخست مدل مورد كاربرد كه در فاز آغاز تهيه شده بود، مورد بازبيني و تصحيح و تكميل قرار مي‌گيرد.



**شكل 12 -** مدل مورد كاربرد فاز جزييات

**مدل معماري لايه‌ها**

اين مدل بر اساس ويژگي‌هاي برنامه، براساس توضيخات بيان شده در بخش 2-4 رسم شده است. همچنين در اين مدل تكنولوژي‌هاي انتخابي براي هر لايه ذكر گشته.

نمايش

(صفحات HTML)

مرور گر

(IE 5+)

منطق تجاري

(صفحات ASP)

پايگاه داده ها

(Ms Access)

HTTP

HTTP

**سرويس يرنده (Client) سرويس دهنده (Server)**

اينترنت

صفحات شخصي

صفحات مديريتي

**شكل 13 -** معماري لايه‌هاي نرم‌افزار

انتخاب تكنولوژي بر اساس نيازهاي برنامه، هزينه، زمان، و سهولت كاربرد هريك تعيين شده است. ابتدا با توجه به هزينه كمتر و سهولت نگهداري ميزبان داراي سكوي ويندوز (نسبت به سكوي لينوكس كه انتخاب ديگر بود)، اين سكو انتخاب شد.

سپس با توجه به اينكه سيستم مورد نظر داراي كاربران و ترافيك زيادي نيست، استفاده از پايگاه داده Access براي آن، با توجه به سادگي انتقال آن به ميزبان اصلي، و نيز هزينه كمتر براي ميزباني آن، توصيه شده است. انتخاب‌هاي ديگري كه در اين زمينه وجود داشت، MS SQL Server و اوراكل و MySQL بودند.

براي لايه مياني، با توجه به مهارت برنامه‌نويس تيم و نيز در دسترس بودن ابزارهاي كار با ASP، استفاده از اين تكنولوژي انتخاب شده است. ساير گزينه‌هاي مطرح در اين لايه عبارت بودند از PHP (كه بر روي سكوي لينوكس كار مي‌كند) و JSP (كه در بستر J2EE كار مي‌نمايد).

براي لايه آخر هم استفاده از HTML به دليل سادگي و پر كاربردبودن و مهارت برنامه نويس تيم در آن انتخاب شد. گزينه ديگر، XHTML بود كه براساس XML ايجاد شده است.

**معماري مسيرياب سايت**

مسيرياب سايت از يك منوي يك سطحي تشكيل شده كه در صفحات فارسي در سمت راست و در صفحات انگليسي در سمت چپ صفحه قرار مي گيرد و شامل اين گزينه‌ها است:

* اطلاعات شخصي
* زمينه‌هاي پژوهشي
* دروس
* دانشجويان
* سمينارها

**بستر سازي**

در اين مرحله مقدمات لازم براي انتقال محصول نهايي بروي ميزبان اينترنت پيش‌بيني مي‌شود، تا در فاز بعد همزمان با ساخت محصول، آماده سازي آن نيز انجام گيرد.

**ارتباط با ساير سيستم‌ها**

بخشي از اطلاعات صفحات شخصي بايد با وب سايت دانشگاه مبادله شوند:

* اطلاعات دانشجويان
* اطلاعات دروس
* اطلاعات سمينارها

### ميزباني وب

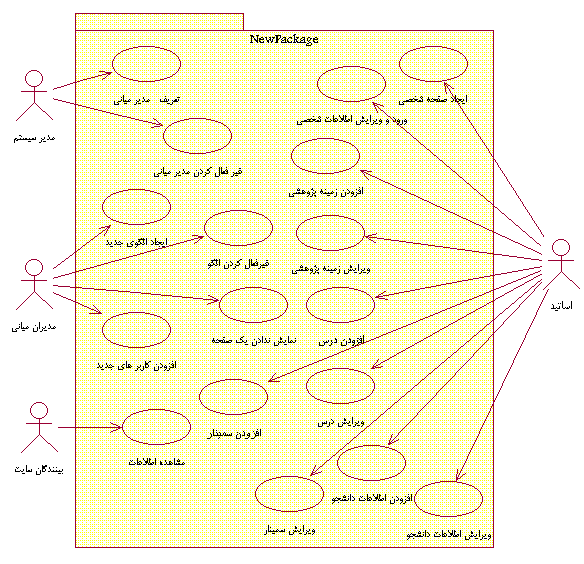
ميزبان اين سايت بايد داراي قابليت‌هاي زير باشد:

* پشتيباني از اسكريپت‌هاي ASP
* پشتيباني از ODBC براي ارتباط با پايگاه داده Access
* پشتيباني از زبان فارسي (استاندارد UTF8) براي ارتباط با پايگاه داده

**5-3- فاز ساخت**

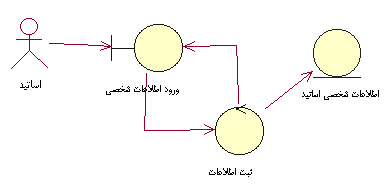
### مدل مورد كاربرد

در اين فاز مدل مورد كاربرد كه در فاز جزييات تكميل شده بود، مورد بررسي نهايي قرار مي‌گيرد و همه موارد كاربرد (چه كلي و چه جزيي) به آن افزوده مي‌گردد.



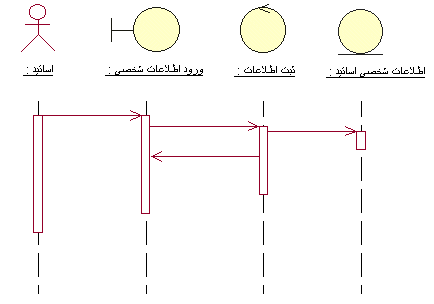
**شكل 14 -** مدل مورد كاربرد اصلاح شده

### نمودارهاي تقويت



**شكل 15 -** يك نمونه از نمودارهاي تقويت

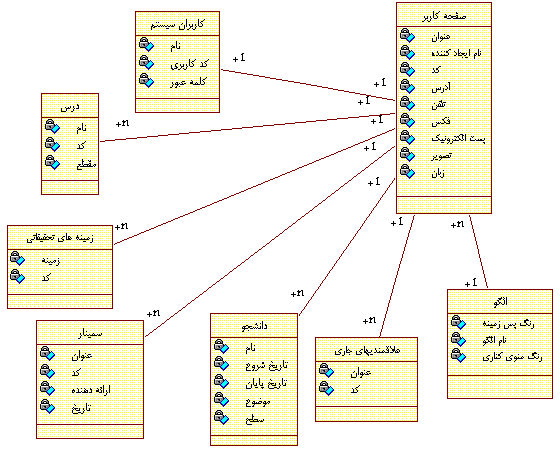
### نمودارهاي ترتيب



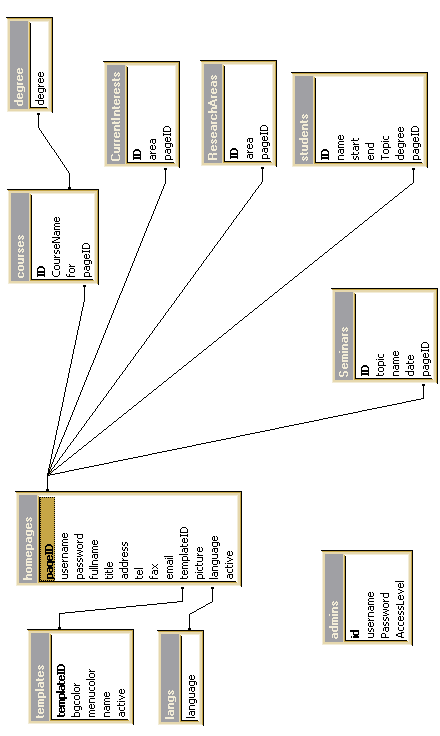
**شكل 16 -** يك نمونه از نمودارهاي ترتيب

### مدل كلاس سيستم

اين مدل بر اساس جزييات بدست آمده از نمودارهاي تقويت و ترتيب رسم شده است.

**مدل پايگاه داده**

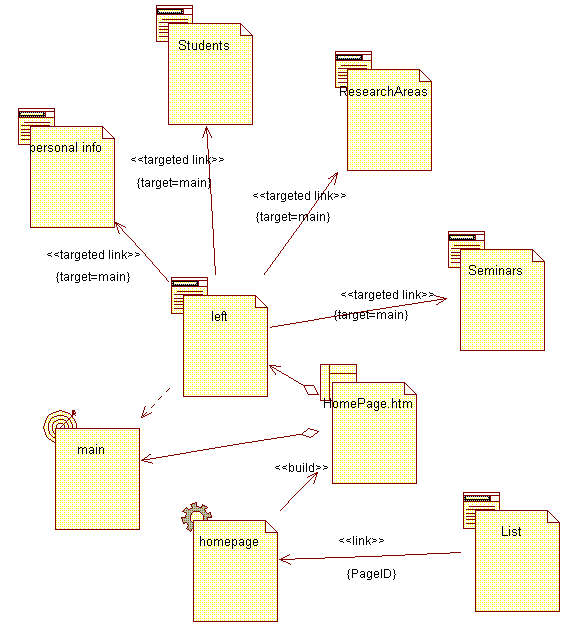
**شكل 17 -** مدل كلاس سيستم



**شكل 18 -** مدل پايگاه داده

### مدل تحليل/طراحي

اين مدل بر اساس جزييات بدست آمده از نمودارهاي تقويت و ترتيب و نيز مدل كلاس سيستم و با استفاده از ملحقات وب كنالن رسم شده است.



**شكل 19 -** مدل تحليل/طراحي سيستم

**پياده سازي**

پياده سازي سيستم با استفاده از ابزار EditPlus كه برنامه‌اي رايگان است انجام شد. در مرحله پياده سازي از Personal Web Server محصول مايكروسافت به عنوان سرويس دهنده وب استفاده گرديد.

**6- ارزيابي**

از آنجا كه چارچوب فرآيند RUP شامل توصيه‌ها و روش‌هاي دقيقي براي مراحل مختلف فرآيند ساخت نرم‌افزار است باقي ماندن در اين چارچوب سازندگان نرم‌افزار را از پشتوانه قدرتمندي در مواجهه با شرايط متفاوت برنامه‌هاي كاربردي وب گوناگون برخوردار مي‌سازد. از سوي ديگر در روش جديد سعي شده است تا با معرفي مجموعه كمينه‌اي از مستندات ضروري، هزينه مالي و زماني پروژه‌هاي تحت وب كاهش يابد. بر اين اساس براي فاز ساخت نرم‌افزار، استفاده از فرآيند ICONIX توصيه شده، و اين فرآيند نيز براي سازگاري با نرم‌افزارهاي كاربردي وب مورد تصحيحاتي قرار گرفته است. اين تصحيحات براي پوشش دادن به دو مدل ضروري در برنامه‌هاي كاربردي وب (مدل پايگاه داده و مدل تحليل/طراحي) انجام گرفته‌اند. براي رسم مدل پايگاه داده از امكانات داخلي ابزارهاي مديريت پايگاه داده و يا از نمودارهاي فيزيكي UML و براي مدل تحليل/طراحي از ملحقات كنالن براي UML استفاده شده است.

از آنجا كه برنامه‌هاي كاربردي وب به سرعت در حال گسترش در سيستم هاي سنتي و به عنوان جزيي از ساير برنامه هاي كاربردي هستند، داشتن روشي كه به روش‌هاي ساخت ساير برنامه هاي كاربردي نزديك باشد هزينه توليد سيستم هاي مختلط (عادي و تحت وب) را از طريق صرفه جويي در زمان و هزينه آموزش نيروها و نيز بهره‌مندي از تجربيات اندوخته شده در آن روش‌ها، كاهش و به روند انجام پروژه سرعت خواهد داد. به دليل سرعت افزايش درخواست براي برنامه‌هاي كاربردي وب در اندازه‌هاي كوچك و متوسط، هر روشي كه در اين حيطه ارائه شود بايد ارضاي اين دو پارامت (هزينه و زمان) را مد نظر قرار دهد. در جدول يك مقايسه اي بين فرآيند Web-ICONIX با ساير روش هاي نام برده شده در اين مقاله انجام شده است.

تغييرات تكنولوژيكي در ابزارها و استانداردهاي پياده سازي برنامه‌هاي كاربردي وب، به شدت بر قابليت‌هاي كلي اين برنامه‌ها و در نتيجه بر مراحل طراحي و مدل سازي آنها تاثيرگذار است، از اين رو بهينه سازي مستمر اين مدل فرآيند اجتناب ناپذير خواهد بود. قدم‌هاي بعدي در تكميل اين مدل بررسي موردي سازگاري تكنولوژي‌هاي گوناگون با اين فرآيند و افزودن گام‌هاي لازم براي هر كدام است. در اولين قدم، اين كار بايد در مورد عمده‌ترين تكنولوژي‌هاي موجود بخصوص J2EE (از شركت سان) و .NET (از شركت مايكروسافت) انجام پذيرد.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **عنوان** | **متدولوژي** | **پايه** | **رده** | **تعداد مراحل**  **اصلي** | **قابليت**  **توسعه** | **قابليت استفاده**  **در سيستم هاي**  **مختلط** | **پوشش مدلها** |
| HDM | ساخت يافته | - | روش مدل سازي | 2 | \* | \* | مدل صفحات، مدل مسيرياب،  مدل پايگاه داده |
| RMM | ساخت يافته | HDM | فرآيند افزايشي | 7 | \* | \* | مدل صفحات، مدل مسيرياب،  مدل پايگاه داده،  واسط كاربر |
| OOHDM | شيءگرا | RMM  UML | فرآيند افزايشي | 4 | \*\*  (به دليل  استفاده از  UML) | \*\* | مدل صفحات، مدل مسيرياب،  منطق تجاري،  مدل پايگاه داده،  واسط كاربر |
| Web-ICONIX | شيءگرا | RUP  ICONIX  UML | فرآيند تكراري-  افزايشي | 4 | \*\*\*  (به دليل  استفاده از  UML و RUP) | \*\*\* | مدل صفحات، مدل مسيرياب،  منطق تجاري،  مدل پايگاه داده،  واسط كاربر |
| WebML | شيءگرا | XML | روش مدل سازي | 4 | \*\*\*  (به دليل استفاده از  XML) | \* | مدل صفحات، مدل مسيرياب،  واسط كاربر |

**جدول 1** – مقايسه روش هاي مختلف ساخت برنامه هاي كاربردي وب

**مراجع :**

1. Ambler, S., Be Realistic About the UML, 2001  
    (http://www.agilemodeling.com/essays/realisticUML.htm)
2. Baresi, L. , Garzotto, F. , and Paolini, p. , Extending UML for Modeling Web Applications,  
    HICSS34 , (http://dlib2.computer.org/conferen/hicss/0981/pdf/09813055.pdf)

1. Braun D., Sivils J., Shapiro A., Versteegh J., What is UML, Kennesaw State University   
    (<http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/index.htm>)
2. Conallen, J. , Modeling Web Application Architectures with UML, Communications of   
    ACM, October 1999 (http://www.compapp.dcu.ie/~renaat/gdip/p63-conallen.pdf)

1. Conallen , J. , Building Web Applications with UML, Addison-Wesley, 2000
2. Conallen , J. , Modeling Web-Tier Components, Software Development Magazine  
    January 2001 (http://www.sdmagazine.com/)

1. Fraternali, P. , Tools and Approaches for Developing Data-Intensive Web Applications:   
    A Survey , Electronic and Information Dept., Milan Polytechnics, 1999

1. Howard,W. , Using UML and Rational Rose for Data Modeling, 2000  
    (http://www.infoadvisors.com/articles/UML/UML\_RationalRose\_DM.htm)

1. Rational Software Corporation , Concepts: ProjectLifecycle,   
    Rational Unified Process Documents -1999

1. Rational Software Corporation, Whitepaper: The UML and Data Modeling, Rational   
    Website, 1999 (http://www.rational.com/products/whitepapers/101516.jsp)

1. Rational Software Corporation, Roadmap: Developing Web Solutions,   
    Rational Unified Process Documents , 1999

1. Rosenberg , D. , and Scott, K. , Driving Design with UseCase, Software  
    Development Magazine, 2001 (http://www.sdmagazine.com)
2. Standley, J. , Internet Architecture, 2001  
    (http://www.surfscranton.com/architecture/internet.htm)

1. Sweat, J. , Using PHP to Develop Three-Tier Architecture Applications - Part 1, 2001   
    (http://www.zend.com/zend/tut/tutsweatpart1.php)

1. Schwabe , D. , and Rossi, G. , An Object Oriented Approach to Web- Based Application   
    Design, Departamento de Informلtica. PUC- RIO, Brazil, 2000
2. Yaeger C., What are Web Services? , Deitel & Associates, Inc., 2002  
    )http://www.deitel.com(

1. Iteratively [↑](#footnote-ref-2)
2. Incrementally [↑](#footnote-ref-3)
3. Use-Case Model [↑](#footnote-ref-4)
4. Text-Only [↑](#footnote-ref-5)
5. Online [↑](#footnote-ref-6)
6. Scalability [↑](#footnote-ref-7)
7. Hypermedia [↑](#footnote-ref-8)
8. Component [↑](#footnote-ref-9)
9. N-Tiered [↑](#footnote-ref-10)
10. Hypertext Transfer Protocol [↑](#footnote-ref-11)
11. Carriage Return [↑](#footnote-ref-12)
12. Uniform Resource Locator [↑](#footnote-ref-13)
13. Transmission Control Protocol [↑](#footnote-ref-14)
14. Open Systems Interconnection [↑](#footnote-ref-15)
15. Secure Sockets Layer [↑](#footnote-ref-16)
16. eXtensible Markup Language [↑](#footnote-ref-17)
17. Cascade Style Sheet [↑](#footnote-ref-18)
18. Standard Generalized Markup Language [↑](#footnote-ref-19)
19. XML Parser [↑](#footnote-ref-20)
20. Document Type Definition [↑](#footnote-ref-21)
21. Remote Method Invocation [↑](#footnote-ref-22)
22. Simple Object Access Protocol [↑](#footnote-ref-23)
23. Common Object Request Broker Architecture [↑](#footnote-ref-24)
24. Java Serialization [↑](#footnote-ref-25)
25. Java Remote Method Protocol [↑](#footnote-ref-26)
26. Transport [↑](#footnote-ref-27)
27. Internet Iner-ORB Protocol [↑](#footnote-ref-28)
28. Stub [↑](#footnote-ref-29)
29. Skeleton [↑](#footnote-ref-30)
30. Interface Definition Language [↑](#footnote-ref-31)
31. Object Request Broker [↑](#footnote-ref-32)
32. Simple Object Access Protocol [↑](#footnote-ref-33)
33. Envelope [↑](#footnote-ref-34)
34. Header [↑](#footnote-ref-35)
35. Business to Business [↑](#footnote-ref-36)
36. Web Services Description Language [↑](#footnote-ref-37)
37. Universal Description , Discovery and Integration [↑](#footnote-ref-38)
38. Hypermedia Design Model [↑](#footnote-ref-39)
39. Relationship Management Methodology [↑](#footnote-ref-40)
40. Visual [↑](#footnote-ref-41)
41. Slices [↑](#footnote-ref-42)
42. Component [↑](#footnote-ref-43)
43. Navigation [↑](#footnote-ref-44)
44. Abstract [↑](#footnote-ref-45)
45. Design Pattern [↑](#footnote-ref-46)
46. Rational Unified Process [↑](#footnote-ref-47)
47. Notation [↑](#footnote-ref-48)
48. iteration [↑](#footnote-ref-49)
49. milestone [↑](#footnote-ref-50)
50. vision [↑](#footnote-ref-51)
51. collaborations [↑](#footnote-ref-52)
52. framework [↑](#footnote-ref-53)
53. Analysis/Design Model [↑](#footnote-ref-54)
54. Object Constraint Language [↑](#footnote-ref-55)
55. Business Model [↑](#footnote-ref-56)
56. Navigation Bar [↑](#footnote-ref-57)
57. Dispatcher [↑](#footnote-ref-58)