به نام خدا

دانشگاه صنعتی امیرکبیر

دانشکده ریاضی و علوم کامپیوتر

پایان‌نامه کارشناسی ارشد

رشته علوم کامپیوتر- گرایش سیستم‌های کامپیوتری

عنوان:

طراحی و پیاده‌سازی محیطی برای مهندسی معکوس در وب

تهیه:

صادق سلیمانی

اساتید راهنما:

دکتر محمدابراهیم شیری – دکتر سعید پارسا

دی‌ماه ۱۳۸۴

چکیده

مهندسی معکوس نرم‌افزار مشتمل بر ایجاد یک یا بیشتر سطوح تجرید با روش از پایین به بالا و به شیوه‌ی افزایشی است. این امر شامل یافتن ساختارهای سطح پایین پیاده‌سازی شده و سپس جایگزینی آن‌ها با همتاهای هم سطح یا سطح بالاتر است. هدف آن، تسهیل تغییرات، با ایجاد درک درباره‌ی ماهیت کار نرم‌افزار، چگونگی عملکرد آن و معماری‌اش است. با وجود قدمت و تعداد کمتر رویکردهای مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب نسبت به برنامه‌های کاربردی غیرمبتنی بر وب، روش‌های مختلفی تاکنون در این زمینه عرضه شده‌اند.

در این پایان نامه ابتدا به جمع‌آوری جامع مطالب مرتبط با مهندسی معکوس نرم‌افزار، خصوصیات برنامه‌های کاربردی مبتنی بر وب، رویکردهای مهندس معکوس این نوع برنامه‌ها، نحوه‌ی توزیع بارکاری کارگزاران وب و رویکردها و الگوریتم‌های توزیع بارکاری این کارگزاران پرداخته شده است. سپس بر اساس آن فرآیندی چند مرحله‌ای برای بازیابی معماری برنامه‌های کاربردی مبتنی بر وب و توزیع بارکاری آن‌ها بر چندین کارگزار، پیشنهاد، بررسی و پیاده‌سازی گردیده است.

رویکرد ارائه شده مبتنی بر گام‌های مختلفی است و در چهار مرحله صورت می‌پذیرد. این فرآیند ابتدا کد مبنای برنامه‌ی کاربردی مبتنی بر وب را به یک گراف چندگانه تبدیل می‌نماید و پس از تبدیل نمودن آن به گراف ساده، اعمال الگوریتم‌های خوشه‌بندی بر گراف ساده میسر می‌گردد و در نهایت بر اساس نتایج خوشه‌بندی، توزیع بار کاری برنامه‌ی مورد نظر، بر چندکارگزار انجام می‌شود.

این رویکرد دارای مزایایی از جمله عمومیت، خودکار بودن (تا حد امکان) و قابلیت توسعه است. ابزاری نیز به نام RELB برای آن ایجاد گردیده است و فرایند ایجاد ابزار نیز تشریح شده است تا امکان پیاده‌سازی به زبان‌های دیگر را نیز داشته باشد. همچنین یک مطالعه‌ی موردی نیز برای آزمایش آن صورت پذیرفته است.

کلمات کلیدی: مهندسی معکوس، نرم‌افزار کاربردی مبتنی بر وب، بازیابی معماری، توزیع بارکاری، خوشه‌بندی

فهرست مطالب (خلاصه)

فهرست مطالب (تفصیلی)

فهرست شکل‌ها

فهرست جداول

فصل اول

مقدمه

۱-۱- هدف

مهندسی معکوس نرم‌افزار، مشتمل بر ایجاد یک یا بیشتر سطوح تجرید[1] با روش از بالا به پایین یا پایین به بالاست. این امر شامل یافتن ساختارهای سطح پایین پیاده‌سازی شده و سپس جایگزینی آن‌ها با همتاهای هم سطح یا سطح بالاتر است. هدف آن، تسهیل تغییرات، با ایجاد درک درباره‌ی ماهیت کار نرم‌افزار، چگونگی عملکرد آن و معماری‌اش است. مزایای پی‌گیری این هدف، بازیابی اطلاعات برای تسهیل مهاجرت بین بسترها[2]، بهبود مستندات یا تأمین مستندات جدید، استخراج قطعات قابل استفاده‌ی مجدد، کاهش هزینه‌های نگه‌داری، غلبه بر پیچیدگی، ردیابی اثرات جانبی[3]، کمک به مهاجرت به یک محیط CASE و تولید نرم‌افزارهای مشابه رقیب است[۴].

با وجود قدمت و تعداد کمتر رویکردهای مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب[4] (که در ادامه ممکن است آن را به اختصار WA بنامیم) نسبت به برنامه‌های کاربردی غیرمبتنی بر وب، روش‌های مختلفی تاکنون در این زمینه عرضه شده‌اند.

ارائه‌ی رویکردی عمومی برای بازیابی معماری برنامه‌های کاربردی مبتنی بر وب با استفاده از فنون خوشه‌بندی و سپس توزیع بارکاری آن برنامه‌ها براساس رویکردی جدید و مرتبط با خوشه‌های بازیابی شده، هدف اصلی این پایان‌نامه است.

---

[1] Abstraction
[2] Platforms
[3] Side Effects
[4] Web Applications

۱-۲- مروری بر مطالب پایان‌نامه

۱-۲-۱- اهمیت تحقیق

در یک جهان ایده‌آل، تمام سیستم‌های نرم‌افزاری، چه در گذشته و چه در آینده بایستی با توجه به منافع مهندسی نرم‌افزار، به خوبی ساختاریافته، تولید و نگه‌داری شوند. اما در جهان واقعی بسیاری سیستم‌ها دارای طراحی ایده‌آل نیستند و علاوه بر آن دارای معایب ویژه‌ی خود نیز هستند، بدین ترتیب نگه‌داری و تغییرات آینده‌ی سیستم به سادگی امکان‌پذیر نخواهد بود و بنابراین باید ابزارها و روش‌شناسی‌هایی[1] برای مواجهه با این موارد وجود داشته باشد.

مهندسی معکوس[2] ریشه در تحلیل سخت‌افزار برای منافع تجاری یا نظامی دارد. هدف آن، تسهیل تغییرات با ایجاد درک درباره‌ی ماهیت کار نرم‌افزار، نحوه‌ی کار آن و معماری‌اش است (بخش ۲-۲-۱). به‌طور خلاصه می‌توان مهندسی معکوس نرم‌افزار را "تحلیل یک سیستم نرم‌افزاری برای تشخیص قطعات و وابستگی‌های آن‌ها، استخراج و ایجاد تجرید سیستم و اطلاعات طراحی" [۲] نامید. مهندسی معکوس مشتمل بر به‌کارگیری یک یا بیشتر سطوح تجرید با روش از بالا به پایین یا پایین به بالاست(شکل ۲-۱). این امر شامل یافتن ساختارهای سطح پایین پیاده‌سازی شده و سپس جایگزینی آن‌ها با همتاهای سطح بالاست. فرآیند مذکور احتمالاً به یک فرموله‌سازی معماری کلی برنامه نیز منجر خواهد شد. البته شایان توجه است که محصول مهندسی معکوس لزوماً به سطح تجرید بالاتری منجر نمی‌گردد. معمولاً متداول‌ترین نقطه‌ی شروع آن، کد مبنای برنامه است [۴]. یک دلیل ساده این است که سال‌ها پس از تولید، مشخصه‌ها و اطلاعات طراحی سیستم ممکن است دقیق یا حتی موجود نباشد.

رویکردهای متنوعی برای برای مهندسی معکوس سیستم‌های نرم‌افزاری وجود دارد، که برخی از آن‌ها به اختصار عبارتند از[۱۲] : مطالعه‌ی مستندات و کدمبنای کنونی، اجرای نرم‌افزار و تولید و تحلیل ردیابی‌های اجرا، مصاحبه با کاربران و تولیدکنندگان نرم‌افزار، استفاده از ابزارها و فنون متنوع، تحلیل تاریخچه‌ی ویرایش‌ها، ارزیابی یک سیستم نرم‌افزاری و کیفیت آن با کمک معیارهای نرم‌افزاری. که محافل علمی بیشتر در زمینه‌ی سه مورد آخر فعالیت داشته‌اند(بخش ۲-۲-۴).

در این میان برنامه‌های کاربردی تحت‌وب[3] به علت ویژگی‌های به‌خصوص حساسیت عمده‌ای دارند. توسعه سریع اینترنت و برنامه‌های کاربردی تحت وب عرصه‌ی جدیدی برای تحقیقات مهندسی نرم‌افزار را

---

[1] Methodologies
[2] Reverse Engineering
[3] Web Applications

فراروی قرار داد. آشنایی با تکنولوژی‌های جدید مانند زبان‌های اسکریپت[1] سبب تبادلات قوی‌تر جانب مشتری با جانب کارگزار[2] شد. این امر سبب تسهیل تولید برنامه‌های مقیاس‌بزرگ[3] گردید که توانایی‌های وب را بسیار گسترده‌تر کرد. لزوم تبادل داده با مشتری سبب ایجاد برنامه‌های کاربردی تحت وب گردید که این نوع برنامه‌ها بر خلاف وب‌سایت‌ها پویا هستند و با پایگاه داده در تماس می‌باشند و امکانی برای تغییر در وضعیت سایت توسط کاربر، در اختیار وی می‌گذارد. موتور اصلی و زیرساختی هر تجارت الکترونیکی همین برنامه‌های کاربردی تحت وب هستند[۸]. تعریف، دسته‌بندی و قطعات برنامه‌های کاربردی مبتنی بر وب و تفاوت آن‌ها با سایر برنامه‌های کاربردی در بخش ۲-۳ آمده است.

با توجه به پیچیدگی فزاینده‌ی برنامه‌های کاربردی تحت وب، جامعه‌ی نرم‌افزار نیاز به روش‌شناسی‌ها و فن‌آوری‌های[4] تواناتر برای فرایند تولید اصولی‌تر را گوشزد کرده است و روش‌شناسی‌های متعددی برای تولید وب‌سایت‌ها یا برنامه‌های کاربردی تحت وب در متون، پیشنهاد گردیده‌اند [۳]. اما فشار زیاد زمان کوتاه تولید برنامه‌های کاربردی تحت وب اغلب تولیدکنندگان را وادار می‌کند تا از رویکرد منظمی استفاده نکنند و اثر منفی سنگینی بر کیفیت و مستندسازی برنامه خواهد گذاشت. بدین ترتیب این برنامه‌ها معمولاً بدون تولید هیچگونه مستند مفیدی برای نگه‌داری و تغییر، تولید می‌شوند و کاستن از انعطاف‌پذیری مناسب، قابلیت نگه‌داری و تطابق‌پذیری[5] در جهت برآوردن نیازهای بازار صورت می‌گیرد[۸]. رسالت مهندسی معکوس با توجه به آن‌چه بیان گردید، برای برنامه‌های کاربردی تحت وب که عمر کمتری نیز نسبت به برنامه‌های کاربردی عادی دارند، بیش از پیش مشهود است.

با وجود قدمت و تعداد کمتر رویکردهای مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب، نسبت به سایر برنامه‌های کاربردی، روش‌های مختلفی تاکنون در این زمینه عرضه شده‌اند. دسته‌بندی رویکردهای مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب در مقالات معمولاً به دو گروه عمده اشاره دارد[۱]: گروه اول در جهت دستیابی به یک نما از معماری برنامه که قطعات (مانند صفحات یا قطعات صفحات داخلی) و روابط آن را در سطوح جزییات مختلف نمایش دهد، تلاش دارد. اما دیگری از فنون خوشه‌بندی برای خلاصه‌سازی نیازهای عملیاتی که توسط برنامه‌کاربردی مبتنی بر وب برآورده می‌شود و معمولاً با نمودارهای UML مدل می‌شود، استفاده می‌کند. بیشتر رویکردهای ارائه شده به شدت به زبان پیاده‌سازی وابسته هستند و برای اعمال آن‌ها به محصولاتی با زبان دیگر مشکلات عدیده‌ای وجود دارد(۲-۴).

---

[1] Script
[2] Server-Side
[3] Large-Scale
[4] Technologies
[5] adoptability

ارائه‌ی رویکردی عمومی برای بازیابی معماری برنامه‌کاربردی مبتنی بر وب، آن هم با استفاده از فنون خوشه‌بندی، نه تنها یک دسته کلی از رویکردهای مهندسی معکوس این برنامه‌ها را تحت تأثیر قرار خواهدداد، بلکه سبب خواهد شد تا از تجربیات به‌کارگیری فنون خوشه‌بندی که بیشتر در بازیابی غیر معماری مورد استفاده بوده است، در حیطه‌ی بازیابی معماری نیز بهره‌برداری بیشتری شود.

استفاده از مهندسی معکوس تنها به بازیابی معماری و تسهیل نگه‌داری محدود نمی‌شود و می‌توان از آن بهره‌های متعدد دیگری نیز برد. در اینجا از نتایج مهندسی معکوس برای ارائه‌ی الگوریتمی جدید در توزیع بارکاری[1] کارگزارن وب[2] استفاده شده است(فصل ۳). توزیع بارکاری در حالت کلی به پخش وظیفه‌ی یک ماشین بین چند ماشین، برای جلوگیری از اشباع ظرفیت خدمت‌دهی آن، اشاره دارد و روش‌های عمده‌ی آن عبارتند از توزیع در جانب کاربر، توزیع توسط توزیع کننده[3] و توزیع در جانب کارگزار. توزیع در جانب کاربر به علت نیاز به احاطه بر مرورگر مشتری و توزیع در جانب کارگزار به علت پیچیدگی و سربار زیاد، کارایی کمتری از توزیع در جانب توزیع کننده دارند. در یک دسته‌بندی کلی سه رویکرد زیر برای توزیع درخواست‌ها توسط توزیع کننده وجود دارد[۳۰]:

۱. سوییچینگ در لایه‌ی چهار به همراه هدایت[4] بسته در لایه‌ی دو (L4/2)

۲. سوییچینگ در لایه‌ی چهار به همراه هدایت بسته در لایه‌ی سه (L4/3)

۳. سوییچینگ در لایه‌ی هفت به همراه هدایت بسته در لایه‌ی دو یا سه (L7)

هر کدام از این رویکردها ممکن است یکی از سه الگوریتم زیر را برای توزیع درخواست‌ها استفاده کنند[۳۴]:

- الگوریتم نوبت گردشی[5]: درخواست‌ها به شیوه‌ی نوبت گردشی به کارگزارهای مختلف ارسال می‌شوند. در این شیوه به بارکاری کنونی کارگزارها توجهی نمی‌شود.
- الگوریتم کمترین ارتباط[6]: درخواست بعدی به کارگزاری ارسال می‌گردد که دارای کمترین تعداد ارتباطات HTTP است. این الگوریتم، یک الگوریتم زمان‌بندی پویا به حساب می‌آید زیرا که لازم است تعداد ارتباطات باز هر کارگزار را در نظر داشته باشد.

---

[1] Load Balancing
[2] Web Servers
[3] Dispatcher
[4] Forwarding
[5] Round Robin
[6] Least Connection

- الگوریتم کمترین بارکاری[1]: در این شیوه، کارگزار توزیع کننده، درخواست بعدی را به کارگزاری که دارای کمترین بارکاری است ارسال می‌کند. این الگوریتم لازم است اطلاعات زمان خدمت تمام درخواست‌های کاربران را داشته باشد. این اطلاعات اغلب در زمان رسیدن درخواست، نامشخص هستند و به این ترتیب دشواری‌های عمده‌ای در ارتباط با پیاده‌سازی این الگورتیم وجود دارد.

در این پایان‌نامه، پس از انجام مهندسی معکوس، از نتیجه‌ی آن برای افزودن الگوریتمی به الگوریتم‌های توزیع که از رویکرد L7 بهره می‌برد، استفاده خواهد شد.

۱-۲-۲- رویکرد پیشنهادی

محرز است که یکی از بهترین شیوه‌ها برای نمایش ساختار یک سیستم‌نرم‌افزاری، استفاده از گراف است، این امر نه‌تنها روش مناسبی برای نمایش موجودیت‌های مرتبط با هم است بلکه دارای پیش‌زمینه‌ی نظری قوی نیز هست و برخی الگوریتم‌های خوشه‌بندی هم برای کار با آن تطابق داده شده‌اند[۲۶].

روش ارائه شده مبتنی بر گام‌های مختلفی است که ابتدا کد مبنای برنامه‌ی کاربردی مبتنی بر وب را به یک گراف چندگانه تبدیل می‌نماید و پس از تبدیل نمودن آن به گراف ساده، الگوریتم خوشه‌بندی را بر آن انجام می‌دهد، سپس بر اساس نتیجه‌ی خوشه‌بندی، پیکربندی جدیدی برای توزیع درخواست‌های ارجاعی به برنامه‌کاربردی مبتنی بر وب، پیشنهاد می‌کند (شکل ۴-۱).

در مرحله‌ی اول ابتدا کد مبنای سیستم تجزیه[2] و تحلیل می‌گردد و مدلی از آن در قالب گراف‌چندگانه[3] نمایش داده خواهد شد. انتخاب گره‌ها در گراف‌چندگانه می‌تواند بر اساس دیدهای مختلفی به برنامه‌کاربردی مبتنی بر وب انجام گیرد و در هر دید می‌توان بر اساس معیارهای مشخصی به یال‌ها وزن داد که در این مطالب در بخش‌های ۴-۲-۱ تا ۴-۲-۴ تشریح شده‌اند. به‌طور خلاصه مدل انتخاب شده برای گراف چندگانه‌ای که نمایانگر برنامه‌کاربردی مبتنی بر وب است، صفحات وب از قبیل صفحات جانب مشتری و جانب کارگزار را به عنوان گره‌های گراف در نظر می‌گیرد و روابط بین آن‌ها از قبیل Link، Submit، Include و Redirect را لبه‌های گراف فرض می‌کند(شکل ۴-۲). با توجه به میزان انسجامی که

[1] Least Loaded
[2] Parsed
[3] Multigraph، تعمیمی از گراف‌هاست با این مشخصه که تعداد دلخواهی از لبه‌ها بین دو گره و حلقه (که عبارت است از اتصال یک گره به خودش) را اجازه می‌دهد.

این روابط بین دو صفحه ایجاد می‌کنند، به هر یک از آن‌ها وزنی منتسب می‌شود، برای مثال وزن Submit از Link بیشتر است، زیرا این رابطه همراه با تبادل داده از مرورگر مشتری به کارگزار نیز هست، در حالی که Link تنها برای تغییر جریان مرورگری از یک صفحه به صفحه‌ی دیگر به‌کار می‌رود. این نحوه‌ی انتخاب روابط و قطعات فراتر از زبان پیاده‌سازی یک برنامه‌کاربردی مبتنی بر وب است و تقریباً تمام این برنامه‌ها را می‌توان بر این اساس مدل نمود.

مرحله‌ی بعدی به‌کارگیری تحلیل خوشه‌بندی است. اما از آن‌جا که خوشه‌بندی قابل اعمال بر روابط چندگانه بین موجودیت‌ها نیست [۹] و معمولاً از یک مقدار تکی با عنوان *معیار تشابه یا عدم تشابه* استفاده می‌کند، لازم است این مقدار تکی به نحوی از نوع و وزن لبه‌ها استخراج گردد. به بیان دیگر لازم است که گراف‌چندگانه‌ی حاصل را به یک گراف معمولی تبدیل نماییم، که لبه‌های این گراف باید دارای مقادیر تکی منتسب شده باشند، که نشانگر تشابه بین گره‌هایی است که لبه به هم متصل کرده است. این امر در مرحله دوم (بخش ۴-۳) و بر اساس جمع وزنی لبه‌های بین دو گره صورت می‌گیرد.

الگوریتم‌های خوشه‌بندی را می‌توان به‌طور کامل بر نمایش گرافی سیستم اعمال نمود. حاصل، ساختاری خلاصه خواهد بود، که کل اطلاعات اولیه را به موازات اطلاعات ساختار در خود دارد. این گراف همان خوشه‌بندی نتیجه است. نکات قابل توجه در این‌جا عبارتند از الگوریتم خوشه‌بندی که از نوع سلسله‌مراتبی تجمعی انتخاب شده است زیرا ماهیت برنامه‌های کاربردی مبتنی بر وب دارای ساختار سلسله‌مراتبی است و قاعده‌ی به‌هنگام‌سازی که از نوع پیوند ساده انتخاب شده است و نقطه‌ی برش برابر با تعداد کارگزاران موجود در مزرعه‌ی کارگزاران مرتبط با توزیع درخواست‌هاست (بخش ۴-۴).

در نهایت بر اساس شیوه‌ی توزیع که توزیع درخواست‌ها را مد نظر دارد، یک کارگزار به عنوان کارگزار توزیع‌کننده در نظر گرفته می‌شود و یک کپی از برنامه‌کاربردی مبتنی بر وب بر هر کارگزار قرار داده می‌شود تا کارگزار توزیع کننده بر اساس درخواست‌های رسیده برای صفحات در لایه هفتم شبکه در سرآیند بسته‌ها در پروتکل HTTP، و با توجه به جدول نگاشتی که از مرحله‌ی خوشه‌بندی به دست آمده است، درخواست‌های متعلق به هر خوشه را به کارگزار مربوط ارسال نماید(بخش ۴-۵).

رویکرد کنونی توزیع‌بار، امکان آمارگیری از درخواست‌ها و تشخیص خوشه‌های پردرخواست و کم‌درخواست را میسر کرده است و براساس نتایج آن، پس از مدتی از کار سیستم، می‌توان با افزایش کارگزاران اختصاصی به خوشه‌های پردرخواست و ادغام خوشه‌های کم درخواست، بارکاری را به نحو هوشمندانه‌تری کنترل نمود. همچنین کپی تمام برنامه‌ی کاربردی مبتنی بر وب در هر خوشه سبب

جلوگیری از ارجاع بین خوشه‌ای و پیچیدگی‌های مرتبط با آن می‌گردد و به این ترتیب با پاسخ‌گویی محلی، روند سریع‌تری در دسترسی ایجاد شده است.

غیر از مرحله‌ی اول که تجزیه و تحلیل کد مبنای برنامه است، سایر موارد مستقل از برنامه هستند و تا حد امکان خودکار. این امر سبب می‌شود انعطاف فوق‌العاده‌ی روش‌ها و ابزار طراحی شده برای مراحل بعد خواهد شد و به رویکرد پیشنهادی، عمومیت بالایی نسبت به سایر روش‌های ارائه شده برای مهندسی معکوس نرم‌افزارهای کاربردی مبتنی بر وب داده است.

در این راستا در فصل پنجم ابزاری با نام RELB (Reverse Engineering and Load Balancing) به زبان جاوا ایجاد گردیده است که بر مراحل چهارگانه‌ی ذکر شده در فصل پیش منطبق است و برای هر مرحله از فرآیند مهندسی معکوس که در شکل ۴-۱ آمده است، ابتدا مفاهیم پیش‌نیاز ضروری بیان گردیده است و سپس به بیان جزییات پیاده‌سازی (به صورت شبه‌کد) و روند آن پرداخته شده است. در ارائه‌ی رویکرد و تهیه‌ی ابزار، معیارهای عمومیت، خودکارسازی و قابلیت توسعه، به عنوان معیارهای ارزیابی در نظر گرفته شده‌اند و پس از پیاده‌سازی آن، میزان وصول به هر یک از این اهداف بررسی شده است.

نتایج هر مرحله برای افزایش سرعت محاسبات در قالب فایل‌های متنی ذخیره می‌شوند و در اختیار مراحل دیگر قرار می‌گیرند. با فرض این‌که برنامه‌ی کاربردی مبتنی بر وب به زبان JSP پیاده‌سازی شده باشد، ابزار مذکور از ایده‌ی نمودار انتقال که برای ساخت تحلیل‌گر لغوی در کامپایلرها مورد استفاده قرار می‌گیرد، جهت استخراج روابط بین صفحات در مرحله‌ی اول استفاده نموده است. دو برنامه دیگر که پیاده‌سازی دو مرحله‌ی تبدیل و خوشه‌بندی را بر عهده دارند، بر اساس وزن منتسب شده به روابط و عدد وارد شده توسط کاربر برای تعداد خوشه‌ها، ساده‌سازی گراف چندگانه و خوشه‌بندی بر آن را مطابق با جزییات تشریح شده در رویکرد انجام می‌دهند و در نهایت با استفاده از امکانی به نام صافی[1] و با کمک نتیجه‌ی حاصل شده از خوشه‌بندی که در فایلی به نام Clusters.txt ذخیره شده است، عمل نگاشت درخواست‌ها به کارگزاران صورت می‌گیرد.

در نهایت یک برنامه‌کاربردی مبتنی بر وب به عنوان مطالعه‌ی موردی، در فصل ۶ مورد بررسی قرار گرفته و نتایج حاصل از به‌کارگیری ابزار بر آن عرضه شده است. نتایج حاصل شده، گردآمدن صفحات از لحاظ منطقی مرتبط در خوشه‌ها را تأیید می‌کنند.

از مزایای عمده‌ی رویکرد پیاده‌سازی شده، نسبت به روش‌های مشابه، می‌توان به عمومیت، نسبتاً خودکار بودن و قابلیت توسعه‌اش اشاره نمود. زیرا تنها مرحله‌ای از فرایند که به زبان و قلمرو برنامه وابسته

[1] Filter

است، مرحله‌ی اول است و نیز رویکرد مورد نظر را می‌توان تقریباً بر تمام برنامه‌های کاربردی تحت وب اعمال نمود، زیرا به استخراج روابطی از آن‌ها می‌پردازد که در تمام زبان‌های پیاده‌سازی برنامه‌های کاربردی مبتنی بر وب مشترک است. همچنین حجم برنامه نیز هرگز مانعی برای استفاده از این رویکرد نیست و رویکرد قادر است بر برنامه‌ی کاربردی مبتنی بر وب با هر اندازه‌ای اعمال شود.

از دلایل خودکار بودن آن می‌توان نیاز بسیار کم به اطلاعات و مستندات حیطه‌ی نرم‌افزار اشاره کرد و نیز مراحلی که با ارتباط با عامل انسانی برای تسهیل رویکرد انجام می‌گیرد، معدود است. گرچه بدیهی است که نمی‌توان نیاز به عامل انسانی را به طور مطلق از مراحل بازیابی معماری یا طراحی نرم‌افزار حذف نمود[۴].

نتایج تحقیقات جانبی در کنار این پایان‌نامه را نیز نباید دست کم گرفت، از آن جمله می‌توان به استخراج تفاوت‌های فنی و غیر فنی برنامه‌های کاربردی غیر مبتنی بر وب و مبتنی بر وب پرداخت که سبب خواهد شد، با بررسی رویکردهای مهندسی معکوس برنامه‌های کاربردی غیر مبتنی بر وب و دانستن تفاوت‌های آن‌ها با برنامه‌های مبتنی بر وب، بتوان از زمینه‌ی غنی مهندسی معکوس نرم‌افزارهای غیر مبتنی بر وب در زمینه‌ی وب نیز بهره برد.

۱-۲-۳- نحوه‌ی نگارش

در ادامه‌ی این پایان نامه ابتدا در فصل دوم به جمع‌آوری جامع مطالب مرتبط با مهندسی معکوس نرم‌افزار، خصوصیات برنامه‌های کاربردی مبتنی بر وب، رویکردهای مهندس معکوس این نوع برنامه‌ها پرداخته خواهد شد. نحوه‌ی توزیع بارکاری کارگزاران وب و رویکردها و الگوریتم‌های توزیع بارکاری این کارگزاران در فصل سوم ارائه می‌گردد. سپس بر اساس آن، فرآیندی چند مرحله‌ای برای بازیابی معماری برنامه‌های کاربردی مبتنی بر وب و توزیع بارکاری آن‌ها بر چندین کارگزار، در فصل چهارم پیشنهاد خواهد شد. پیاده‌سازی دقیق رویکرد همراه با ذکر شبه‌کدها و الگوریتم‌ها در فصل پنجم خواهد بود. فصل هفتم بیان جزییات پیاده‌سازی ابزاری به نام RELB (Reverse Engineering and Load Balancing) به زبان جاوا است که دارای قطعات نسبتاً مستقلی از هم است که هر قطعه، پیاده‌سازی مرحله‌ای از رویکرد جدید پیشنهادی است. فصل ششم محتوی ارزیابی برخی آزمایشات بر سیستم RELB است، که بر یک برنامه‌کاربردی مبتنی بر وب پیاده‌سازی شده با زبان JSP اعمال شده است. نتیجه‌گیری و کارهای آتی نیز در فصل هفت آمده‌اند.

روند نگارش پایان‌نامه بدین شرح است که در آغاز، به علت حجم مطالب، دو فهرست مطالب عرضه شده است، فهرست خلاصه و فهرست تفصیلی. در هر فصل ابتدا مقدمه‌ای خواننده را برای ورود به مطالب آماده می‌کند و در انتهای فصل نیز یک نتیجه‌گیری که خلاصه‌ی مطالب و ره‌یافت‌های آن را تشکیل می‌دهد آمده و یک نتیجه‌گیری نهایی نیز برای پایان‌نامه که حاوی یافته‌های به دست آمده در این نوشتار است عرضه شده است. همچنین پیوست‌هایی نیز جهت تکمیل مطالب فصول آخر به پایان‌نامه الحاق شده است. منابع و مراجع مورد استفاده نیز جهت بررسی‌ها و ارجاعات اضافی در خاتمه‌ی نوشتار ذکر گردیده است.

فصل دوم

مهندسی معکوس نرم‌افزار کاربردی مبتنی بر وب

۲-۱- مقدمه

پیش از پرداختن به مقوله‌ی مهندسی معکوس نرم‌افزارهای کاربردی مبتنی بر وب، آگاهی از مهندسی معکوس نرم‌افزار به صورت اجمالی ضروری است. به خصوص که با درک اهداف، مشکلات و سطوح مهندسی معکوس نرم‌افزار که نرم‌افزارهای تحت وب زیرمجموعه‌ای از آن‌ها هستند، زمینه‌ای مناسب برای ورود به حیطه‌ی مهندسی معکوس نرم‌افزارهای کاربردی مبتنی بر وب فراهم می‌گردد. همچنین دسته‌بندی خلاصه انواع رویکردهای ارائه شده برای مهندسی معکوس نرم‌افزار (به مفهوم فراگیر) پیش از پرداختن به جزییات مفید خواهد بود.پس از آن لازم است تا تعریف درست و در حدامکان عمومی، از نرم‌افزارهای کاربردی مبتنی بر وب ارائه داده شود و انواع مختلف آن متمایز گردد و وجوه تفاوت آن با برنامه‌های کاربردی عادی(غیر مبتنی بر وب) مشخص شود. به این ترتیب در ارائه‌ی رویکردهای جدید، می‌توان از زمینه‌ی غنی مهندسی معکوس نرم‌افزارهای کاربردی غیر مبتنی بر وب برای ارائه‌ی ایده‌های جدید در مهندسی معکوس نرم‌افزارهای کاربردی تحت وب نیز بهره گرفت.

با مقدمه چینی مناسبی که در بالا عرضه شد، اکنون می‌توان دسته‌بندی مناسبی از رویکردهای مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب عرضه نمود و تحت آن به بررسی روش‌های مختلف در این زمینه اقدام نمود و آن‌ها را با هم مقایسه نمود و به نقاط ضعف و قوت هر یک پرداخت.

با توجه به آنچه بیان گردید، در بخش‌های مختلف این فصل، مهندسی معکوس نرم‌افزار، نرم‌افزارهای کاربردی مبتنی بروب و روش‌های مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب بررسی خواهد گردید.

۲-۲- مهندسی معکوس نرم‌افزار

در راستای درک مناسب از مهندسی معکوس نرم‌افزار، پرداختن به مطالبی مانند: اهداف مهندسی معکوس، مشکلات آن و سطح‌بندی‌های انجام شده در این قلمرو ضروری است. با دانستن اهداف، ارزیابی رویکردهای مهندسی معکوس نیز امکان‌پذیر می‌شود، زیرا هر رویکردی که اهداف مهم‌تر یا کل اهداف را برآورده سازد، بیشتر مورد تأیید است. با درک دشواری‌های مهندسی معکوس می‌توان در ارائه‌ی رویکردهای جدید، آن‌ها را مد نظر داشت یا در جهت تسهیل‌شان کوشید و در نهایت با دسته‌بندی رویکردهای ارائه شده در این زمینه و راه‌های پیشنهادی، می‌توان به تاریخچه‌ای از مهندسی معکوس نرم‌افزار نیز دست یافت.

۲-۲-۱- اهداف مهندسی معکوس

آشنایی با اهداف مهندسی معکوس از آن جهت مهم است که سبب می‌شود تا در مقایسه‌ی روش‌های مختلف پیشنهاد شده، معیار مشخصی برای ارزیابی داشته باشیم و هر کدام از رویکردها که این اهداف را بهتر برآورده سازند نسبت به دیگران برتر شناخته شوند. به این ترتیب در تهیه یک محک[1] برای مقایسه و برگزیدن الگوهای بهتر، با ابهام کمتری روبرو شویم.

قطعات[2] یک سیستم، محصولات مراحل مختلف چرخه‌ی تولید نرم‌افزارند. برای مثال، تعیین نیازها، معماری و طراحی دقیق و کد مبنا. گرچه مهندسی معکوس می‌تواند از هر یک از این محصولات شروع شود، متداول‌ترین نقطه‌ی شروع همان کد برنامه است [۴]. یک دلیل ساده آن است که سال‌ها پس از تولید، مشخصه‌ها و اطلاعات طراحی سیستم ممکن است دقیق یا حتی موجود نباشد. این نیز بدان خاطر است که عملیات نگه‌داری به خوبی مستند نشده‌اند.

هدف مهندسی معکوس، تسهیل تغییرات با ایجاد درک درباره‌ی ماهیت کار نرم‌افزار، چگونگی نحوه‌ی کار آن و معماری‌اش است [۴]. پیامدهای پی‌گیری این هدف، بازیابی اطلاعات برای تسهیل مهاجرت بین

---

[1] Benchmark
[2] Component

بسترها[1]، بهبود مستندات یا تأمین مستندات جدید، استخراج قطعات قابل استفاده‌ی مجدد، کاهش هزینه‌های نگه‌داری، غلبه بر پیچیدگی، ردیابی اثرات جانبی[2]، کمک به مهاجرت به یک محیط CASE و تولید نرم‌افزارهای مشابه رقیب است. علاوه بر شرح مختصری بر هریک از اهداف فوق خلاصه‌ای از آن‌ها در جدول ۱-۲ آمده است.

- **بازیابی اطلاعات ازدست‌رفته:** با گذر زمان، سیستم دستخوش تغییرات اساسی می‌شود. به خاطر عواملی مانند فشارهای مدیریتی و محدودیت زمانی، مستندات لازم برای تشخیص خواسته‌ها و برای طراحی، ممکن است به‌روز نگه‌داری نشوند یا حتی ممکن است وجود نداشته باشند. این امر کد را تنها منبع اطلاعات درباره‌ی سیستم تبدیل می‌کند. ابزارهای مهندسی معکوس بازیابی این اطلاعات (ویژگی‌های خواسته‌ها و طراحی) را امکان‌پذیر می‌سازند. جزییات بازیابی شده بهتر است به زبان‌هایی مانند Z عرضه شوند و طراحی نیز با نمودارهایی مانند DFD[3] و ERD[4] یا UML[5] نمایش داده می‌شود.
- **تسهیل مهاجرت بین بسترها:** برای بهره‌گیری از بستر نرم‌افزاری جدید، کاربرد دارد (به‌عنوان مثال در یک محیط CASE یا بستر سخت‌افزاری جدید مانند معماری موازی ترکیبی از مهندسی معکوس و مستقیم می‌تواند به‌کار برده شود). ویژگی‌ها و طراحی با استفاده از ابزارهای مهندسی معکوس خلاصه‌سازی[6] می‌شود. سپس مهندسی مستقیم[7] طبق استانداردهای بستر جدید به ویژگی‌ها اعمال می‌شود.
- **بهبود یا تأمین مستندات:** همانطور که پیش از این اشاره گردید، یکی از مشکلات اصلی سیستم‌های قدیمی[8] مستندات ناقص، خارج از رده[9] و یا عدم وجود مستندات است. درخلال مستندسازی مجدد، ابزارها را می‌توان برای بهبود مستندات ناقص یا ایجاد مستندات جدید استفاده نمود.

---

[1] Platforms
[2] Side Effects
[3] Data Flow Diagram
[4] Entity Relationship Diagram
[5] Unified Modeling Language
[6] Abstracted
[7] Forward
[8] Legacy
[9] Out-of-date

- **تأمین نماهای اضافی:** ابزارهای مستندسازی مجدد، می‌تواند برای ایجاد مستندات دیگر علاوه بر مستندات کنونی، مورد استفاده قرار گیرد. این امر سبب ایجاد نماهای دیگری از سیستم می‌شود. برای مثال DFD سیستم را از دید جریان داده نمایش می‌دهد در حالی که نمودار کنترل جریان[1] جریان کنترل بین قطعات مختلف را به تصویر می‌کشد.
- **استخراج قطعات قابل استفاده‌ی مجدد:** با فرض این‌که استفاده از قطعات برای تولید برنامه‌های کنونی، می‌تواند به افزایش سودمندی[2] و بهبود کیفیت محصول منجر شود، مفهوم استفاده‌ی مجدد می‌تواند به‌طور فزاینده‌ای بین مهندسین نرم‌افزار رایج شود. موفقیت در استفاده‌ی مجدد از قطعات نرم‌افزاری به دردسترس بودن آن‌ها وابسته است. روش‌ها و ابزارهای مهندسی معکوس امکان دسترسی و استخراج قطعات را فراهم می‌کنند.
- **غلبه بر پیچیدگی:** یکی از مشکلات اساسی سیستم‌های قدیمی آن است که همراه با تکامل آن‌ها پیچیدگی‌شان هم افزایش می‌یابد. با ایجاد یک تغییر این پیچیدگی را می‌توان اینگونه تصور نمود که اطلاعات سیستم را که با تغییر مرتبط است، خلاصه‌سازی می‌کنیم و آن‌چه را که به تغییر ربط ندارد، در نظر نمی‌گیریم. مهندسی معکوس به همراه ابزارهای CASE نگه‌دارنده را با شکلی از پشتیبانی داده و عملیات تأمین می‌کند.
- **کشف اثرات جانبی:** زمانی که نگهدارنده نمای کلی سیستم را گم می‌کند، حاصل ایجاد تغییر در سیستم اثرات ناخودآگاهی در پی خواهد داشت. بدین ترتیب اثرات جانبی نامطلوب به وجود می‌آیند و ناهنجاری[3] بدون آگاهی حاصل می‌گردد. ابزارهای مهندسی معکوس می‌توانند معماری کلی سیستم را مشهود سازند، و بدین ترتیب پیش‌بینی اثر تغییر و ردگیری مشکلات منطقی و جریان داده را تسهیل بخشند.
- **کاهش هزینه‌های نگهداری:** این مورد یکی از عوامل اصلی افزایش علاقه به مهندس معکوس است. درصد بالایی از کل زمانی که برای ایجاد یک تغییر در سیستم‌نرم‌افزاری لازم است، به درک برنامه اختصاص می‌یابد. دو دلیل عمده برای این امر فقدان مستندات مناسب و دانش ناکافی از قلمرو است. مهندسی معکوس پتانسیل کم‌کردن این مشکلات را داراست و درنتیجه هزینه‌ی نگهداری را کاهش می‌دهد، زیرا که وسیله‌ای برای فراهم نمودن اطلاعات از دست رفته در اختیار قرار می‌دهد.

---

[1] Control Flow Diagram
[2] Productivity
[3] anomaly

جدول ۲-۱ خلاصه‌ی اهداف و مزایای مهندسی معکوس

اهداف	مزایا
۱. بازیابی اطلاعات ازدست‌رفته ۲. تسهیل مهاجرت بین بسترها ۳. بهبود و (یا) تأمین مستندات ٤. تأمین نماهای اضافی ٥. استخراج اجزای قابل‌استفاده‌ی مجدد ٦. غلبه بر پیچیدگی ۷. ردیابی اثرات جانبی ۸. کاهش هزینه‌ی نگهداری	۱. نگهداری أ. بهبود درک سیستم، که در تشخیص خطاها مؤثر است ب. تسهیل شناسایی و استخراج قطعات تأثیرپذیرفته براثر تغییرات تطبیقی یا تکمیلی ت. تأمین مستندات یا نماهای اضافی از سیستم ۲. استفاده‌ی مجدد: شناسایی و استخراج قطعات قابل استفاده‌ی مجدد ۳. بهبود کیفیت سیستم

۲-۲-۲- مشکلات مهندسی معکوس

مهندسی معکوس به عنوان فرآیندی در جهت عکس نمودن محصولی که اکنون مورد استفاده است، برای ایجاد تغییر در آن و یا تسهیل نگه‌داری، در تمام قلمروها با مشکلات مخصوص به خود روبه‌رو است. در دنیای تولید نرم‌افزار نیز برگشت از پیاده‌سازی به سطوح مختلف طراحی یا ایجاد مستنداتی در جهت تسهیل درک نرم‌افزار کنونی برای هر سیستمی به سادگی میسر نیست. مهندسی معکوس نرم‌افزار کار پردردسری است زیرا مشتمل بر نگاشت بین دنیای متفاوت پنج عرصه است [۱]:

۱. **قلمرو برنامه در مقابل زبان برنامه‌نویسی**

یک زبان برنامه‌نویسی تنها محیطی نمادین[1] برای حل یک مسأله‌ی واقعی است. در حالی که ابزارهایی برای درک اینکه از منظر کدنویسی یک کد چه بکند وجود دارد، تعداد کمی ابزار برای کمک به مهندسی معکوس از دیدگاه قلمرو مسأله وجود دارد که یک کد بخصوص چه می‌کند.

۲. **ماشین‌ها و برنامه‌ها در مقابل طراحی چکیده شده و سطح بالا**

[1] Model Environment

مفاهیم ساده تجرید (مثلاً مرتب‌سازی مشتریان بر حسب نام) به سادگی در جزییات برنامه‌نویسی گم می‌شوند. آموزش علوم کامپیوتر بسیار مرتبط با نگاشت تجرید به جزییات برنامه است اما در نگاشت معکوس فعالیت‌های کمی انجام گرفته است.

۳. **انسجام ذاتی، سیستم ساخت‌یافته در مقابل سیستم واقعی، با ساختار تخریب‌یافته**[1]

حتی زمانی که مستندسازی مناسب برای سیستم در دسترس باشد، نگه‌داری پس از زمانی، موجب تحریف ساختار از ویژگی‌های اصلی آن می‌گردد. مهندس معکوس بایستی توانایی تطبیق و همگام‌سازی مجدد طراحی مستندشده و طراحی پیاده‌سازی شده کنونی را دارا باشد.

۴. **برنامه‌های سلسله‌مراتبی در مقابل پیوستگی ادراکی**[2]

برنامه‌های کامپیوتری، عبارات سلسله‌مراتبی رسمی[3] هستند. انسان به قطعه هم‌بسته‌ی داده فکرمی‌کند. یک مهندس معکوس بایستی توانایی "ساخت تکه‌های صحیح سطح‌بالا از جزییات سطح پایین مشهود در برنامه" را داشته باشد.

۵. **تحلیل از پایین به بالای کد در مقابل تحلیل بالا به پایین برنامه**

تحلیل کد در ذاتش پایین به بالاست. برای این امر لازم است مفاهیم سطح بالا، به‌طور همزمان، از قطعات کد استخراج گردند و به پیاده‌سازی سطح پایین نگاشت شوند. آنچه این امر را دشوارتر می‌کند، لزوم توانایی مهندس برای رسیدگی به ابهاماتی مانند درهم‌بودن[4] است که درهم بودن چیدن بین هم وظایف[5] منطقاً مجزا با همان ترتیب اجرا، به صورت عمدی (جهت بهینه‌سازی) یا تصادفی (بر اثر طراحی ناقص یا نگهداری درهم ریخته) است.

امروزه، مهندسی معکوس بسیار به تبادل با انسان و رهنمایی‌های او نیازمند است. در حالی که ابزارهایی برای کمک به مهندس معکوس جهت درک برنامه‌ها وجود دارد، این کار هنوز به یک فرایند کاملاً خودکار[6] تبدیل نشده است. خودکارسازی کامل هیچگاه محتمل نیست زیرا فرایند درک یک سیستم، که مهندسی معکوس در آن نقش دارد، نیازمند به اطلاعات مختص به محدوده‌ی آن سیستم است[۴].

مشکل نامگذاری مشکل دیگری است که در برخی متون به آن اشاره شده است[۴]، حتی اگر خودکارسازی استخراج توصیفات سطح بالا از کد مبنا امکان‌پذیر باشد، نامگذاری هنوز یک مشکل به

---

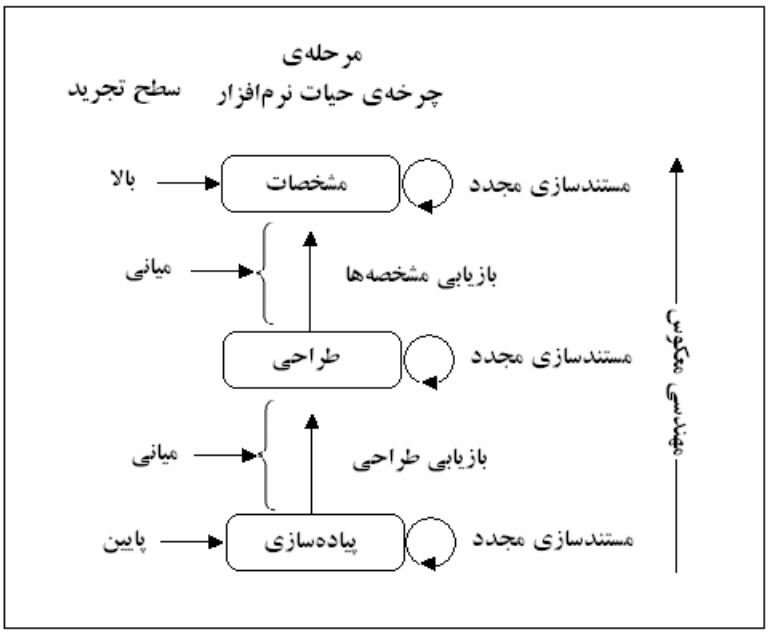
[1] Decaying
[2] Cognitive Association
[3] formal
[4] interleaving
[5] Tasks
[6] Automated

حساب می‌آید. استخراج ویژگی‌ها یک نکته است و نامگذاری خودکار (آن‌هم به صورت با معنی) چیز دیگر. برای مثال کدمبنای الگوریتم مرتب‌سازی دودویی را در نظر بگیرید، که به آن نام BinarySort اطلاق گردد، اما متغییرهای آن اسامی مشابه p3، p4 و ... داشته باشند.

۲-۲-۳- سطوح مهندسی معکوس

۲-۲-۳-۱- سطح‌بندی پایه‌ای مهندسی معکوس

مهندسی معکوس مشتمل بر به‌کارگیری یک یا بیشتر سطوح تجرید با روش از بالا به پایین یا پایین به بالاست. این امر شامل یافتن ساختارهای سطح پایین پیاده‌سازی شده و سپس جایگزینی آن‌ها با همتاهای سطح بالاست. فرآیند مذکور احتمالاً به یک فرموله‌سازی معماری کلی برنامه نیز منجر خواهد شد. البته شایان توجه است که محصول مهندسی معکوس لزوماً نباید به سطح تجرید بالاتری منجر گردد. اگر سطح آن با سطح سیستم مبدأ یکی باشد، معمولاً از آن با نام *مستندسازی مجدد* نام برده می‌شود و از سوی دیگر اگر محصول به دست آمده در سطحی بالاتر از تجرید قرار داشته باشد، تحت نام *بازیابی طراحی* یا *بازیابی مشخصات* از آن نام برده خواهد شد (شکل ۲-۱).

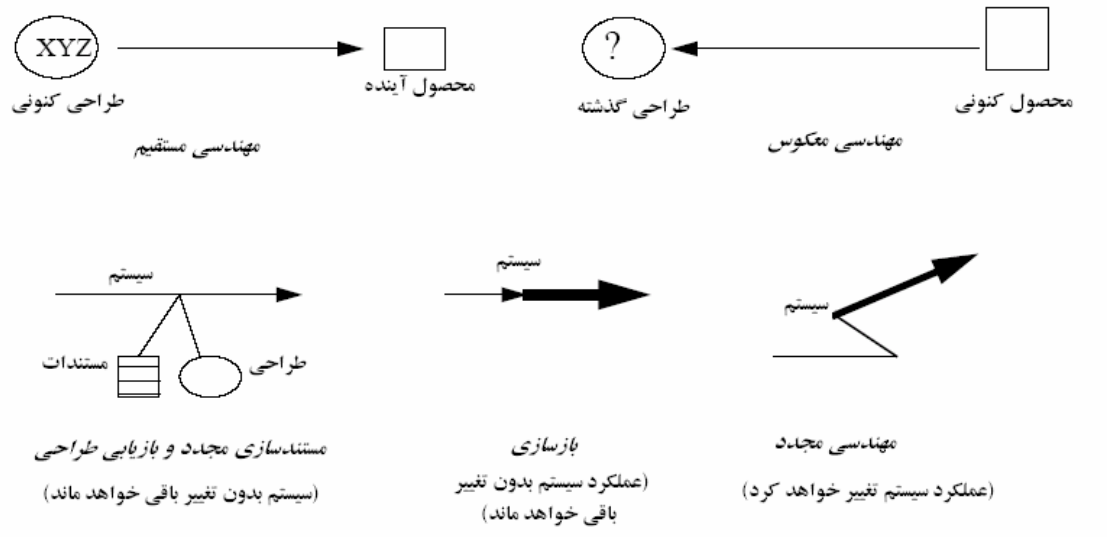


شکل ۲-۱ سطوح تجرید در یک سیستم نرم‌افزاری

بر همین اساس در برخی منابع نیز چهار سطح مهندسی معکوس به شرح زیر آمده‌اند[۱]:

- **مستندسازی مجدد:** ضعیف‌ترین نوع از مهندسی معکوس است که تنها مشتمل بر ایجاد (در صورت عدم وجود) یا مرور مستندات سیستم با همان سطح تجرید است.
- **بازیابی طراحی:** مستندسازی مجدد، اما با استفاده از دانش قلمرو[1] کنونی و نیز اطلاعات خارجی که امکان ایجاد یک مدل از سیستم با سطح تجریدی بالاتر میسر گردد.
- **بازسازی[2]:** تغییرات جانبی سیستم با همان سطح تجرید که البته در همان سطح از عملکرد و مفهوم نیز باقی خواهد ماند.
- **مهندسی مجدد:** افراطی‌ترین و دوردست‌ترین انتخاب در این زمینه. عموماً مشتمل بر ترکیبی از مهندسی معکوس برای درک برنامه و یک به‌کارگیری مجدد مهندسی مستقیم[3] برای اینکه بفهمیم چه عملکردهایی نیازمند به حفظ، حذف یا اضافه شدن دارد.

شکل زیر بیانگر ارتباط بین مهندسی مستقیم و درجات مختلف مهندسی معکوس است. لازم به ذکر است که اصطلاح مهندسی مستقیم برای تشخیص فرایند مهندسی نرم‌افزار تجاری از مهندسی معکوس به کار می‌رود و منظور همان رویکردهای متعارف تولید نرم‌افزار است.



شکل ۲-۲ مهندسی مستقیم، مهندسی معکوس و مشتقات

[1] Domain
[2] Restructuring
[3] Forward Engineering

۲-۲-۳-۲- سطح‌بندی مهندسی معکوس بر اساس هدف

دسته‌بندی دیگری که علاوه بر روش پیشین، برای مهندسی معکوس نرم‌افزار مطرح گردیده است، دسته‌بندی رویکردهای مهندسی معکوس بر اساس هدف است. مهندسی معکوس لزوماً با سیستم‌ها و فن‌آوری‌های قدیمی سروکار ندارد. سیستم مورد نظر ممکن است می‌تواند محصول جدید یا کهنه‌ای باشد. در حالت اول هدف به‌طور خلاصه، تولید نماهای دیگر جهت مستند نمودن نرم‌افزار است. این امر می‌تواند مفید باشد، حتی زمانی که نرم‌افزار ارائه‌ی خوبی دارد و در ظاهر مشکلی ندارد. جهت سادگی این مورد را مهندسی معکوس برای مستندسازی می‌نامیم. اما زمانی که سیستم قدیمی است، اهداف مهندسی معکوس می‌تواند اولاً تسهیل نگه‌داری نرم‌افزار و ثانیاً استفاده‌ی مجدد از قطعات آن برای تولید محصولی جدید باشد. که می‌توان آن را مهندسی معکوس جهت نگهداری یا مهندسی معکوس برای استفاده‌ی مجدد نامید. در حالت عمومی زمانی که هدف مهندسی معکوس قرار دادن نرم‌افزار یا اطلاعات مرتبط با آن تحت کنترل یک ابزار است، از عنوان مهندسی معکوس برای مجتمع‌سازی استفاده می‌شود. این امر سبب می‌شود که ابزارهای جدید، نرم‌افزار قدیمی را برای انتقال فن‌آوری با نرم‌افزار جدید مجتمع سازند [۶].

۲-۲-۴- رویکردهای معمول مهندسی معکوس

هدف افراد از مهندسی معکوس سیستم نرم‌افزاری ایجاد مدل‌های ذهنی تصحیح‌شده از سیستم در جهت کمک به تصمیم‌گیری‌های تغییر و تصحیح نرم‌افزار است[۱۲]. گرچه این امر برای نرم‌افزارهای مقیاس کوچک به سادگی و معمولاً از طریق مرور و بررسی کد امکان‌پذیر است، اما در سیستم‌های بزرگ که دارای صدها هزار تا میلیون‌ها خط کد بودن مستندات هستند، به همراه افزایش اندازه، پیچیدگی نیز افزایش می‌یابد. برای ایجاد مدل‌های کامل‌تر از سیستم، مهندس معکوس بایستی اطلاعات مرتبط با سیستم که او را در این فرایند کمک می‌کنند، جمع‌آوری نماید.

رویکردهای زیادی برای مهندسی معکوس سیستم‌های نرم‌افزار وجود دارد، که برخی از آن‌ها عبارتند از[۱۲]:

۱. مطالعه‌ی مستندات و کدمبنای کنونی

افراد مختلف بررسی کد مبنا، مطالعه‌ی کد و تمرینات مرورآن را بررسی کرده‌اند که برخی از آن‌ها در منبع ۱۲ آمده‌اند. استفاده از این رویکرد در زمان عدم دسترسی به مستندات یا وجود مستندات ناقص دشوار است. مطالعه‌ی کد مبنا رایج‌ترین نحوه است، اما قابل توسیع نیست زیرا بررسی میلیون‌ها خط کد ممکن

است هفته‌ها و ماه‌ها به طول بینجامد و لزوماً به افزایش درک سیستم نیز منجر نشود. علاوه بر آن در آغاز فعالیت مهندسی معکوس، فرد به دنبال اطلاعات ریز و جزیی نیست، بلکه بیشتر مایل به داشتن یک نمای کلی از سیستم است.

۲. اجرای نرم‌افزار و تولید و تحلیل ردیابی‌های اجرا

استفاده از اطلاعات پویا، برای مثال اطلاعات جمع‌آوری شده در خلال اجرای یک قطعه از نرم‌افزار، نیز در رابطه با مهندسی معکوس مورد استفاده قرار گرفته است. مواردی در این زمینه قابل استناداند که در منبع ۱۲ آمده‌اند. اما این مورد نیزدر زمینه‌ی مقیاس‌پذیری (ردیابی در چند ثانیه ممکن است بسیار حجیم شود) و تفسیر (هزاران فراخوانی پیام ممکن است سبب پنهان شدن اطلاعات خاص مورد جستجو گردد) دارای مشکلاتی است.

۳. مصاحبه با کاربران و تولیدکنندگان نرم‌افزار

این امر ممکن است سبب درک مؤثری از سیستم نرم‌افزاری گردد، اما به خاطر تفاوت ذاتی دیدگاه‌های ذهنی افراد مصاحبه شونده و دشواری فرموله نمودن و استفاده‌ی مجدد از این دیدگاه‌ها، به سادگی امکان‌پذیر نیست. همچنین یافتن کسانی که در برهه‌ی طولانی تولید و نگهداری نرم‌افزار عضو تیم تولید بوده‌اند و به این ترتیب دارای دانش کامل از چرخه‌ی حیات نرم‌افزار بوده‌اند نیز به سختی میسر است.

۴. استفاده از ابزارها و فنون متنوع

استفاده از ابزارها (مصورسازها، برش‌دهنده‌ها[1]، موتورهای پرس‌وجو و ...) و فنون (مصورسازی، خوشه‌بندی، تحلیل مفهومی و ...) متنوع برای تولید نماهای سطح بالا از کد مبنا در این مورد رایج است. ابزارهای ایجاد شده توسط مراکز تحقیقاتی و به روش‌های متنوع تهیه شده‌اند.

۵. تحلیل تاریخچه‌ی ویرایش‌ها

در این مورد کار کمتری انجام گرفته است. این زمینه که در آن تحول یک قسمت از نرم‌افزار توسط فنونی مانند نمودارسازی مجدد[2]، مصورسازی، تحلیل مفهومی و داده‌کاوی مشخص می‌گردد، هنوز یک

[1] Slicers
[2] Graph rewriting

زمینه‌ی جوان به حساب می‌آید. نتایج برای درک گذشته‌ی یک بخش نرم‌افزاری و گاهی پیش‌بینی تغییرات آن در آینده مفیداند.

۶. ارزیابی یک سیستم نرم‌افزاری و کیفیت آن با کمک معیارهای نرم‌افزاری

ابزارهای سنجش نرم‌افزار برای ارزیابی کیفیت و کمیت کد مبنا با محاسبه‌ی معیارهای مختلف مطلوب مورد استفاده قرار می‌گیرند. برای مثال کلاس‌های منسجم، زیرسیستم‌ها و .... را می‌توان مورد ارزیابی قرار داد.

۲-۳- نرم‌افزار کاربردی تحت وب

با توسعه‌ی اینترنت، نوع جدیدی از برنامه‌کاربردی عرضه شد: برنامه‌کاربردی تحت وب، که از بستر اینترنت استفاده می‌کند. با توجه به فراگیر شدن استفاده از برنامه‌های کاربردی تحت وب و با توجه به تفاوت‌های موجود بین برنامه‌های کاربردی تحت وب و غیر مبتنی بر وب، شناخت برنامه‌های کاربردی تحت وب و درک تفاوت بین آن‌ها و برنامه‌های کاربردی غیر مبتنی بر وب در جهت ارائه‌ی رویکردهای بهتر برای مهندسی معکوس با احتساب تفاوت‌های بین این دو محرز است. همچنین اگر فرق بین این دو نوع برنامه‌کاربردی‌ها مشخص‌تر شود، امکان تغییر برخی از آن‌ها از غیر مبتنی بر وب، به تحت وب با اشراف به میزان و نحوه‌ی تفاوت نیز وجود خواهد داشت. در ادامه ابتدا تعریفی از برنامه‌های کاربردی تحت‌وب ارائه می‌دهیم و سپس انواع مختلف آن را دسته‌بندی می‌کنیم سپس قطعات مختلفی که یک برنامه‌ی تحت وب می‌تواند دارا باشد را معرفی خواهیم کرد و در نهایت به تفاوت برنامه‌های کاربردی عادی و تحت‌وب خواهیم پرداخت.

۲-۳-۱- تعریف برنامه‌کاربردی تحت وب

هدف اولیه‌ی WWW نمایش اطلاعات بود. امروزه برنامه‌های پیشرفته‌ی تحت وب تا حد برنامه‌های توزیع شده‌ی پیچیده نیز رشد یافته‌اند. در نتیجه‌ی اینترنت و وب، بسیاری از برنامه‌های کاربردی جدید تجاری به جای تولید از طریق فن‌آوری‌های مشتری-کارگزار (غیر مبتنی بر وب) با استفاده از فن‌آوری‌های وب مانند مرورگرها و کارگزارهای وب، ایجاد می‌شوند.

برنامه‌های کاربردی تحت وب، برنامه‌هایی هستند که از بستر اینترنت برای تحویل نتایج عملکردشان استفاده می‌کنند[۷]. گزارشات سال ۱۹۹۹ نشان می‌دهد[۲] که برنامه‌های کاربردی تحت وب بیش از ۳۰ درصد از برنامه‌های نرم‌افزاری را در کل صنعت نرم‌افزار از آن خود کرده‌اند، هر چند نگارنده موفق به اطلاع از وضعیت کنونی نرم‌افزارهای کاربردی تحت وب نشده است، اما بدیهی به نظر می‌رسد که رشد روزافزون اینترنت و تعدد وب‌سایت‌های اینترنتی این آمار را بیشتر نموده است. دو علت عمده برای ترجیح برنامه‌های کاربردی تحت وب به برنامه‌های کاربردی غیر مبتنی بر وب وجود دارد:

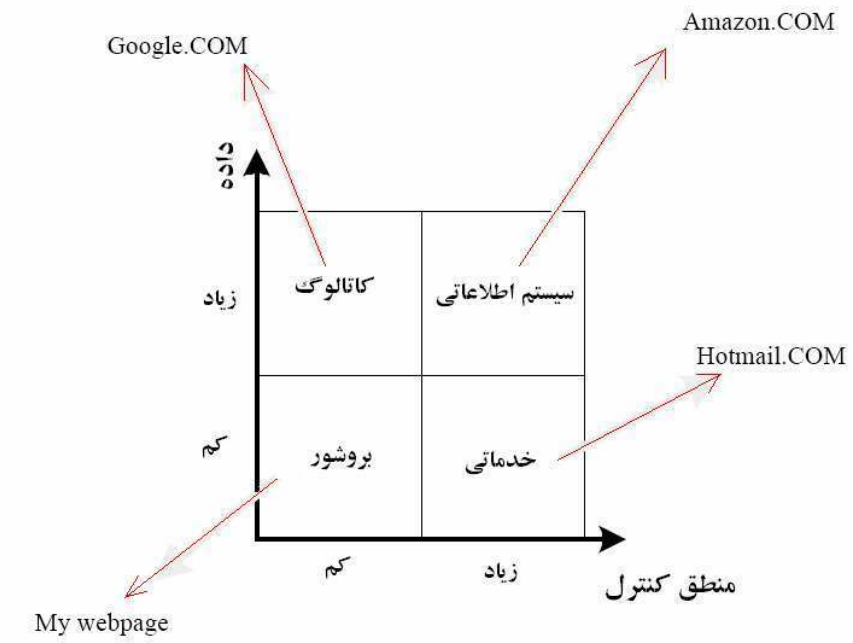
۱. *برنامه‌های کاربردی تحت وب دسترس‌پذیرترند.* پروتکل HTTP که در برنامه‌های کاربردی تحت وب مورد استفاده قرار می‌گیرد، پروتکل استانداردی است که می‌تواند حتی پشت

دیواره‌های آتش نیز مورد استفاده قرار گیرد. همچنین برنامه‌های کاربردی تحت وب بر بسیاری از بسترها[1] در دسترسند.

۲. *هزینه‌های نگهداری و تولید برنامه‌های کاربردی تحت وب کمتر است.* بدان علت که برنامه‌های کابردی تحت وب در مرورگر اجرا می‌شوند، به نصب برنامه‌ی مشتری به ازای هر کامپیوتر استفاده کننده، نیازمند نیستند. نگهداری برنامه‌های کاربردی تحت وب با تغییر کد واقع بر کارگزار قابل امکان‌پذیر است. این امر سبب کاهش زمان و هزینه‌ی ارتقا و توسعه‌ی برنامه‌ی کاربردی تحت وب نسبت به برنامه‌های کاربردی تجاری مشتری-کارگزار می‌شود.

۲-۳-۲- دسته‌بندی برنامه‌های کاربردی تحت وب

برنامه‌های کاربردی تحت وب، به نوع خاصی از برنامه‌ها محدود نمی‌شوند. محدوده‌ی آن‌ها می‌تواند از صفحات‌وب ایستای ساده تا برنامه‌های پیچیده باشد. انواع مختلف برنامه‌های تحت وب بر مبنای پیچیدگی داده‌ای و کنترلی‌شان همانند شکل زیر گروه‌بندی می‌شوند[۷] :



شکل ۲-۳ دسته‌بندی برنامه‌های کاربردی تحت وب

[1] Platform

۱. *برنامه‌های کاربردی تحت وب از نوع بروشور*[1]: این نوع برنامه‌ها، اولین برنامه‌های کاربردی تحت وب هستند از صفحات ایستا تشکیل شده‌اند و معمولاً منطق برنامه نویسی زیادی را دربرندارند. مثال‌هایی در این زمینه عبارتند از صفحات وب شخصی، که مشتمل بر رزومه و اطلاعات شخصی است یا صفحاتی درباره‌ی محصولات یک شرکت است.

۲. *برنامه‌های کاربردی تحت وب مبتنی بر خدمات (خدماتی)*: این وب‌سایت‌ها خدماتی را به کاربران عرضه می‌کنند. برنامه‌های کاربردی مبتنی بر خدمات، مشتمل بر منطق برنامه‌نویسی لازم جهت پیاده‌سازی خدمت‌اند. طرح‌بندی[2] داده معمولاً در اولویت دوم است. در خلال نگهداری، تولیدکنندگان لازم است درک مناسبی از منطق کنترل داشته باشند. از مثال‌هایی در این زمینه خدمات نامه‌ی الکترونیکی تحت وب یا سیستم‌های پردازش کلمات برخط را می‌توان نام برد.

۳. *برنامه‌های با حجم داده‌ای زیاد*[3]: این برنامه‌ها سایت‌هایی هستند که رابطی برای مرورگری و پرس‌وجو در مقادیر زیاد داده فراهم می‌کنند. تأکید اصلی در این برنامه‌ها بر داده است و منطق برنامه در کمترین حالت ممکن قرار دارد. در خلال نگهداری تولیدکنندگان لازم است که درک خوبی از جریان داده داشته باشند. مثالی در این زمینه کاتالوگ کتابخانه‌های برخط است.

۴. *برنامه‌های سیستم‌های اطلاعاتی*: این برنامه‌های برنامه‌های مبتنی بر خدمات و برنامه‌های با حجم داده‌ای زیاد را با هم ترکیب می‌کنند. تولیدکنندگان برنامه‌های کاربردی سیستم‌های اطلاعاتی با جریان داده (برای مرورگری و بازیابی داده) . جریان کنترل (جهت مراحل مختلف عملیات بر داده) مواجه هستند. کتابفروشی‌های الکترونیکی یا عملیات بانکی برخط مثال‌هایی در این زمینه هستند.

همانطور که در برنامه‌های تجاری محرز است، تولیدکنندگان نیازمند به درکی مناسب از جریان داده و کنترل در برنامه‌هایشان هستند. علاوه بر آن برنامه‌های کاربردی تحت وب دارای وابستگی روابط و جذابیت بیشتری مانند مرورگری پیوندها بین صفحات مختلف از برنامه کاربردی تحت وب هستند.

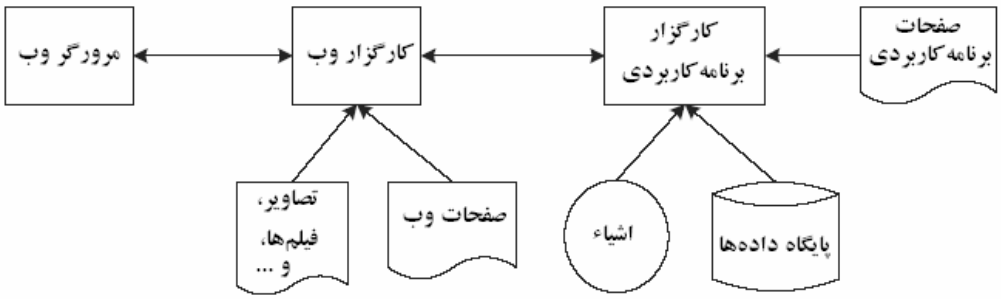
۲-۳-۳- قطعات یک برنامه کاربردی تحت وب

در یک برنامه کاربردی تحت وب، بسیاری قطعات از داخل با هم مرتبط شده‌اند تا عملکرد آن را شکل دهند. قطعات زیر در یک برنامه کاربردی تحت وب به چشم می‌خورند[۵]:

---

[1] Brochure
[2] Layout
[3] Data intensive

۱. مرورگر وب (که توسط مشتری[۱] مورد استفاده قرار می‌گیرد)
۲. کارگزارن وب[۲]
۳. صفحات وب
۴. کارگزاران برنامه‌های کاربردی[۳]
۵. صفحات برنامه‌های کاربردی[٤]
۶. پایگاه داده‌ها
۷. اشیای توزیع شده[٥] مانند CORBA، EJB و COM
۸. اشیای وبی چند رسانه‌ای[٦] مانند تصاویر، فیلم[۷]
۹. ...



شکل ۲-۴ جریان داده بین قطعات یک برنامه کابردی تحت وب

شکل ٤-۲ جریان داده بین قطعات مختلف یک برنامه کاربردی تحت وب را نشان می‌دهد. کاربرِ برنامه‌کاربردی تحت وب، یک مرورگر وب به عنوان واسط دسترسی به برنامه‌کاربردی تحت وب، جهت استفاده از آن را به کار می‌گیرد. کاربر توسط کلیک کردن بر پیوند[۸] و پرکردن فیلدهای فرم، با مرورگر کار می‌کند. مرورگر نیز در مقابل عمل‌های[۹] کاربر را به کارگزار وب انتقال می‌دهد. درخواست‌ها توسط

[1] Client
[2] Web Servers
[3] Application Servers
[4] صفحات برنامه‌های کاربردی، صفحات وبی هستند که مشتمل بر کدهای اجرایی می‌باشند که پیش از انتقال به مرورگر درخواست کننده، در داخل کارگزار برنامه‌کاربردی اجرا می‌شوند.
[5] Distributed Objects: این اشیاء از جنس کد مبنای برنامه نویسی شئ‌گرا نیستند، در عوض بخش‌هایی از کد کامپایل شده می‌باشند که خدمتی را برای بقیه سیستم نرم‌افزاری از طریق یک واسط معین، فراهم می‌کنند.
[6] Multi-media web objects
[7] Video
[8] Link
[9] Action

پروتکل HTTP فرستاده می‌شود. به محض دریافت درخواست، کارگزار وب تعیین می‌کند که آیا خودش می‌تواند یا اینکه کارگزار نرم‌افزار کاربردی به کار گرفته شود. کارگزار نرم‌افزار کاربردی و کارگزار وب می‌توانند بر یک ماشین واقع شده باشند، یا اینکه بر ماشین‌های مختلفی باشند. بسیاری کارگزارهای وب و کارگزارهای نرم‌افزار کاربردی می‌توانند درخواست را توسط یک برنامه‌کاربردی پاسخ دهند. کارگزار وب می‌تواند صفحات HTML و محتوی چندرسانه‌ای مانند عکس، فیلم یا صوت را پس بفرستد یا اینکه درخواست را به کارگزار نرم‌افزارکاربردی بفرستد. کارگزار نرم‌افزار کاربردی صفحه برنامه‌کاربردی را پردازش می‌کند و یک صفحه HTML به کارگزار وب برمی‌گرداند. در نهایت کارگزار وب صفحه درخواست شده را به مرورگر وب برمی‌گرداند، که آن هم، صفحه را به کاربر نمایش می‌دهد.

۲-۳-۴- وجوه تمایز از برنامه کاربردی غیر مبتنی بر وب

با توجه به بررسی‌هایی که در منابع مختلف انجام گرفته است، اشاره‌ی صریحی به این موضوع صورت نگرفته است، اما تفاوت‌های بین برنامه‌های کاربردی تحت وب و غیر آن را می‌توان به دو مقوله‌ی اصلی برگرداند، اول تفاوت‌های فنی برنامه‌های کاربردی تحت وب و برنامه‌های کاربردی غیر مبتنی بر وب و دوم تفاوت‌های غیر فنی. نکات متعدد دیگری نیز درباره‌ی تفاوت‌های این دو وجود دارد که به علت محرز بودن و عدم تأثیر مفید، در اینجا به آن‌ها پرداخته نمی‌شود، از جمله‌ی آن‌ها می‌توان به مواردی از قبیل بستر اجرای آن‌ها، امنیت، تعداد مشتری‌ها در یک زمان معین، متغییر یا ثابت بودن پهنای باند در دسترس برای اجرای آن‌ها، زمان پاسخ و ... اشاره نمود.

۲-۳-۴-۱- تفاوت‌های فنی

برنامه‌های کاربردی مبتنی بر وب بیشتر با جداول داده، اشیای توزیع شده و اشیای چندرسانه‌ای در ارتباطند در حالی که برنامه‌های کاربردی عادی بیشتر با متغییرها، روال‌ها و اشیای ساده در ارتباطند. در برنامه‌های کاربردی تحت وب استفاده‌های حجیمی از زبان‌های اسکریپت برای ترکیب قطعات مختلف با هم به کار می‌رود، همچنین برنامه‌های کاربردی مبتنی بر وب از زبان‌های متعددی برای پیاده‌سازی یک برنامه ممکن است استفاده کنند، در حالی که این موارد در برنامه‌های کاربردی عادی کمتر به چشم می‌خورد [۱۹]. همچنین بسیاری از فن‌آوری‌های وب به جای روش‌های تولید مناسب، در جهت القای روش‌های بد هستند

و اصول تولید برنامه کاربردی‌های قلمرو مهندسی نرم‌افزارمبتنی بر مؤلفه‌ای بودن، کپسوله‌کردن داده، جداسازی نگرانی‌ها[1] یا پنهان‌سازی اطلاعات، در حقیقت نه توسط HTML و نه سایر زبان‌ها اسکریپتی (از قبیل JavaScript، JSP و VBS) که به وفور در تولید برنامه‌های تحت وب کنونی مورد استفاده هستند، رعایت نشده است[۲۱].

۲-۳-۴-۲- تفاوت‌های غیرفنی

در حالیکه برنامه‌های کاربردی مبتنی بر وب، امکان‌های اضافه نسبت به برنامه‌های کاربردی عادی برای تبلیغ محصولات و تأمین اطلاعات و خدمات تجارت الکترونیکی فراهم می‌کنند[۲۱]، زمان ایجاد آن‌ها معمولاً کوتاه‌تر از برنامه‌های کاربردی عادی است[۱۹] و برنامه‌ریزی فشرده برای تولید برنامه‌های کاربردی تحت وب اغلب به تولیدکنندگان آن‌ها جهت تولید فاقد یک فرایند مناسب فشار وارد می‌کند[۲۰]. همچنین بیشتر اوقات اصول مهندسی نرم‌افزار در تولید نرم‌افزارهای کاربردی تحت وب لحاظ نمی‌شوند. همانطور که پرسمن می‌گوید، عدم تمایل تولید کنندگان وب برای تطابق دادن اصول ثابت شده‌ی نرم‌افزار، جای تأسف دارد [۱۸]. از تفاوت‌های دیگر برنامه‌های کاربردی غیرمبتنی بر وب آن است که مقدار زیادی اطلاعات متنی در صفحات دربرگیرنده‌ی آن‌ها وجود دارد و توسط متمایز کردن کلمات موجود در آن‌ها از قبیل زیرخط‌دار کردن یا کشیده کردن به مرتبط‌ترین اطلاعات مرتبط ارجاع داده می‌شود[۲۰].

[1] Concerns

۲-۴- روش‌های مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب

تاکنون روش‌های متنوعی برای مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب عرضه گردیده است. این روش‌ها به‌قدری فراوان‌اند که بررسی دقیق تمام آن‌ها به زمان زیادی نیاز دارد. جمع‌آوری و بررسی این روش‌ها در دسته‌های منطقی که کار مقایسه‌ی آن‌ها را نیز ساده سازد، می‌تواند منجر به پی بردن به نقاط ضعف و قوت آن‌ها گردد و ارائه‌ی یک روش جدید برای این مورد را نیز تسهیل بخشد. دسته‌بندی رویکردهای مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب در مقالات معمولاً به دو گروه عمده اشاره دارد[۱۶]: گروه اول در جهت دستیابی به یک نمای معماری از برنامه که قطعات (مانند صفحات یا قطعات صفحات داخلی) و روابط آن را در سطوح جزییات مختلف نمایش دهد، تلاش دارد. اما دیگری از فنون خوشه‌بندی برای خلاصه‌سازی نیازهای عملیاتی که توسط برنامه‌ی کاربردی مبتنی بر وب برآورده می‌شود و با نمودارهای UML مدل می‌شود، استفاده می‌کند. البته بی‌شک روش‌های دیگری نیز وجود دارند که به کلی در یکی از این دسته‌بندی‌ها جای نمی‌گیرند، اما بررسی آن‌ها با توجه به خاص‌منظوره بودن اغلب آن‌ها و کوتاه بودن مجال مناسب دیده نشد.

آن‌چه در ادامه مورد بررسی قرار گرفته است، چکیده‌سازی روش‌هایی است که به نظر جهت‌ده‌تر و مفیدتر می‌رسیدند. در این میان سعی شده است پیش‌فرض‌ها و شرایط اعمال هر روش نیز بیان گردد تا درک درست‌تری از آن در ذهن القا شود.

۲-۴-۱- روش‌های بازیابی معماری

این روش‌ها اغلب با تولید نمودارهایی از روند کنونی عملکرد نرم‌افزار، با استفاده از کد مبنا، مستندات کنونی و در صورت امکان استفاده از عامل انسانی، سبب ایجاد شناخت و تسهیل نگهداری آن می‌شوند. سطوح مختلف تجرید در روش‌های به کار گرفته شده وجود دارد و سعی نگارنده در این بوده است تا از هر سطح یک مقاله‌ی جهت‌ده‌تر انتخاب نماید، به نحوی که پوشش خوبی از سطوح مختلف درشتی[1] در این زمینه انجام پذیرد. به عنوان مثال رویکردهایی برای بازیابی معماری، بازیابی طراحی و فنون به‌کار گرفته شده در سطح کد در ادامه آمده است.

[1] Granularity

۲-۴-۱-۱- روش اول

این روش [۵] به ارائه یک چهارچوب[1] برای بازیابی معماری برنامه‌های کاربردی تحت وب[2] می‌پردازد. تولیدکنندگان[3] می‌توانند معماری بازیابی شده را مصورسازی[4] نموده و آن را مرور کنند[5] و در جهت دستیابی به درکی بهتر از برنامه‌کاربری تحت‌وب‌شان تحلیل کنند.

روش کنونی در واقع ادامه‌ی روند تکمیل نرم‌افزاری با نام محیط کتاب‌فروشی سیار[6] (PBS) است که قسمت اعظم دانش و فن توسعه یافته در طول ده سال پیش در زمینه فهم برنامه را ترکیب کرده است. برنامه‌ی مذکور جهت بازیابی طراحی برنامه‌های بزرگی مانند لینوکس[7](باهشت‌صد هزار خط کد)، آپاچی[8](با هشتاد هزار خط کد) و موزیلا[9](با دومیلیون و یک‌صدهزار خط کد) به کار رفته است. مقاله‌ی کنونی استفاده مجدد و توسعه قابلیت‌های PBS برای پشتیبانی از بازیابی طراحی برنامه‌های کاربردی تحت وب را شرح می‌دهد. در آن به تولید مجموعه‌ای از ابزارها با قابلیت تجزیه[10] و بسط روابط بین قطعات[11] برنامه‌های کاربردی تحت وب پرداخته شده است. همچنین مصورساز PBS را برای کار با طبیعت ناهمگون[12]برنامه‌های کاربردی تحت وب تغییر یافته است.

۲-٤-۱-۱-۱- قطعات[13] برنامه کاربردی تحت وب

در این مرجع قطعات برنامه‌کاربردی تحت وب که عملکرد آن را شکل می‌دهند، همان موارد ذکر شده در بخش ۲-۳-۳ هستند.

---

[1] Framework
[2] Web Application
[3] Developers
[4] Visualize
[5] Navigate
[6] Portable Bookshelf Environment برای اطلاعات بیشتر در زمینه‌ی این برنامه می‌توان به آدرس http://www turing toronto.edu/pbs مراجعه نمود.
[7] Linux
[8] Apache
[9] Mozilla
[10] Parsing
[11] Components
[12] Heterogonous
[13] Components

۲-٤-۱-۱-۲- نرم‌افزار مصورسازی

نرم‌افزارهای کاربردی تحت وب از تعداد زیادی قطعات تشکیل شده‌اند و هر قطعه ممکن است معماری یا طراحی داخلی خودش را داشته باشد. یک تولیدکننده‌ی نرم‌افزار کاربردی تحت وب در عمل با چیدمان[1] قطعات سروکار دارد. در این شیوه‌ی ارائه‌ی معماری نرم‌افزار کاربردی تحت وب، معماری داخلی کارگزار وب و مرورگر وب، به خاطر جلوگیری از ایجاد پیچیدگی بیشتر مصورسازی[2] سیستم و به خاطر عدم سهیم بودن در فهم کلی سیستم نرم‌افزاری، نشان داده نمی‌شوند و در عوض به بازیابی قطعات برنامه‌کاربردی تحت وب که در پیش تشریح شد، پرداخته می‌شود. نمودارهای معماری برای برنامه‌های کاربردی تحت وب بایستی توانایی نمایش مواردی از قبیل اشیای توزیع شده، جدول‌های پایگاه‌داده‌ها، اشیای چند رسانه‌ای (مانند فایل‌های فیلم، عکس و صوت) که در جهت پیاده‌سازی برنامه‌های کاربردی تحت وب، کنار هم گرد می‌آیند را داشته باشد.

۲-٤-۱-۱-۳- بازیابی معماری

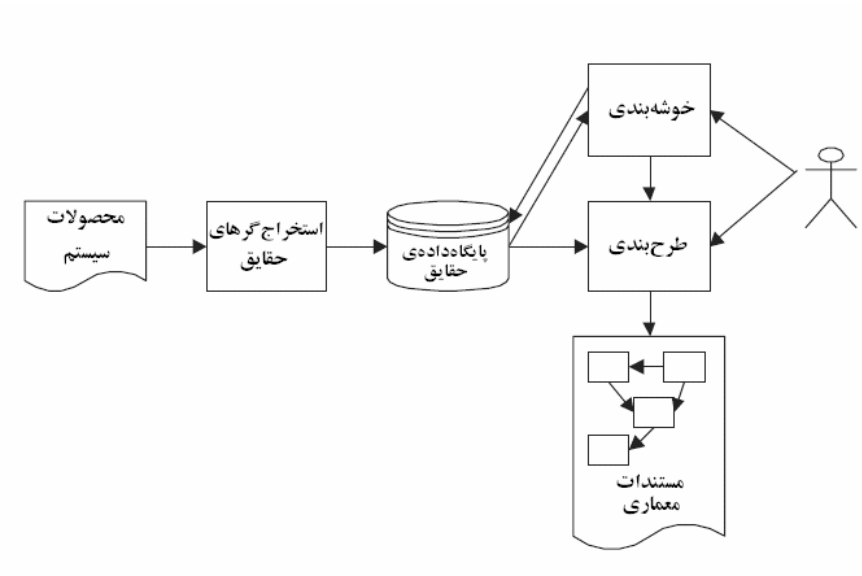
در این رویکرد از یک فرایند نیمه‌خودکار[3] برای بازیابی معماری برنامه‌های کاربردی مبتنی بر وب استفاده شده است. این فرایند از برخی ابزارها برای آزمون کد مبدأ برنامه‌های کاربردی تحت وب بهره می‌گیرد. خروجی ابزار با ورودی‌هایی از یک خبره‌ی سیستم ترکیب می‌شود تا منجر به تولید نمودارهای معماری گردد. فرآیند ایجاد مستند معماری برنامه‌های کاربردی تحت وب توسط محیط کتاب‌فروشی سیار (BPS) در زیر نمایش داده شده است.

در تصویر زیر ابتدا محصولات[4] سیستم‌نرم‌افزاری (مانند کد مبدأ، مستندسازی، ردگیری‌های اجرا[5]، صفحات وب و ...) با کمک استخراج‌گرهای حرفه‌ای پردازش می‌شوند. استخراج‌گرها به‌طور خودکار حقایقی[6] را درباره سیستم‌نرم‌افزاری بر اساس این محصولات، تولید می‌کنند. این حقایق می‌توانند به‌طور جزیی مورد بررسی قرار گیرند، مانند: *تابع "f" از متغییر "a" استفاده می‌کند*، یا در سطوح بالاتری از تجرید عرضه شوند، مانند: *فایل "f1" ازفایل "f2" استفاده می‌کند*. سطح تجرید حقایق استخراجی به استخراج‌گر و سطح تحلیلی که بر حقایق بازیابی شده اعمال خواهد شد، بستگی دارد. به عنوان مثال برای تحلیل در سطح

---

[1] Topology
[2] Visualizing
[3] Semi-Automated
[4] Artifacts
[5] Execution Traces
[6] Facts

معماری، حقایق در سطح توابع مورد نیاز نیستند و می‌توانند در سطحی بالاتر مورد بررسی قرار گیرند. حقایق استخراجی در قالب TA[1] ذخیره می‌شوند. شکل ۲-۵ جزییات بیشتری درباره چگونگی انجام این استخراج از برنامه‌های کاربردی تحت وب عرضه می‌کند.



شکل ۲-۵ استفاده از PBS برای ایجاد مستندات معماری

برای کاهش پیچیدگی نمودارهای تولید شده، سیستم‌نرم‌افزاری با استفاده از فنون خوشه‌بندی[2] به زیر سیستم‌های بامعنا تجزیه می‌شود. خوشه‌بندی در ابتدا به‌طور خودکار و توسط ابزاری که تجزیه را براساس مکاشفه‌های[3] متعدد مانند توافق‌های نام‌گذری فایل‌ها، ساختار تیم تولید، ساختار فهرست[4] یا معیارهای نرم‌افزاری ترتیب می‌دهد، انجام می‌گیرد. سپس تولیدکننده به‌طور دستی خوشه‌بندی خودکار تولید شده را با استفاده از شناخت کنونی سیستم[5] و مستندات موجود آن، اصلاح می‌کند. اطلاعات تجزیه‌سازی به همراه حقایق استخراجی در قالب TA ذخیره می‌شوند.

در نهایت یک ابزار خودکار طرح‌بندی[6]، حقایق ذخیره شده را در جهت ایجاد نمودارهایی مانند آن‌چه در شکل‌های ۲-۶ و ۲-۷ آمده است، پردازش می‌نماید. ابزار طرح‌بندی در جهت کاهش خطوط متقاطع

[1] مختصر Tuple-Attribute Language، که برای آشنایی بیشتر با آن می‌توان به منابع [۱۵،۱۴] مراجعه نمود.

[2] Clustering

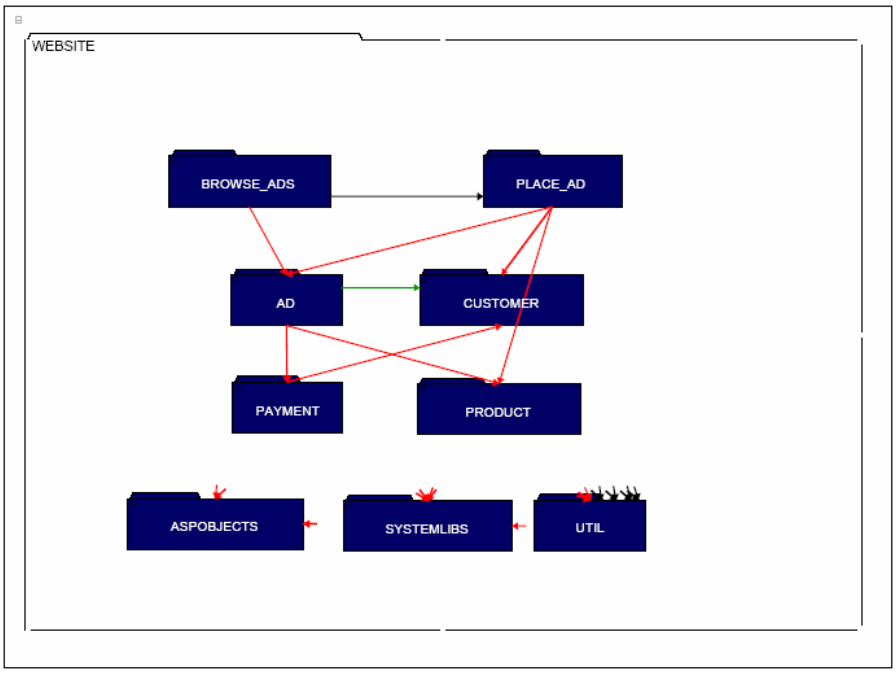
[3] Heuristics

[4] Directory

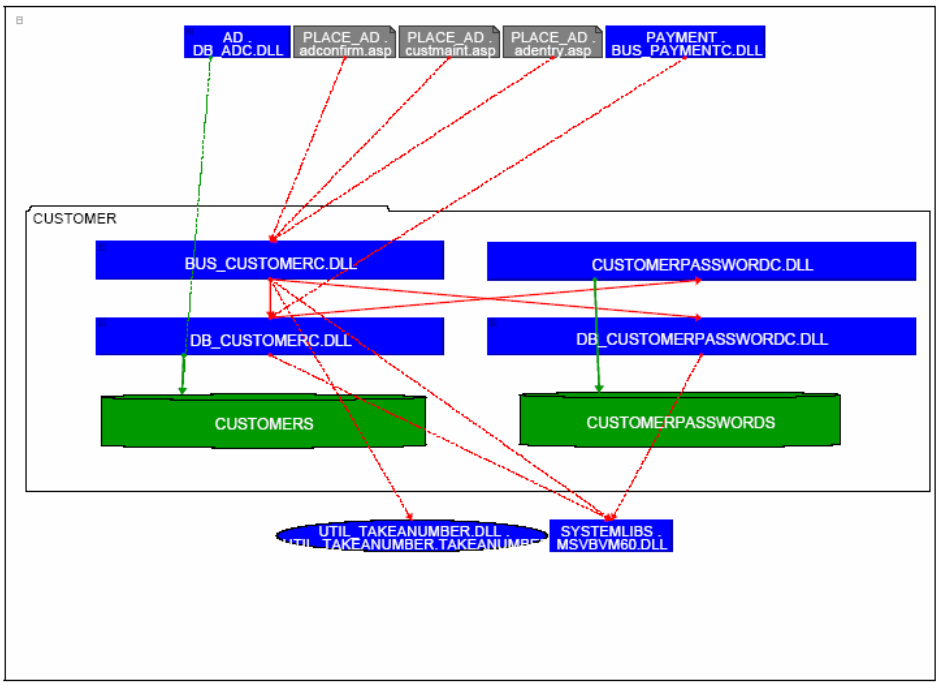
[5] Domain Knowledge

[6] Layout

در نمودارهای ایجاد شده از معماری، تلاش خواهد کرد. تولیدکننده می‌تواند طرح‌بندی ایجاد شده را تغییر دهد [در جهت بهبود].



شکل ۲-۶ یک نمونه معماری بازیابی شده

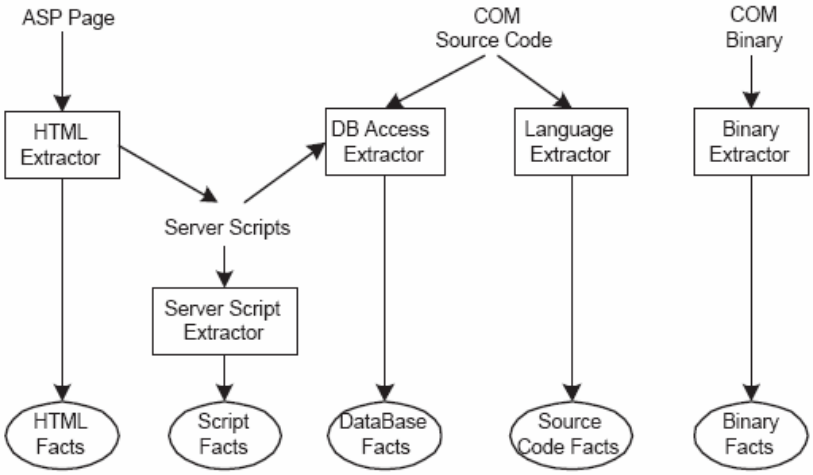


شکل ۲-۷ زیرسیستم Customer از شکل قبل

فرایند فوق حمایت ابزار و تفسیر انسان را برای بازیابی معماری ترکیب می‌کند. حمایت ابزار، زمان بازیابی را به‌طور خارق‌العاده کم می‌کند، و تفسیر انسانی بالاترین معیار سازمان‌دهی و خوشه‌بندی انبوه اطلاعات استخراج شده است.

۲-۴-۱-۱-۴- استخراج‌گرها

برنامه‌های کاربردی تحت وب با استفاده از زبان‌های برنامه‌نویسی متعدد ایجاد می‌شوند و از قطعاتی ترکیب شده‌اند که ممکن است کد مبدأ نداشته باشند یا برای آن‌ها استخراج‌گر خوبی یافت نشود. ویژگی‌های برنامه‌های کاربردی تحت وب مسایل متعددی را فراروی چارچوب‌های[1] بازیابی معماری نرم‌افزارهای تجاری‌ای قرار می‌دهد که معمولاً به یک استخراج‌گر وابسته‌اند. رویکردی که در این‌جا بررسی می‌شود، از یک مجموعه استخراج‌گرها استفاده می‌کند که برای استخراج حقایق از برنامه‌های کاربردی تحت وب آزمایش شده، همکاری می‌کنند. شکل ۲-۸ دورنمایی[2] از استخراج‌گرهای متنوع، ورودی‌های آن‌ها و نوع حقایقی که استخراج می‌کنند را نشان می‌دهد. پنج نوع استخراج‌گر به کار گرفته شده‌اند: استخراج‌گر HTML، استخراج کننده اسکریپت‌های کارگزار، استخراج‌گر دسترسی به پایگاه‌داده‌ها، استخراج‌گر زبان و استخراج‌گر دودویی[3]. هر استخراج‌گر مسوول آزمون یک قطعه یا بخشی از یک قطعه و تولید حقایق مناسب است.



شکل ۲-۸ معماری مفهومی استخراج کننده حقایق

[1] Frameworks
[2] Overview
[3] Binary

زمانی که تمام حقایق استخراج شدند، اطلاعات خوشه‌بندی ترکیب می‌شوند و تمام داده‌ها (حقایق و خوشه‌بندی) در جهت کاهش پیچیدگی آن‌ها مورد پردازش قرار می‌گیرند. سپس یک ابزار طرح‌بندی، حقایق را برای نمایش توسط یک مصورساز، آماده می‌کند.

استخراج‌گرهای متنوع، توسط یک اسکریپت پوسته[1] که ساختار درختی کد مبنای برنامه‌کاربردی تحت وب را می‌گردد، فراخوانی می‌شوند[2]. اسکریپت نوع قطعه را تعیین می‌کند و استخراج‌گر متناظر را فراخوانی می‌کند. برای مثال، اگر اسکریپت دریابد که فایل کنونی یک فایل دودویی است، استخراج‌گر دودویی صدا زده می‌شود. هر استخراج‌گر مجموعه‌ای از حقایق را تولید می‌کند و نتایج‌اش را در فایلی با همان نام فایل ورودی به‌علاوه نام استخراج‌گر به عنوان پسوند، ذخیره می‌کند. سپس یک اسکریپت دیگر فهرست‌های پردازش شده‌ی پیشین را می‌گردد[3] و تمام فایل‌های تولید شده را در فایلی به نام THEFACTS ادغام می‌کند. فایل THEFACTS با اطلاعات خوشه‌بندی شده که برنامه‌ی کاربردی تحت وب را به زیرسیستم‌هایی تجزیه می‌کند، ترکیب می‌شود.

۲-۴-۱-۲- روش دوم

این راه‌کار [۸] روشی را که برگرفته از عرصه‌ی مهندسی معکوس است و ابزاری را که از فعالیت‌های مهندسی معکوس در وب حمایت می‌کند، در جهت نگهداری، درک و تحول برنامه‌های کاربردی تحت وب ارائه می‌کند. این روش با عنوان مجموعه‌ای از نماهای مجرد، که توسط نمودارهای UML مدل شده‌اند و برای سهولت مقایسه در یک سلسله مراتب از سطوح تجرید سازمان یافته‌اند، عرضه شده است. نویسندگان مقاله معتقدند که نمایش‌مناسب، که بطور خودکار یا نیمه‌خودکار از برنامه‌کاربردی تحت وب کنونی ایجاد شده است، می‌تواند فرآیند تغییرات برنامه‌کاربردی تحت وب را ساده‌تر سازد.

۲-٤-۱-۲-۱- مدل‌سازی برنامه‌کاربردی تحت وب

در این مقاله، از الحاقی که جی کنالن[4] برای مدل‌سازی و تولید برنامه‌های کاربردی تحت وب، بر UML افزوده است[۱۳,۳۹]، جهت مدلسازی برنامه کاربردی مبتنی بر وب استفاده شده است. بطور خلاصه هم

---

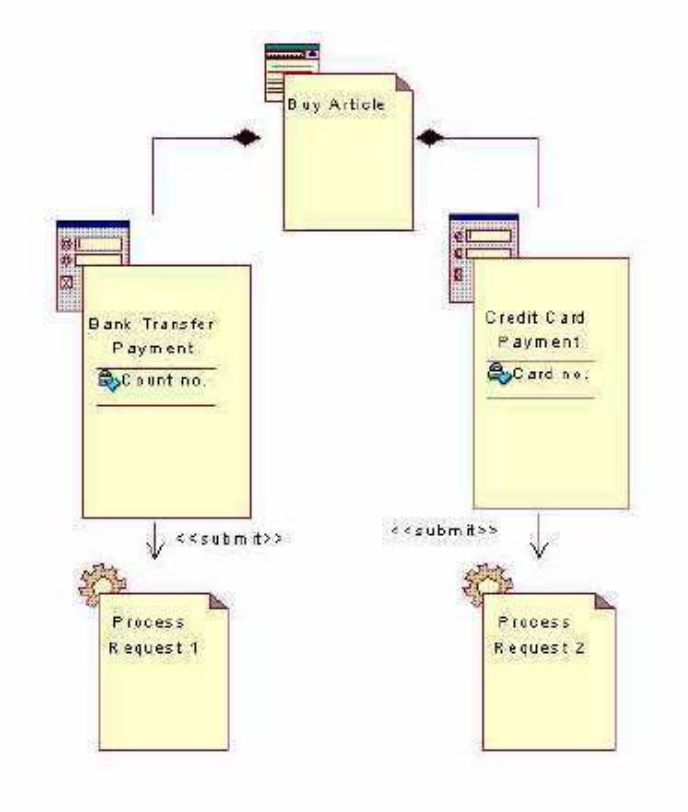
[1] Shell Script
[2] Invoke
[3] Crawls
[4] J. Conallen

نمودارهای ایستا (نمودارهای کلاس، مورد استفاده[1]، قطعه[2] و توسعه[3]) و هم پویا (نمودارهای ترتیب و همکاری[4]) مانند زیر توسعه یافته و به‌کار گرفته شده‌اند:

- **نمودارهای مورد استفاده**: اغلب در مراحل آغازین تحلیل، برای برجسته کردن محاوره‌ی یک عامل با سیستم تولید می‌شود (محتویات اطلاعاتی همان هستند که در تولید یک برنامه‌ی تجاری است). آن‌ها رفتار یک فرم برنامه‌کاربردی تحت وب را از دیدگاه کاربر مدل می‌کنند.
- **نمودارهای کلاس**: صفحات و رابطه‌ی صفحات را مدل می‌کنند; هر صفحه به عنوان یک کلاس مورد توجه قرار می‌گیرد; تمایز بین انواع مختلف از صفحات (صفحات مشتری، کارگزار، مجموعه قاب‌ها و ...) توسط کلیشه‌ها[5] مشخص می‌گردد. قطعات صفحه‌ها (اسکریپت‌های مشتری و کارگزار، قاب‌ها، اپلت‌ها و ...) نیز همچنین با کلاس‌ها (و نمونه‌های ایجاد شده از آن‌ها با اشیا و نمودارهای اشیا) نمایش داده شده‌اند. رابطه‌ها مفاهیم تصویری را به شکل مصور مدل می‌کنند (برای مثال یک صفحه‌ی کارگزار یک نام متناظر دارد که با هر صفحه‌ی مشتری که تولید می‌کند ایجاد خواهد شد; یک صفحه می‌تواند تجمعی از فرم‌ها باشد; یک کلید رادیویی مصورسازی‌ایی از یک فیلد ورودی است). نمودارهای کلاس می‌توانند برای مدل کردن معماری ایستای یک برنامه‌کاربردی تحت وب به کار بروند.
- **نمودار قطعه یا نمودار بسته**: نمایانگر رابطه‌ی بین منابع (کتابخانه‌ها، DBMS، صفحات مشتری و صفحات کارگزار) مختلف‌اند که برنامه‌کاربردی تحت وب را ایجاد می‌کنند. نمودار قطعه، معماری برنامه‌کاربردی تحت وب را مدل می‌کند.
- **نمودارهای توسعه**: بر پیکربندی زوج‌های مختلف موجود در یک پروژه متمرکز است و بر مکان‌ها و ارتباطاتشان. این نمودارها بسیار در مقایسه‌ی برنامه‌های کاربردی تحت وب مفیداند.

---

[1] Use Case
[2] Component
[3] Development
[4] Collaboration
[5] stereotypes



شکل ۲-۹ ترکیب بین صفحات و فرم‌ها و ثبت به صفحات کارگزار

۲-٤-۱-۲-۲- رویکرد مهندسی معکوس

فرآیندهای مهندسی معکوس توسط اهداف، مدل‌ها و ابزارها مشخص می‌شود. در حالی که ابزار در پی پوشش دادن به کل فرآیند هستند، اهداف و مدل‌ها در پی تعیین هسته فرآیند مهندسی معکوس هستند. اهداف بر انگیزه‌های مهندسی معکوس متمرکزند، و در تعیین مجموعه‌ای از نماهای مجرد برای برنامه مهندسی معکوس شده کمک می‌کنند. مدل‌ها در تعریف اطلاعاتی که باید بسط داده شوند دخالت دارند، و معمولاً توسط نمایش میانی نماها کامل می‌شوند. چون رفتار برنامه‌کاربردی تحت وب از عناصر پویا و ایستا سرچشمه می‌گیرد، مهندسی معکوس بایستی موارد زیر را بازیابی کند:

- معماری ایستا
- تبادلات[1] پویا

[1] Interaction

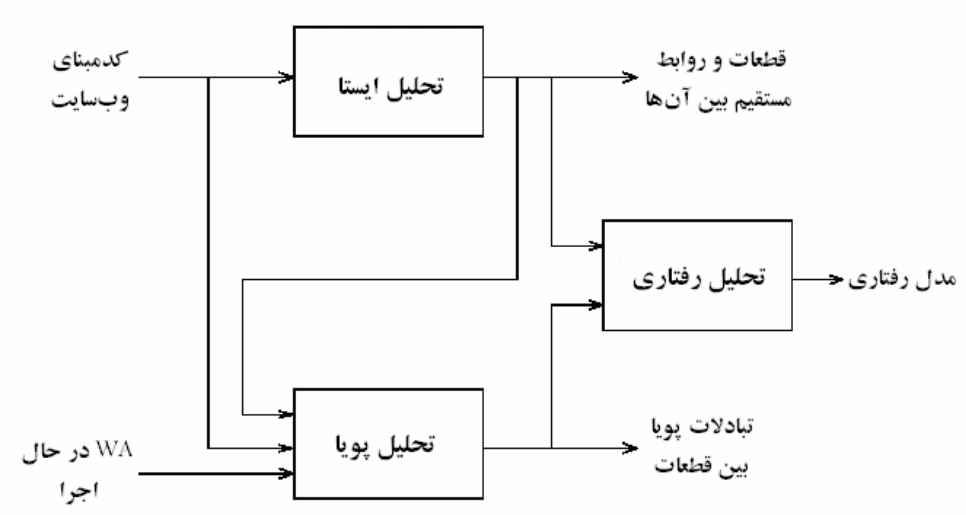
- رفتار

اهداف و مدل‌ها با توجه به عناصر بالا عرضه می‌شوند، و متعاقباً ما نیز فرآیند مهندسی معکوس مشتمل بر مراحل زیر را پیشنهاد می‌کنیم:

۱. تحلیل ایستا
۲. تحلیل پویا
۳. تحلیل رفتاری

مراحل فوق نماهایی را موجب می‌شوند که می‌توانند به‌طور مکفی توسط نمودارهای UML توسعه‌یافته نمایش داده شوند. در واقع در این روش نمودارها چنین مطابقت داده شده‌اند: نمودارهای کلاس برای نمایش معماری برنامه‌های کاربردی تحت وب، نمودارهای ترتیب و همکاری برای نمایش مدل پویا و نمودارهای مورد استفاده برای نمایش رفتار برنامه‌کاربردی تحت وب.

بازیابی به کمک نمودارهای UML به عنوان یک مرحله مقدماتی، به محلی نمودن و شناسایی عناصری مانند صفحات، قاب‌ها، فرم‌ها، اسکریپت‌ها (عناصری که برنامه‌کابردی را تشکیل می‌دهند) و روابط بین آن‌ها نیاز دارد.



شکل ۲-۱۰ فرآیند مهندسی معکوس یک برنامه‌کاربردی تحت وب

این به خودی خود یک عمل پیچیده مشتمل بر تجزیه‌ی[1] یک فایل چند زبانه است که با HTML، زبان‌های اسکریپتی و کد جاوا نیز آمیخته گشته است. علاوه بر آن برای تأمین استقلال از محیط و ابزار، و برای تأمین انعطاف سطح بالا اطلاعات بازیابی شده به یک نمایش میانی که در یک انبار[2] ذخیره شده، نگاشت می‌شوند.

۲-٤-۱-۲-۱- تحلیل ایستا

تحلیل ایستا برنامه را اجرا نمی‌کند، بلکه قطعه‌های معماری برنامه‌کاربردی تحت وب و روابط ایستای بین آن‌ها را بازیابی می‌کند. فایل‌های HTML، ساختار فهرست‌ها و کدهای زبان‌های اسکریپت مانند هر اطلاعات ایستای دیگری (از قبیل ساختار پایگاه داده و کد اپلت‌ها و سرولت‌ها[3]) پردازش خواهند شد.

صفحات HTML و زیرعناصرصفحات (مشتمل بر قاب‌ها، فرم‌ها و غیره) صفحه داده شده را به صورت محلی، دسته‌بندی شده و ثبت شده در یک نمایش میانی، ترکیب (عرضه) می‌کنند.

قلب فرآیند مهندسی معکوس، نگاشت بین عناصر برنامه‌کاربردی تحت وب و موجودیت‌های شیء گرا، مطابق با پیشنهادهای کن‌آلن [۱۳,۳۹] است و در اینجا نیز فرض شده که صفحات HTML و زیر عناصر مرتبط (مانند ارتباطات پایگاه‌داده[4]) به کلاس‌ها نگاشت می‌شوند و این در حالی است که پیوندها به روابط نگاشت می‌گردند. به زبان دیگر هر قطعه‌ی مشخص، کاندیدی برای یک کلاس است، در حالی که ارتباط بین صفحات یا عناصر موجود در صفحات، به عنوان کاندیدهای روابط شناخته می‌شوند. پارامترهای هر قطعه به عنوان خصایص کلاس‌ها شناخته می‌شوند یا به عنوان کلاس‌هایی جدید. پارامترهای ساده[5] می‌تواند مانند خصیصه‌ها مدل شود، در حالی که بقیه (مانند ارتباطات پایگاه‌داده) ممکن است نیاز به معرفی کلاس‌های جدید داشته باشند، برای آگاهی از جزییات به مرجع [۱۳] مراجعه کنید.

فاز تحلیل ایستا را می‌توان به فعالیت‌های زیر تجزیه کرد:

- **ذخیره‌سازی**[6] (برای مثال فایل‌های برنامه‌کاربردی تحت وب، پایگاه داده‌ها و بطورعام قطعه‌ها)
- **محلی‌سازی قطعات**
- **بازیابی رابطه‌ها**

---

[1] Parsing
[2] Repository
[3] Servlet
[4] Database Connections
[5] Elementary
[6] Inventory

- **تولید نمایش میانی**

نمایش میانی حاصل شده، مخزنی[1] ایجاد می‌کند که پرس‌وجوها و نماها از آن ساخته خواهند شد. لازم به تکرار نیست که در انتهای مرحله تحلیل ایستا اولین تقریب از نمودارهای کلاس برنامه‌کاربردی تحت وب فراهم می‌شود. البته قطعات آماده‌ی تجاری[2] یا پایگاه داده‌های تجزیه[3] نشده، ممکن است برای تولید مستندات HTML که توسط مرحله تحلیل ایستا قابل بازیابی نیستند، مورد استفاده قرار گرفته باشد.

۲-٤-۱-۲-۲-۲- تحلیل پویا

این مرحله بر مبنای نتایج تحلیل ایستا استوار است. برنامه‌کاربردی تحت وب اجرا می‌شود و محاوره‌ی پویا بین قطعات توصیف شده در نمودار کلاس، ثبت می‌گردد. تحلیل پویا با *مشاهده* اجرای برنامه‌کاربردی تحت وب، *ردگیری*[4] *کردن* هر رویداد یا عمل در کدمبنا (و متعاقباً کلاس‌های نمایش داده شده در نمودار کلاس) انجام می‌شود.

رویدادهای ردگیری شده آن‌هایی هستند که توسط کاربر قابل مشاهده‌اند یا با قطعه‌های خارج از برنامه‌کاربردی تحت وب (مانند پایگاه‌داده‌ها یا وب‌سایت‌های دیگران) مرتبط‌اند. رویدادها عبارتند از نمایش[5] صفحات، قاب‌ها یا فرم‌ها. همچنین ثبت فرم‌ها، پردازش داده، پیمایش پیوندها[6]، پرس‌وجوی پایگاه داده‌ها نیز رویداد به حساب می‌آیند.

تمام عناصر سرچشمه‌ی این اعمال[7] (به‌خصوص پیوندها، اپلت‌ها و اسکریپت‌ها) و عمل‌های داده شده به متدهای[8] کلاس‌های استخراجی، محلی می شوند . دنباله اعمالی که با یک رویداد شروع می‌شوند، یا از جریان کنترلی برنامه‌کاربردی تحت وب سرچشمه می‌گیرند (مانند دسترسی به یک پایگاه‌داده که با پرکردن یک فرم دنبال می‌شود)، یا دنباله اعمالی که از اعمال کاربر ناشی می‌شوند (مانند کلیک کردن بر یک پیوند یا پر کردن یک فرم) متناظر با[9] تبادل دنباله‌ای از پیام‌ها بین اشیای برنامه‌کاربردی تحت وب هستند.

---

[1] Repository
[2] Commercial on-the-shelf
[3] Parse
4 Trace
[5] Visualization
[6] Traversal
[7] action
[8] Methods
[9] Associated

این دنباله را می‌توان با نمودار ترتیب(یا با نمودار همکاری) نشان داد. البته باید توجه داشت که اطلاعات پویا همچنین برای چک[1]، اعتبار سنجی[2] و درنهایت تکمیل نمودار کلاسی که از مرحله‌ی قبل به‌دست آمده است، به‌کار می‌رود. به بیان دیگر انبار اطلاعات تکمیل می‌شود، اضافه می‌گردد و توسط مرحله[3] تحلیل پویا مورد استفاده قرار می‌گیرد.

فاز تحلیل پویا را می‌توان به فعالیت‌های زیر تجزیه کرد:

- **اجرای برنامه کاربردی مبتنی بر وب:** در زمان اجرای برنامه‌کاربردی تحت وب، کد مبنا و نمودارهای کلاس ردگیری می‌شود
- **تعیین صحت[4] و اعتبارسنجی[5]:** نمودار کلاسی که به ازای هر صفحه نمایش داده شده در نظر گرفته شده است، بایستی مشتمل بر یک کلاس منطبق بر همان صفحه باشد، برای هر قطعه به‌کارگرفته شده در صفحه بایستی یک کلاس در نظر گرفته شود؛ رابطه‌های مقتضی توسط تناظرهای[6] بین کلاس‌ها نمایش داده می‌شوند
- **تشخیص تکرارها:** عناصری که توسط پیام‌ها (سرمنشأ آن‌ها از جریان کنترل برنامه‌کاربردی تحت وب یا اعمال کاربران است) با هم محاوره دارند، تشخیص داده می‌شوند و با کلاس‌ها ردگیری می‌شوند.
- **تجرید نمودارهای ترتیب و همکاری:** این نمودارها برای تشریح سناریوهای عملی مورد بازیابی قرار می‌گیرند

این نکته لازم به یادآوری نیست که تشخیص مرحله محاوره[7]، آن عناصر "فعالی" را که مسوول "فعالیت‌های" برنامه‌کاربردی تحت وب هستند، محلی می‌کند. این قطعه‌ها می‌توانند اسکریپت‌ها باشند، اپلت‌ها، پیوندهای ابرمتنی و ... . هر عمل انجام شده با یک سرویس (خدمت) در کلاسی که مسوول آن است، تناظر[8] دارد. در حالی که تحلیل ایستا می‌تواند بطور خودکار انجام گیرد، تحلیل پویا به تبادل با انسان نیاز دارد. فایل بایگانی کارگزار وب می‌تواند برای استخراج[9] ترتیب رویداد[10] که شبیه‌ساز تراکنش کاربر

---

[1] Verify
[2] Validate
[3] Phase
[4] Verification
[5] Validation
[6] Association
[7] Interaction
[8] Associated
[9] Extract
[10] Event

است به رود، در حالی‌که بارگزاری یک صفحه در مرورگر می‌تواند به عنوان یک رویداد قابل رویت کاربر مورد توجه قرار گیرد.

۲-٤-۱-۲-۲-۳- تحلیل رفتاری

تحلیل رفتاری یا عملکردی الزاماً مشتمل بر فرآیند تجرید، مبتنی بر کشف رفتار برنامه‌کاربردی تحت وب، از دیدگاه کاربر است. رفتار حاصله توسط نمودارهای جنبه‌ی استفاده تشریح می‌شود. این مرحله[1] می‌تواند به کارهای[2] زیر تجزیه گردد:

- **تحلیل نمودارهای ترتیب یا همکاری**: تمام تبادلات نمودارها برای خلاصه‌سازی[3] رفتارهای عملیاتی که در قالب نمودارهای جنبه‌ی استفاده[4] گروه شده‌اند، مورد تحلیل قرار می‌گیرند.
- **تعریف جنبه‌ی استفاده**: نمودارهای جنبه‌ی استفاده، Actorها، روابط uses و extends برمبنای رفتارهای عملیاتی تعیین می‌شوند.
- **خلاصه‌سازی نمودارهای جنبه‌ی استفاده**: جنبه‌ی استفاده‌های تعریف شده در مرحله پیشین در نمودارهایی با سطوح جزییات مختلف نشان داده می‌شوند.

۲-٤-۱-۲-۳- یک مدل مفهومی برای برنامه‌کاربردی تحت وب

یک مدل مفهومی باید تجریدهای نمایش‌دهنده برنامه، قطعه‌های آن و ارتباط بین آن‌ها را مشخص نماید. در ابتدا و حالت کلی‌تر برنامه‌کاربردی تحت وب می‌تواند مانند ترکیبی از صفحات HTML در نظر گرفته شود. یک صفحه یا گروهی از صفحات، مسوول رفتار تعریف شده برای برنامه‌کاربردی تحت وب هستند. صفحات[5] بر یک کارگزار وب نهاده[6] می‌شوند؛ کارگزار وب درخواست‌های کاربران را پردازش کرده و برای آن‌ها کد HTML، اسکریپت‌های مشتری، اپلت‌ها، تصاویر و غیره برمی‌گرداند. آنچه کارگزار وب به مشتری بر می‌گرداند، ممکن است با یک فایل فیزیکی روی کارگزار مطابق باشد یا نباشد: CGI bin،

---

[1] Phase
[2] Tasks
[3] Abstract
[4] Use case
[5] Pages
[6] Deploy

Servlet، Server-side include، ASP و تکنولوژی‌ها و ابزارهای مرتبط ممکن است صفحاتی متغییر[1] تولید کنند. رده‌بندی مقدماتی زیر در ارتباط با صفحات مورد توجه قرار گرفته است:

- *صفحات کارگزار:* (صفحاتی که بر کارگزار قرار گرفته‌اند) در مقابل صفحات مشتری (صفحاتی که در حقیقت برای مشتری فرستاده می‌شوند).
- *صفحات ایستا در مقابل صفحات پویا*؛ محتوی[2] صفحه ایستا ثابت است، در حالی که محتوای صفحه پویا در طول زمان متغییر است
- *صفحات ساده در مقابل صفحات دارای قاب*؛ صفحه می‌تواند توسط دستور ویژه‌ای به نام frame-set، به قاب‌هایی تقسیم گردد. صفحه‌ای که در یک قاب ظاهر می‌شود، توسط یک مقصد مشخص می‌گردد.
- *صفحات غیر پیوندی*[3] *در مقابل صفحات پیوندی*؛ یک صفحه یا قطعه‌ی صفحه ممکن است دارای پیوندهای ابرمتنی[4] به خود یا صفحات دیگر باشد. پیوندها در عوض می‌توانند پویا یا ایستا باشند، بدین معنی که پیوند قطعه همیشه یکی است یا اینکه می‌تواند در زمان اجرا تعیین گردد.

یک صفحه همیشه یک تجمع[5] یا ترکیبی[6] از قطعه‌های ریزتر[7] مانند متن[8]، تصویر، فرم‌های ورودی\خروجی، جعبه‌های متن، اشیای چندرسانه‌ایی (صدا و تصویر)، anchorها، اسکریپت‌ها، اپلت‌ها و غیره است. قطعه‌های صفحه می‌توانند فعال (مانند اسکریپت‌ها یا اپلت‌ها) باشند. یک قطعه فعال، قطعه‌یی است که برخی اعمال پردازشی مانند تبادل داده با سایر صفحات، را انجام می‌دهد.

یک صفحه، معمولاً یک صفحه‌ی کارگزار، می‌تواند به سایر اشیاء که امکان ارتباط برنامه‌کاربردی تحت وب با DBMS را برقرار می‌سازند، داده‌های برنامه‌کاربردی تحت وب را مدیریت می‌کنند یا امکان ارتباط با سیستم‌های دیگر را فراهم می‌سازند، پیوند گردد.

---

[1] Pages on-the-fly
[2] Content
[3] Unlinking
[4] Hypertextual
[5] Aggregation
[6] Composition
[7] Finer grained
[8] text

برای دستیابی به تجرید برنامه‌کاربردی تحت وب، لازم است تمام اطلاعات شناساینده صفحه، قطعه‌های صفحه، روابط[۱] بین صفحات (چه ایستا و چه پویا) و قطعه‌ها، از کدمبنا استخراج[۲] شوند. تجرید برنامه‌کاربردی تحت وب باید نمایشگر موارد زیر باشند:

- **معماری ایستای[۳] برنامه‌کاربردی تحت وب**
- **تبادلات پویای بین قطعه‌ها**
- **رفتار برنامه‌کاربردی تحت وب و نسبت دادن قطعه‌ها به هر رفتار داده شده**

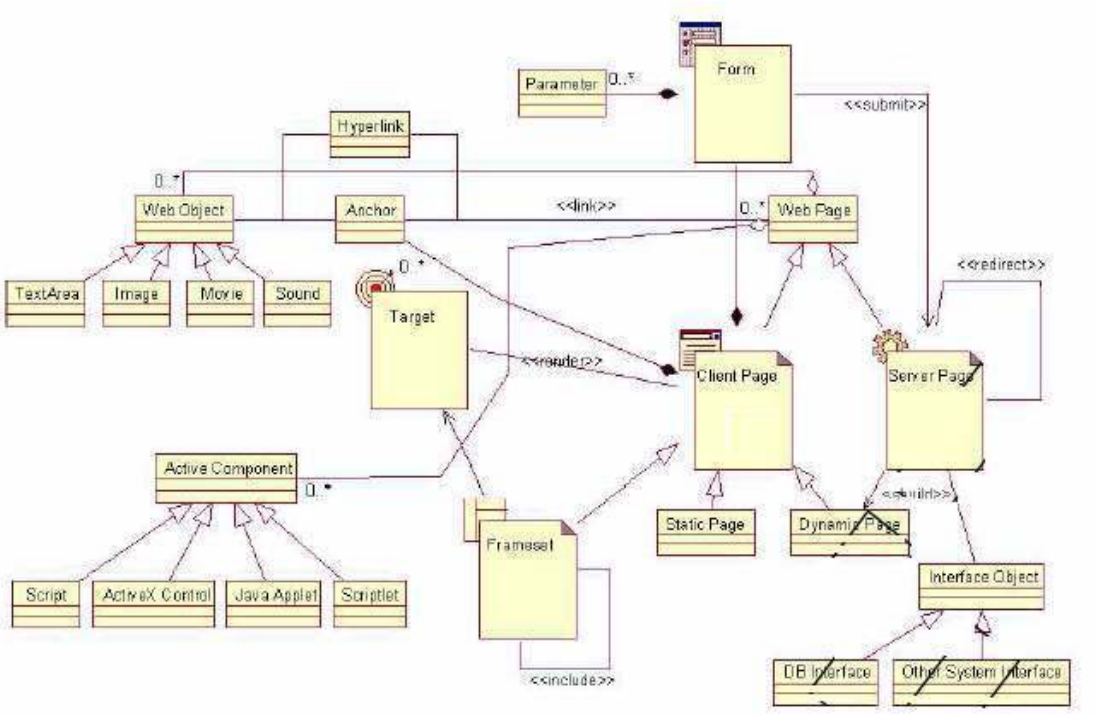
روش مهندسی معکوس پیشنهادی، مبتنی بر بازیابی قطعه‌ها که می‌تواند بر رفتار برنامه‌کاربردی تحت وب (و بنابراین در درک آن نیز) مؤثر باشد، است؛ منظور از قطعه[۴] یعنی صفحه‌ها، اسکریپت‌ها، اپلت‌ها، فرم‌های I/O، قاب‌ها و پیوندهای بازیابی‌شده، در حالی که مفاهیم دیگری از قبیل قالب متن[۵] مانند فونت‌ها و جداول مورد نظر نیستند.

شکل زیر نمایش دهنده یک نمودار کلاس است، که بیانگر مدل مفهومی نمایش میانی‌ایی است برای نشان دادن یک برنامه‌کاربردی تحت وب از آن استفاده شده است. هر صفحه مانند هر قطعه از صفحه، با یک کلاس مطابقت دارد، در حالی که تناظرها بیانگر پیوندهای بین صفحات هستند (یا ارتباط بین قطعه‌های صفحات)، رابطه‌های ترکیب یا تجمع برای توصیف دربرگیری[۶] اجزای[۷] یک صفحه توسط آن مورد استفاده قرار گرفته‌اند.

نمایشی[۸] که در این مقاله مورد استفاده قرار گرفته است، مشابه آن است که توسط جی‌کن‌آلن پیشنهاد گردیده است[۱۳]. پیشنهاد ایشان بر مهندسی مستقیم برنامه‌کاربردی تحت وب مبتنی است، ایشان یک مدل برای ایجاد برنامه‌کاربردی تحت وب با کمک UML همراه با برخی توسیع‌ها برای تطابق بهتر آن مدل‌سازی با برنامه‌کاربردی تحت وب عرضه کرده است.

---

[1] Relation
[2] Extract
[3] Static Architecture
[4] Component
[5] Format
[6] Inclusion
[7] Component
[8] Representation



شکل ۲-۱۱ مدل مفهومی یک برنامه‌کاربردی تحت وب

درحالی که توسیع UML عرضه شده توسط کن‌آلن، برای مهندسی مستقیم مطابقت داده شده است و نمایش عرضه شده در این مقاله بیشتر بر مهندسی معکوس متمرکز است، ولی باز هم این مدل عناصری را که مسوول رفتار و تبادلات پویای برنامه‌کاربردی تحت وب هستند، به‌خوبی نمایش می‌دهد. در این نمایش تفاوت بین صفحات ایستای مشتری (صفحاتی از مشتری که توسط فایل‌های ایستای HTML تعریف شده‌اند) و صفحات پویای مشتری (صفحاتی که بطور پویا در زمان اجرا ماهیت پیدا می‌کنند) به عنوان یک جزء مهم از رده‌بندی مورد توجه قرار گرفته است. علاوه بر این تفاوت بین اشیای[1] غیرفعال[2] وب (مانند تصاویر،صداها و فیلم‌ها) و اشیای فعال[3] (مانند اسکریپت‌ها) نیز مورد توجه قرار گرفته‌اند، همانطور که حضور اشیای واسط[4] (اشیایی که برنامه‌کاربردی تحت وب را به DBMS یا سیستم‌های خارجی مرتبط می‌کنند).

[1] Object
[2] Passive
[3] Active
[4] Interface

۲-٤-۱-۲-٤- حمایت ابزار

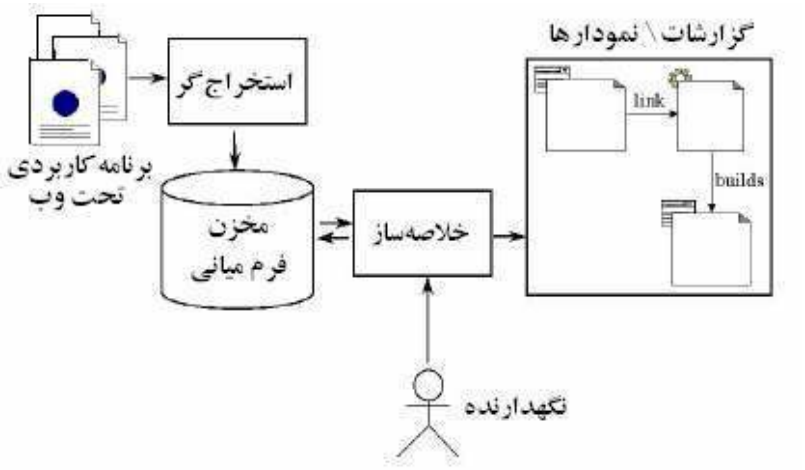
ابزارهای مهندسی معکوس در پی حمایت از فعالیت‌های مهندسی معکوس هستند، که اطلاعات را بطور خودکار از برنامه‌کاربردی تحت وب استخراج می‌کنند. اطلاعات بازیابی شده نمایشی تجریدی از برنامه‌کاربردی تحت وب با سطح درشتی[1] متفاوت که مشخص کننده اشیای برنامه‌کاربردی تحت وب و روابط بین آن‌هاست، تولید می‌کند. نمایش تجریدی درک برنامه و درک تبادل بین قطعه‌های آن در خلال مراحل مختلف چرخه زندگی برنامه‌کاربردی تحت وب (مشتمل بر مراحل نگه‌داری، شدن[2] و فعالیت‌های تست) را بالا می‌برد.

عیب اصلی ابزارهای کنونی مهندسی معکوس برنامه‌کاربردی تحت وب آن است که معمولاً مدل‌سازی صفحات پویا را در بر نمی‌گیرند، جایی که این مورد در کار باشد، صفات غیرطبیعی برنامه‌کاربردی تحت وب مشخص نمی‌گردند (بدین معنی که متدولوژی‌های تجاری به جای توسیع‌های مقتضی که در قسمت سوم تشریح گردید، به کار گرفته می‌شوند).

معماری ابزار پیشنهادی همانطور که در شکل ۲-۱۲ نشان داده شده است، متشکل است از: یک استخراج‌گر، یک انبار[3] نمایش میانی و یک خلاصه‌ساز[4]. ورودی یک برنامه‌کاربردی تحت وب است که قرار است فرآیند مهندسی معکوس بر آن انجام گیرد (مشتمل بر صفحات ایستا و پویا، تصاویر، اپلت و ...)؛ خروجی مجموعه‌ای از تجریدهاست.

- **استخراج‌گر:** این قسمت HTML، زبان‌های اسکریپتی مشتری و Sever-side را تجزیه می‌کند و یک نمایش میانی از هر اطلاعات مفیدی که به‌دست آورد، ایجاد می‌کند.
- **مخزن فرم میانی:** فرم میانی حاصل از اطلاعات بسط داده شده را ذخیره می‌کند
- **خلاصه‌ساز:** عملیات تجرید[5] را بر فرم میانی اعمال می‌کند و نمودارهای UML را که در بخش سوم تشریح شدند از آن‌ها بازیابی می‌کند.

[1] granularity
[2] Evolution
[3] Repository
[4] Abstractor
[5] Abstraction



شکل ۲-۱۲ معماری یک ابزار پیشنهادی

ابزار مذکور هم‌اکنون تحت تولید و پیاده‌سازی است، تا جایی که مخزن آن هم اکنون آماده است، به شیوه‌ای که اسکریپت‌نویسی جانب مشتری و کارگزار (JavaScript و VBScript) را حمایت می‌کند. یک مجموعه کامل[1] از پرس‌و جوها برای به دست آوردن نمودارهای کلاس، پیاده‌سازی شده‌اند، پرس‌و جوها و نماهای انعطاف‌پذیرتر همانند تجزیه‌ی سایر زبان‌های اسکریپت‌نویسی و برنامه‌نویسی(از قبیل PHP و java) تحت تولیداند.

۱-۱-۱-۱- روش سوم

در این روش [۱۶] یک رویکرد مهندسی معکوس برای ساخت نمودارهای UML در سطح سیستم کاری[2] در قلمرو برنامه‌های کاربردی تحت وب، ارائه شده است. این رویکرد امکان ساخت نمودارهای کلاس (برای مدل مفهومی برنامه کاربردی)، نمودارهای ترتیب (برای مدل‌سازی تبادلات بین اشیای سیستم کاری) و نمودارهای مورد کاربرد (برای تعیین قابلیت‌های برنامه کاربردی تحت وب) را فراهم می‌نماید. در رویکرد انتخابی این مرجع [۱۶] از معیارهای مکاشفه‌ای که کدمبنا را مورد بررسی قرار می‌دهند، برای بازیابی نمودارها استفاده شده است. ابزاری هم در این زمینه ساخته شده و کارایی آن با آزمایش تأیید گردیده است.

[1] Fixed
[2] Business

سطح تجرید بیان مطالب این مقاله نسبت به مورد قبل پایین‌تر است و درباره‌ی نحوه‌ی استخراج محصولات از برنامه کاربردی مبتنی بر وب کنونی با جزییات بیشتری صحبت کرده است. ایده در این مقاله آن است که مدل مفهومی که قلمرو برنامه را توصیف می‌کند نیز اطلاعات مناسبی برای نگهداری و تست در اختیار قرار می‌دهد. یکی از این مدل‌های مفهومی، مدل شیء‌گرا در سطح سیستم کاری است. البته لازم به ذکر است که بازیابی اشیا برای نرم‌افزارهای غیر مبتنی بر وب، قبلاً نیز صورت گرفته است و نتایج مفیدی داشته است[۱۶].

۲-۴-۱-۳- بازیابی نمودار کلاس در سطح سیستم کاری

رویکرد کنونی با بازیابی نمودار کلاس در سطح سیستم کاری شروع می‌شود. در یک نرم‌افزار کاربردی مبتنی بر وب، نمودار کلاس در سطح سیستم کاری قطعات مفهومی مرتبط در حیطه مسأله و روابط دوطرفه‌ی آن‌ها را توصیف می‌کند. بازیابی کلاس برای نرم‌افزارهای تجاری غیر مبتنی بر وب در سه مرحله انجام می‌گرفت:

۱. شناسایی کلاس‌های منتخب[1] و مشخصه‌های آن‌ها
۲. نسبت‌دادن متدها به کلاس‌های منتخب
۳. شناسایی روابط بین کلاس‌ها

(در ادامه واژه‌ی کلاس و شیء ممکن است به‌جای هم مورد استفاده قرار گیرد) موارد بالا را می‌توان برای برنامه کاربردی مبتنی بر وب نیز مورد استفاده قرار داد، اما پیش از آن باید برخی پرسش‌ها را که ناشی از تفاوت ذاتی آن‌هاست، پاسخ داد، که در این مقاله ابتدا به آن‌ها پرداخته شده است و سپس هر یک از موارد بالا برای برنامه‌کاربردی تحت وب بررسی شده است.

در نرم‌افزارهای تجاری نحوه‌ی یافتن اشیا و خصایص آن‌ها، جستجو به دنبال داده‌های منطقاً مرتبطی است که حالت شیء را تشکیل داده‌اند. این جستجو اغلب بر مبنای مکانیزم‌هایی در زبان برنامه‌نویسی است که اجازه می‌دهند گروه‌هایی از داده، مفهومی مرتبط (مانند آن‌هایی که برای تعریف ساختمان داده‌ها از قبیل رکورد، نوع داده‌های تعریف شده توسط کاربر و شمای جداول در پایگاه داده‌ها) را پیاده‌سازی نمایند. همچنین یافتن متدها نیز بر محوریت تیکه‌های مفید کد (مانند برنامه‌ها، زیرروال‌ها، برش‌ها[2] و ...)

[1] Candidate
[2] Slices

مبتنی است که می‌توان آن‌ها را با برخی اشیای مورد نظر که دارای بعضی معیارهای پیوستگی باشند، مرتبط کرد. در نهایت برخی معیارهای مکاشفه‌ای برای تعیین رابطه‌ی بین اشیا می‌توان به‌کار برد.

سه مورد بالا، در رابطه با برنامه‌های کاربردی تحت وب، به شکل زیر مطرح می‌شوند:

در رابطه با شناسایی اشیاء و خصایص آن‌ها در برنامه کاربردی مبتنی بر وب، انتخاب راه‌کار عمومی مورد استفاده برای پیاده‌سازی گروه داده‌ی مرتبط با هم در این برنامه‌ها واضح نیست. اغلب فن‌آوری‌ها و زبان‌های وب (مانند HTML، ASP، PHP، VBS، JSP و ...) از ساختارهای لغوی برای اعلان گروه‌های داده مانند رکوردست‌ها[1] یا مجموعه‌ها[2] یا کلاس‌ها استفاده می‌کنند که البته برخی از آن‌ها کم کاربردترند. علاوه بر آن چون برنامه کاربردی مبتنی بر وب اغلب به عنوان یک سیستم چند لایه، به همراه یک کارگزار پایگاه داده به عنوان یکی از لایه‌های آن معرفی می‌شود، یک تحلیل ساده کد آن ممکن است منجر به بازیابی ذخیره‌ی داده‌های ثابت و شمای آن‌ها نشود. پس ممکن است چنین توصیفی از ذخیره‌ی داده در لایه‌ی دیگری از برنامه صورت پذیرد و قابل دسترسی نباشد.

در رابطه با یافتن توده‌های کد که پیاده‌سازی متدهای اشیا را به آن‌ها منتسب نمود نیز نمی‌توان برای برنامه‌های کاربردی مبتنی بر وب همانند برنامه‌های کاربردی تجاری (مانند برنامه‌ها، زیرروال‌ها، برش‌ها و ...) رفتار نمود. در این رابطه واحدهای عملیاتی که باید مورد توجه قرار گیرند برمبنای درجه‌ی جزییات مورد درخواست، مشتمل بر صفحات وب، بلوک‌های اسکریپت داخل صفحات و توابع موجود در یک صفحه‌اند. علاوه بر آن معیارهای مناسب برای انتخاب بخش‌های کد جهت متدهای اشیا و انتساب آن‌ها به اشیای مناسب، باید به دقت تعیین شوند. به همین ترتیب لازم است قواعد مناسب برای تعیین روابط ممکن بین اشیای بازیابی شده معین شود.

در زیر رویکردهای پیشنهادی برای هر قسمت تشریح شده است:

۲-٤-۱-۳-۱- تعیین کلاس‌های منتخب و مشخصه‌های آن‌ها

رویکرد ارائه شده دارای دو مرحله است. مرحله‌ی اول مشتمل بر شناسایی گروه‌های مرتبط داده از کد برنامه‌ی کاربردی مبتنی بر وب است و در مرحله‌ی دوم یک روال خودکار برای تعیین کلاس‌های منتخب از بین گروه‌های تعیین شده در مرحله‌ی قبل، اجرا می‌شود.

---

[1] RecordSets
[2] Collections

عناصر مطلوب برای تعیین خصایص کلاس‌های منتخب در برنامه‌ی کاربردی مبتنی بر وب، گروه‌هایی از اشیای داده‌اند که درگیر در کار ورود و خروج داده‌های کاربرند (مانند داده‌های نمایش داده شده در فرم‌ها یا جداول HTML) یا در خواندن‌ها و نوشتن‌ها از پایگاه داده‌های یک ذخیره‌ی داده (مانند Recordset در ASP یا آرایه‌ای از داده‌های ناهمگن در زبان PHP) یا مجموعه‌داده‌ی مربوط به یک عمل پرس‌وجو در پایگاه داده. علاوه بر آن داده‌هایی که در بین صفحات متمایز یا نمونه‌های کلاس موجود در آن صفحات گذر داده می‌شوند را نیز می‌توان به حساب آورد.

توجیه این امر آن است که داده‌هایی که توسط کاربر وارد می‌شود یا به کاربر نشان داده می‌شود، اغلب مورد نظر کاربر محدوده‌ی برنامه است، همچنین داده‌هایی که از ذخیره‌ی داده خوانده یا در آن ثبت می‌شود نیز مفاهیم بامعنا در قلمرو سیستم کاری به شمار می‌آیند.

در مرحله‌ی مقدماتی فرآیند به یک تحلیل ایستای کد مبنا برای بازیابی این گروه‌های داده و ارجاعات آن‌ها در کد، نیاز است.

پس در مرحله‌ی اول کلمات موجود در گروه‌ها برای حل مسایل *هم‌معنایی*[1] (شناسه‌های[2] دارای نام‌های مختلف اما هم معنی) و *هم‌ریختی*[3] (شناسه‌های دارای هم نام) فیلتر می‌شوند. شناسه‌های هم‌معنی باید به یک شناسه‌ی یکتا نگاشت شوند و شناسه‌های هم نام به اسامی متفاوت. در ضمن در زمان انجام تحلیل هم‌معنا و هم‌ریخت، یک نام با مسما با توجه به نقش داده در قلمرو سیستم کاری، به آن منتسب می‌گردد. این موارد علاوه بر مطالعه و بررسی کد، نیازمند به مستندات در دسترس نیز هست و یک امر انسانی وقت‌گیر به شمار می‌آید. در این مرحله هر گروه از داده می‌تواند به عنوان خصایص یک کلاس بالقوه به شمار آید و یک اعتبارسنجی[4]، جهت تشخیص کلاس‌های سیستم کاری نیز می‌تواند انجام پذیرد. که این اعتبارسنجی اغلب به صورت دستی صورت می‌پذیرد.

جهت کاهش تلاش‌های مورد نیاز تحلیل، رویکرد عرضه شده یک روال خودکار برای تحلیل گروه‌های داده جهت تشخیص محتمل‌ترین گزینه‌های بامعنی برای کلاس را مورد استفاده قرار می‌دهد. بنابراین تنها گروه‌های داده‌ی انتخابی برای اعتبار سنجی برگزیده می‌شوند. این روال Produce_Candidate_Objects نامیده شده و براساس معیارهای مکاشفه‌ایی زیر است و در شکل ۲-۱۳ آمده است:

- GList لیست گروه‌های داده است

---

[1] Synonym
[2] Identifier
[3] Homonym
[4] Validation

- g گروه عمومی در GList است
- Card(g) کاردینالیتی گروه g است
- $N_{ref}(g)$ تعداد ارجاع‌ها به گروه است
- a یک فقره داده‌ی عمومی در g است
- CAND لیستی از اشیای منتخب است
- C یک شیء عمومی منتخب از CAND است
- SORT(A,K) تابعی برای مرتب‌سازی لیست A براساس معیار توصیف‌شده توسط K است
- TOP(A) تابعی برای دسترسی به عنصر بالای لیست A است
- REMOVE(A,x) روالی برای حذف یک مورد x از لیست A است
- INSERT(A,x) روالی برای قرار دادن مورد x در لیست A است
- ADD(C,h) روالی برای افزودن تمام موارد گروه h به گروه C است

روال Produce_Candidate_Objects لیست ورودی Glist را از گروه‌های داده، تحلیل می‌کند و لیست خروجی CAND را از کلاس‌های منتخب تولید می‌کند. اولین قاعده‌ی مکاشفه‌ای که توسط این روال پیاده‌سازی می‌شود، بیانگر این مطلب است که هر چه تعداد ارجاعات به یک گروه داده در کد بیشتر باشد، احتمال با معنا بودن آن بیشتر است. دومین قاعده‌ی مکاشفه‌ای آن است که گروه‌های با اندازه‌ی کوچک بیشتر نمایانگر مفاهیم ساده و غیرقابل تجزیه‌اند تا گروه‌های بزرگتر، همچنین گروه‌های بزرگتر ممکن است نشان‌دهنده‌ی مفاهیم پیچیده‌تر باشند که از اتصال گروه‌های کوچک‌تر ایجاد شده‌اند.

بر اساس دو قاعده‌ی فوق گروه‌های لیست Glist مقدمتاً به ترتیب نزولی تعداد ارجاعات هر گروه g و ترتیب صعودی کاردینالیتی هر گروه مرتب خواهند شد. این ترتیب با روال SORT که خروجی آن مشتمل بر لیست مرتب OrdList از گروه‌های داده است، تولید می‌شود.

با شروع از بالاترین گروه در OrdList روال فوق هر گروه را تحلیل می‌کند و اگر آن گروه دست‌کم مشتمل بر داده‌ی باشد که تاکنون در هیچ‌یک از گروه‌های دیگر CAND نباشد، در لیست اشیای منتخب در لیست CAND گنجانده خواهد شد. Ordlist تا زمانی‌که مشتمل بر حداقل یک گروه باشد یا تا زمانی که اجتماع تمام موارد داده‌ای گروه‌های Glist یکسان باشند.

زمانی که گروه h از لیست OrdList تمام موارد داده‌ای ایجاد کننده‌ی یک یا بیشتر گروه Ci در CAND را دارا باشد، تنها k قلم داده‌ی موجود در h که تاکنون در هیچ گروهی از CAND قرار نگرفته‌اند، به گروه Ci اضافه می‌شوند، که تمام عناصر آن در h هستند. دلیل آن است که گروه h احتمالاً نمایشگر

مفهوم مفهوم مرکبی است که توسط یک پیوند منطقی بین گروه‌های Ci ایجاد شده است. خصایص اضافه شده به گروه‌های Ci برای ثبت این پیوند لازم هستند، که برای اثبات ارتباط بین اشیا، بر طبق روش نشان داده شده در شناسایی ارتباط بین کلاس‌ها مورد استفاده قرار خواهند گرفت.

```
Procedure Produce_Candidate_Objects (in:
GList; out: CAND);

BEGIN
OrdList = SORT (Glist, Descending on
N_ref(g) AND Ascending on Card(g) );
CAND = ∅;
WHILE (OrdList ≠ ∅ OR (∪_i a ∈ C_i ≡ ∪_i a ∈ g_i )) DO
       h=TOP(OrdList);
       IF (∃ a∈h: a∉C ∀ C ∈CAND) THEN
              IF (!∃ C∈CAND: C ⊆ h THEN
                     INSERT(CAND, h)
              ELSE
                     ∀ C_i ∈ CAND: C_i ⊆ h DO
                            BEGIN
                            k = h - ∪_i C_i;
                            ADD(C, k);
                            END
              END IF
       END IF
       REMOVE (OrdList, g);
END WHILE
END
```

شکل ۲-۱۳ روال تولیدکننده‌ی لیست اشیای منتخب

۲-٤-۱-۳-۲- انتساب متدها به کلاس‌ها

در شناسایی متدها برای انتساب آن‌ها به کلاس‌ها به دو عامل نیاز است: درجه‌ی درشتی مقادیر کد که می‌توانند به عنوان متدهای بالقوه به حساب آیند و تعیین معیار انتساب متدهای بالقوه به کلاس‌ها. جستجو به دنبال مقادیر کم یا زیاد کد می‌تواند صورت پذیرد که هر یک مزایا و معایب خود را دارد، به عنوان مثال یافتن مقادیر کم کد برای انتساب آن به کلاس این ایراد را دارد که تلاش بیشتری برای استخراج قطعات از کد و مهندسی مجدد آن به عنوان متدهای شیء نیاز دارد.

رویکرد پیشنهاد داده شده صفحات فیزیکی (از قبیل صفحات کارگزار و مشتری) را به عنوان متدهای بالقوه که به اشیای منتخب در برنامه‌کاربردی تحت وب منتسب شوند، در نظر می‌گیرد و این امر را برای

قطعات داخل صفحه انجام نمی‌دهد. این کار سبب کاهش تلاش مورد نیاز برای مهندسی معکوس شئ‌گرای برنامه‌کاربردی تحت وب خواهد شد.

همچنین درباره‌ی تعیین یک معیار برای انتساب متدها به اشیا، رویکردی مورد توجه قرار گرفته است که در جهت کمینه نمودن پیوستگی بین اشیای متمایز تلاش دارد. معیار پیوستگی بین صفحات و اشیا بر اساس دسترسی صفحات به اشیای منتخب محاسبه می‌شود. یک صفحه زمانی به یک شیء منتخب دسترسی داشته که مشتمل بر دستوراتی بوده که مقدار برخی خصیصه‌های آن را تعریف نموده یا به آن دسترسی داشته است. فرض بر این است که نوع دسترسی‌ایی که مقدار را تعریف می‌کند بیش از نوع دسترسی‌ایی است که آن را مورد استفاده قرار می‌دهد.

بنابراین کمینه‌سازی پیوستگی بین اشیا، توسط انتساب هر صفحه به شیءایی است که بیشترین پیوستگی را با آن دارد. در عمل صفحه‌ایی که به صورت فراگیر به یک شیء دسترسی دارد، به عنوان متدی از آن شیء تعیین می‌شود. اگر شیئی توسط چند شیء مورد استفاده قرار گیرد، به آنکه بیشتر بدان ارجاع داده منتسب خواهد شد. در این حالت دسترسی صفحه به سایر اشیا می‌تواند به عنوان تبادل پیام بین شیئی که صفحه بدان منتسب شده است و سایر اشیا، به حساب آید.

جهت تعیین معیار انتساب اشیا به صفحات به تعاریف زیر نیاز است:

- $M=\{m_i\}$ مجموعه‌ی صفحات است (یا همان متدهای بالقوه برای اشیا، در ادامه واژه‌های صفحه و متد به جای هم به کار خواهند رفت)
- c عضو عمومی لیست CAND از اشیای منتخب است
- α_{def} عددی مثبت است که وزن انتساب داده شده به هر دسترسی به شیء از نوع *تعریف* را تعیین می‌کند
- α_{use} عددی مثبت است که وزن انتساب داده شده به هر دسترسی به شیء از نوع *استفاده* را تعیین می‌کند
- $\alpha_{def} > \alpha_{use}$
- $N_{def,m,c}$ تعداد دسترسی‌های از نوع *تعریف* توسط صفحه‌ی m به شیء c است
- $N_{use,m,c}$ تعداد دسترسی‌های از نوع *استفاده* توسط صفحه‌ی m به شیء c است

همچنین تابع Acc(m,c) که تعداد وزن‌دار دسترسی‌های m به c را نشان می‌دهد نیز به صورت زیر تعریف شده است:

$$Acc\ (m, c)= \alpha_{def} * N_{def,m,c} + \alpha_{use} * N_{use,m,c}$$

با در نظر گرفتن MAX[X] به عنوان تابعی که بیشترین مقدار مجموعه مقادیر X را برمی‌گرداند، معیار زیر برای انتساب یک متد به کلاس c مورد استفاده قرار می‌گیرد:

$$m \text{ is assigned to } c \in CAND \Leftrightarrow$$
$$Acc(m , c) = MAX_{cj \in CAND} [Acc(m , c_j)]$$

که بدان معنی است که صفحه‌ی m به عنوان یک متد به کلاس c منتسب می‌شود، اگر و تنها اگر c به گونه‌ای باشد که دسترسی‌های وزن داده شده‌ی m به c از تمام دسترسی‌های وزن‌دار m به سایر کلاس‌ها بزرگتر باشد. زمانی که در معیار فوق دو یا بیشتر از کلاس‌ها شرایط مشابهی داشته باشند، دخالت مهندس نرم‌افزار برای انتخاب انتساب صحیح m به یکی از کلاس‌ها لازم است. صفحاتی که به هیچ شیء‌یی دسترسی ندارند، به عنوان مؤلفه‌های همکار که اجرای سایر اشیا را کنترل می‌کنند، در نظر گرفته می‌شوند (در برنامه‌ی کاربردی مبتنی بر وب، این صفحه معمولاً صفحه‌ی اصلی است، یا صفحاتی که مرورگری کاربر در برنامه را میسر می‌سازند).

۲-٤-۱-۳-۳- شناسایی روابط بین کلاس‌ها

کلاس‌های منتخب در لیست CAND ممکن است مشتمل بر خصایص مشترکی باشند: این خصایص نمایانگر روابط بالقوه‌ی کلاس‌های درگیرند. این روابط با نمودارهای UML Association نمایش داده می‌شوند.

برای هر مجموعه کلاسی که دارای خطایص مشترک باشند، یک UML Association بین آن‌ها برقرار می‌شود. هر خصیصه‌ی مشترک تنها به یک کلاس از مجموعه منتسب می‌گردد و تمام این کلاس‌ها با روابط Association به هم پیوند می‌شوند. دخالت مهندسی نرم‌افزار برای انتسابات صحیح خصایص به اشیا لازم است.

مواردی که صفحات به بیش از یک کلاس دسترسی داشته باشند، سبب تولید روابط اضافه‌ی دیگری خواهد شد. به بیان دقیق‌تر صفحه‌ایی که دسترسی از آن صورت می‌گیرد، به شیئی نسبت داده خواهد شد که با آن بیشترین پیوستگی را دارد و برای سایر دسترسی‌ها بین کلاسی که صفحه بدان منتسب شده است و هر کلاس باقی مانده که صفحه به آن دسترسی دارد، یک رابطه تعریف می‌شود که از نوع UML Association است. در مرحله‌ی پالایشی بعدی روابط Association بازیابی شده را می‌توان برای تشخیص این‌که آیا رابطه‌های کنونی قابل جایگزینی با Specialization یا روابط تجمع و ترکیب هستند، تحلیل نمود.

در نهایت یک نمودار کلاس UML نمایانگر کلاس‌های بازیابی شده خصایص، متدها و روابط بین آن‌ها خواهد بود.

۲-۴-۱-۴- بازیابی نمودارهای UML مورد کاربرد و ترتیب

بازیابی نمودارهای مورد کاربرد

در این قسمت رویکرد مهندسی معکوس برای بازیابی نمودارهای مورد کاربرد و ترتیب از کدمبنای برنامه کاربردی تحت وب نمایش داده خواهد شد.

در ابتدا به بازیابی نمودارهای Use Case با استفاده از رویکرد خوشه‌بندی خودکار پرداخته خواهد شد که صفحات وب برنامه‌ی کاربردی تحت وب را به خوشه‌های بامعنی (با انسجام بالا) و مستقل (پیوستگی کم) گروه‌بندی می‌کند. رویکرد خوشه‌بندی درجه پیوستگی بین صفحات متصل به هم، بسته به چیدمان صفحات و ارتباطات را در نظر می‌گیرد. پیکربندی رویکرد خوشه‌بندی ارائه شده مشتمل است بر یک اعتبارسنجی و یک انتساب مفهوم. گروه‌های خوشه‌های اعتبارسنجی شده می‌تواند به Use Caseهای بالقوه از برنامه‌کاربردی تحت وب منتسب گردد. برای ایجاد نمودارهای Use Case، ارتباطات بین موارد کاربرد می‌تواند با تحلیل پیوندهای بین خوشه‌ها استنتاج گردد. اگر یک خوشه‌ی منتسب شده به مورد کاربرد A تنها به خوشه‌ی منتسب شده به مورد کاربرد B پیوند داشته باشد، یک رابطه <<include>> از جنبه استفاده‌ی A به مورد کاربرد B قابل تصور است و اگر یک خوشه به بیش از چند خوشه پیوند داشته باشد، یک رابطه‌ی <<extend>> بین مورد کاربرد متناظر با خوشه‌ی اول و سایر خوشه‌ها نیز امکان برقراری دارد. به این ترتیب یک سری پیشنهاد برای مهندس معکوس در زمینه‌ی ترسیم نمودارهای مورد کاربرد مصور است. در نهایت عامل‌های موجود در نمودارهای مورد کاربرد به موارد کاربرد منتسب با خوشه‌های شامل هر صفحه‌ایی که نیازمند به ورود داده از یک کاربر (به عنوان مثال صفحات‌وب شامل فرم‌ها) یا ارتباط با پایگاه داده است، پیوند می‌شود.

بازیابی نمودارهای ترتیب

در زمینه‌ی بازیابی نمودارهای ترتیب به‌ازای هر نمودار مورد کاربرد (خوشه‌ی معتبر)، می‌توان نمودارهای ترتیبی ایجاد نمود که اشیای آن‌ها از کلاس‌های متناظر با صفحات خوشه مشتق شده باشند و تبادلات[1] بین

[1] Interaction

اشیا از دسترسی‌های صفحه‌ایی که به یک متد از یک کلاس نسبت داده شده است به سایر کلاس‌ها صورت پذیرد.

بازیابی نمودار کلاس براساس یک فرایند مکاشفه‌ای صورت می‌پذیرد، که مراحل زیر را دربردارد:

- ترسیم یک نمودار ترتیب برای هر مورد کاربرد تعیین شده
- شناسایی صفحات مشتمل بر خوشه‌ی (یا گروه خوشه‌ها) متناظر با یک مورد کاربرد
- برای هر صفحه در خوشه، شناسایی کلاسی که صفحه‌ای به عنوان متد به آن نسبت داده شده است؛ برای هر کلاس شناسایی شده، یک شیء در نمودار قرار دهید
- اگر صفحه‌ای که به یک شیء از A منتسب شده است، به اشیای دیگری از کلاس‌های Bi دسترسی داشته باشد، بین شیء از A و شیء از B یک محاوره باید ترسیم نمود
- به هر تبادل یک نام بامعنی (هماهنگ با نام متد فراخواننده) نسبت داده شود و لیست پارامترهای مبادله شده بین اشیا تعیین گردد (که توسط تحلیل جریان داده بین موارد متصل برنامه‌کاربردی استنتاج می‌گردد)
- ترسیم مبادله بین یک شیء و یک عامل در صورتی‌که آن شیء به یک صفحه مشتمل بر فرم‌های ورودی یا خروجی منتسب شده باشد
- اگر رابطه‌ای بین صفحات منتسب با یک کلاس وجود داشته باشد، آن رابطه به عنوان یک فراخوانی[1] بین متدهای آن کلاس به حساب می‌آید و بنابراین یک "ارسال پیام به خود" در خط عمر شیء ترسیم خواهد شد

با توجه به علایم UML، در یک نمودار ترتیب، ترتیب زمانی درست از رویدادها (مانند تبادلات بین اشیا) را می‌توان از جریان تبادلات در جهت بالا به پایین در نمودار استخراج نمود. در نمودار ترتیب بازیابی شده این ترتیب زمانی را می‌توان از تحلیل ایستای جریان کنترل در کدمبنا بازسازی نمود. به عنوان یک نتیجه‌ی مقدماتی، ترتیب زمانی ممکن است با دنباله‌ی جملات لغوی منطبق باشد. بنابراین تحلیل پویا را می‌توان برای پالایش ترتیب زمانی ایستای تعیین شده مورد استفاده قرار داد.

[1] Call

۲-۴-۲- روش‌های مبتنی بر خوشه‌بندی

از رایج‌ترین شیوه‌های مهندسی معکوس برای نرم‌افزارهای کاربردی تحت وب، خوشه‌بندی است. پیش از پرداختن به روش‌های ذکر شده برمبنای خوشه‌بندی، مرور مختصری بر خوشه‌بندی در مهندسی نرم‌افزار جهت مهندسی معکوس آن، مفید به نظر می‌رسد. این امر در پیوست ۱ صورت گرفته است. پس از آن انتقال مفاهیم مهندسی معکوس با استفاده از فنون خوشه‌بندی به عرصه برنامه‌های کاربردی تحت وب و سپس در بخش بعد دو مقاله‌ی شاخص در این زمینه مورد بررسی قرار خواهد گرفت.

زمانی که فنون خوشه‌بندی به عرصه‌ای به‌خصوص (مانند برنامه‌های کاربردی تحت‌وب) اعمال می‌گردد، تصمیم‌گیری‌های متعددی باید انجام گیرند، که بر نتیجه‌ی نهایی اثرگذار خواهند بود[۱۱]:

شرح موجودیت[1] موجودیت‌هایی که باید خوشه‌بندی گردند، بر اساس مجموعه‌ای از خصوصیات که آن‌ها را متمایز می‌کنند، توصیف می‌شوند. تشابه بین موجودیت‌ها به تشابه بین خصایص موجودیت‌ها خلاصه می‌شود. خواص متفاوت به تشابهات متفاوت منجر می‌شود. در روش پیوندهای برادر[2]، موجودیت‌ها توسط موارد داخلی که هر موجودیت را در جداسازی از دیگران متمایز می‌کند، توصیف می‌شوند. به عبارت دیگر موجودیت‌ها زمانی با هم گروه‌بندی می‌شوند که خصیصه‌های مشابه را مورد پردازش قرار دهند. اما در روش پیوند مستقیم، ارتباطات[3] بین موجودیت‌ها برای تمایز آن‌ها مورد استفاده قرار می‌گیرد، یعنی زمانی موجودیت‌ها با هم خوشه می‌شوند که ارتباط دوطرفه‌ی آن‌ها یک زیرگراف قویاً متصل را تشکیل دهد.

گروه‌بندی موجودیت‌ها موجودیت‌ها (یا به‌طور عام‌تر، خوشه‌های موجودیت‌ها) زمانی که بسیار مشابه یا منسجم[4] باشند، خوشه‌بندی می‌شوند. اندازه‌گیری‌های متعدد دیگری از تشابه (یا تفاوت) و انسجام داخلی[5] (یا انسجام خارجی[6]) با توجه به شرح داده شده از موجودیت‌ها، امکان‌پذیرند. علاوه بر این، انتخاب‌های مختلفی در زمان مقایسه‌ی دو خوشه (به جای دو موجودیت)، برای تعیین تشابه (فاصله) یا انسجام داخلی(انسجام خارجی)بین‌شان، خارج از مقیاس‌های محاسبه شده برای موجودیت‌های داخلی آن‌ها، وجود دارد.

---

[1] Entity Description
[2] Sibling
[3] Connections
[4] Cohesive
[5] Cohesion
[6] Coupling

الگوریتم‌های خوشه‌بندی مراحلی که برای ایجاد خوشه‌بندی نهایی طی می‌شود، می‌توانند متفاوت باشند. رویکرد سلسله‌مراتبی[1] درختی می‌سازد که همه‌ی موجودیت‌های یک عمق درخت به خوشه‌هایی تقسیم‌بندی[2] می‌شوند. سطوح[3] بالاتر با خوشه‌های بزرگتر مرتبط‌اند[4]، و همینطور به طرف ریشه که می‌رویم، اندازه خوشه‌ها بزرگتر و تعداد آن‌ها کمتر می‌شود تا اینکه در ریشه تنها یک خوشه خواهیم داشت. برگ‌ها محتوی خوشه‌های تک‌عضوی[5] هستند. روش سلسله مراتبی می‌تواند به صورت بالا به پایین (الگوریتم‌های تقسیمی) یا پایین به بالا (الگوریتم‌های تجمعی[6]) انجام گیرد. خانواده‌ی دیگری از رویکردها بر مبنای تعریف یک مقیاس مؤلفه‌ای[7] که بایستی بهینه شود، است. در نهایت تنها یک خوشه‌بندی (تقسیم‌بندی موجودیت‌ها) آنهم با توجه به بیشترین ارزش معیارهای انتخابی برای اندازه‌گیری کیفیت خوشه‌بندی، تعیین می‌شود.

در زمینه‌ی خوشه‌بندی برنامه‌های کاربردی تحت وب، تاکنون دست‌کم سه روش که می‌توانند برای خوشه‌بندی صفحات وب مورد استفاده قرار گیرند، شناسایی شده‌اند، که این روش‌ها بر مبنای یکی از موارد زیراند[۱۱]:

۱. **ساختار**[8] ساختار نحوی[9] صفحات وب، همانند آن‌چه در درخت نحوی مجرد توسط یک پارسر[10] ایجاد می‌شود، شرح موجودیتی است که برای خوشه‌بندی مورد استفاده قرار می‌گیرد. تفاوت بین دو درخت نحوی به عنوان فاصله‌ی ویرایش درختی تعیین می‌شود و دو صفحه در صورتی با هم خوشه می‌شوند که بر اساس الگوریتم خوشه‌بندی تجمعی[11] کمترین فاصله را داشته باشند.

۲. **اتصال**[12] در اینجا **ابرپیوندهایی** که صفحات وب را به هم وصل می‌کنند، به عنوان شرح موجودیت مورد استفاده قرار می‌گیرند[۱۰]. زمانی‌که گروهی از صفحات دارای ابرپیوندهای زیادی باشند که آن‌ها را به هم وصل کرده باشند، به معیار انسجام مربوطه یک نمره‌ی بالا داده می‌شود. خوشه‌هایی تعیین خواهند شد که یک معیار "کیفیت خوشه‌بندی" را بیشینه نمایند.

---

[1] Hierarchical
[2] Partitioned
[3] Level
[4] Associate
[5] Singleton
[6] agglomerative
[7] Modularity
[8] Structure
[9] Syntactic
[10] Parser
[11] agglomerative clustering
[12] Connectivity

۳. **کلمات کلیدی** محتوی صفحات وب برای تعیین مجموعه‌ای از کلمات کلیدی که هر یک از آن‌ها را متمایز می‌سازد، مورد تحلیل قرار می‌گیرد. دو صفحه زمانی با هم خوشه می‌شوند که تعداد زیادی کلمه‌کلیدی را به اشتراک داشته باشند.

۲-۴-۲-۱- روش اول

نویسنده‌ی مقاله [۱۰] معتقد است که مشکل رویکردهایی که در مقالات پیش از آن درباره‌ی مهندسی معکوس در وب تشریح شده‌اند، آن است که این اطلاعات که بیشتر در قالب نمایش گرافیکی، مبتنی بر تکنیک‌های مصورسازی ارائه شده‌اند، اغلب برای درک برنامه‌های کاربردی تحت‌وب نسبتاً کوچک، مناسب‌اند. اما نمایش گرافیکی کارایی خود را برای برنامه‌های کاربردی بزرگ مقیاس، به تدریج از دست می‌دهد. رویکردی برای غلبه بر این محدودیت، عبارتست از ساختاردهی نمایش گرافیکی به بخش‌های کوچک‌تر. به عنوان مثال با جمع‌آوری گره‌ها و لبه‌ها به زیرگراف‌های منسجم بر اساس یک معیار انسجام.

خوشه‌بندی در این زمینه موارد زیر را مورد توجه قرار می‌دهد[۱۰]: انتخاب مدلی که قطعات برنامه‌های کاربردی تحت‌وب را به‌نحوکافی[1] توصیف کند، تعریف معیار تعیین کننده‌ی زمان خوشه‌بندی دوقطعه به یک واحد منسجم و تعریف الگوریتم خوشه‌بندیی که باید به کار گرفته شوند.

رویکرد ارائه شده در این مقاله برمبنای مدل مفهومی برنامه‌کاربردیی است که قطعات و ارتباطات بین آن‌ها را شامل می‌شود و نیز بر مبنای میزان[2] پیوستگی بین اجزای متصل‌به‌هم است که هم چیدمان[3] آن‌ها و هم چیدمان ارتباطات بینشان را به حساب می‌آورد. میزان پیوستگی توسط یک الگوریتم خوشه‌بندی سلسله‌مراتبی که یک سلسله‌مراتب از خوشه‌ها را ایجاد می‌کند، مورد استفاده قرار می‌گیرد. رویکرد مذکور برای برنامه‌های کاربردی تحت‌وب با اندازه‌ی متوسط نیز مورد آزمایش قرار گرفته است و پاسخ‌های امیدوارکننده‌ای تولید کرده است. در این مقاله ابتدا یک مدل مفهومی از برنامه‌ی کاربردی تحت‌وب که رویکرد خوشه‌بندی ارائه شده بر مبنای آن است، عرضه گردیده است.

---

[1] Adequately
[2] Measure
[3] Topology

۲-۴-۲-۲- مدل مفهومی برای برنامه‌های کاربردی تحت‌وب

ابتدا به یک دسته‌بندی سه کلاسه از وب‌سایت‌ها به ترتیب افزایش پیچیدگی اشاره شده است، که برنامه‌کاربردی با محتوی پویا در کلاس ۳ قرار می‌گیرد. یک برنامه‌کاربردی کلاس ۳ با تعداد زیادی از فناوری‌های به‌کارگرفته شده، سروکار دارد، از قبیل Java Servlets, PHP, Java Server Pages (JSP) و CGI, XML, ODBC, JDBC Microsoft's Active Server Pages (ASP). برنامه‌های کاربردی تحت‌وب با محتوی پویا، که صفحات آن‌ها در هوا[1] تولید می‌شوند، در این رویکرد مورد نظر بوده‌اند.

قطعات اصلی برنامه‌ی کاربردی مبتنی بر وب با محتوی پویا، همان صفحات آن درنظر گرفته شده‌اند، که می‌توانند به *صفحات کارگزار*، صفحاتی که بر کارگزار وب ایجاد می‌شوند[2]، و *صفحات* مشتری ،صفحاتی که کارگزار وب در پاسخ به مشتری پس می‌فرستد، تقسیم شوند.

صفحات مشتری می‌توانند به *صفحات ایستا*، اگر محتوی آن‌ها ثابت باشد و به روشی پایدار ذخیره شود، یا *صفحات ایجاد شده توسط کاربر*، اگر محتوی آن‌ها با زمان متغییر باشد و توسط کارگزار در هوا ایجاد شود، دسته‌بندی نمود. علاوه بر این یک صفحه‌ی جانب مشتری (که *صفحه‌ی کاربر با قاب* نامیده می‌شود) می‌تواند توسط یک ساختار صفحه‌ی مخصوص که *مجموعه قاب*[3] نامیده می‌شود، به *قاب‌هایی*[4] تقسیم شود. همچنین یک برنامه‌ی کاربردی مبتنی بر وب می‌تواند مشتمل بر *مؤلفه‌های مشتری* یا *اشیای وب*، مانند عکس‌ها، اشیای چندرسانه‌ای (صدا و فیلم)، اپلت‌ها[5]، اشیای فلش[6] و غیره .... باشد.

زمانی که یک صفحه (یا یک قطعه از صفحه) مشتمل بر پیوندهای ابرمتنی به خودش یا سایر صفحات باشد، رابطه‌ای که link نام دارد بین صفحه (یا قطعات صفحه) و سایر صفحات، در مدل، نمایش داده خواهد شد. همچنین رابطه‌ی submit[7] بین یک فرم و صفحه‌ی کارگزار که فرم داده را ایجاد[8] می‌کند، رابطه‌ی redirect بین یک اسکریپت و یک صفحه‌ی وب، رابطه‌ی include بین یک اسکریپت مشتری و مؤلفه‌ی[9] مشتری (یا بین دو صفحه‌ی کارگزار) نیز در مدل نمایش داده شده‌اند. رابطه‌ی build بین یک صفحه‌ی کارگزار و صفحه‌ی مشتری که به صورت پویا توسط آن ایجاد می‌شود، همانند رابطه‌ی

---

[1] On-the-fly
[2] Deploy
[3] Frameset
[4] Frames
[5] applets
[6] Flash
[7] Submit
[8] Elaborate
[9] Module

load_in_frames که بین یک صفحه‌ی مشتری دارای قاب و مجموعه صفحاتی است که با علامت[1] <frame> به آن‌ها ارجاع می‌دهد، ایجاد می‌گردد.

در رویکرد درک برنامه‌ی تحت وبی که در این مقاله آمده است، درجه‌ی درشتی[2] صفحات وب هم درنظر گرفته شده است و به این ترتیب لازم است به قطعات و روابط زیر نیز توجه شود. قطعات مشتمل بر web pages، server pages، client pages، client pages with frame، client modules و web objectها است. روابط نیز مشتمل بر link، submit، redirect، build، load_in_frames و include هستند.

۲-۴-۲-۳- رویکرد خوشه‌بندی

هدف رویکرد ارائه شده، گروه‌بندی قطعات نرم‌افزاری یک برنامه‌ی کاربردی مبتنی بر وب به خوشه‌های بامعنی (با انسجام بالا) و مستقل (دارای پیوستگی پایین) است. مقاله‌ی مذکور در قسمت قبل توصیف مدلی که برنامه با آن نمایش داده می‌شود را عرضه نمود. برای تعیین زمانی که مجموعه‌ای از قطعات به یک واحد منسجم خوشه‌بندی می‌شوند نیز پیوستگی بین قطعات بر اساس وجود ارتباط مستقیم بین آن‌ها کمی می‌شود. هرچه پیوندهای بیشتری بین اجزا باشد، پیوستگی بین آن‌ها بیشتر خواهد بود. به‌علاوه جهت تنظیم بهتر، یک استراتژی جهت وزن دادن به پیوندها برقرار شده که بر این فرض مبتنی است که درجه‌ی پیوستگی به تعداد بیشتر پیوندها وابسته است.

درنهایت، انتخاب سوم الگوریتم خوشه‌بندی است. الگوریتم‌های مختلفی برای خوشه‌بندی در مرجع [۹] مطرح گردیده است. روش کنونی با استفاده از خوشه‌بندی سلسله‌مراتبی پیشنهاد شده است که می‌تواند برای بخش‌بندی‌های مختلف[3] یک سیستم در سطوح متنوع از تجرید مورد استفاده قرار گیرد.

۲-۴-۲-۳-۱- تعیین پیوستگی بین قطعات برنامه‌ی کاربردی مبتنی بر وب

انتخاب یک معیار برای اظهار درجه‌ی پیوستگی یک زوج از قطعات در موفقیت الگوریتم خوشه‌بندی بسیار حیاتی است. تعریف مرجع کنونی از پیوستگی، برخی معیارهای ذاتی[4] مشتق شده از دانش و تجربه در زمینه‌ی تولید و نگه‌داری برنامه‌های کاربردی تحت وب را نیز به حساب آورده است.

---

[1] tag
[2] granularity
[3] Different Partitioning
[4] intuitive

این تجربه پیشنهاد می‌کند که هم *چیدمان* و هم *چیدمان* ارتباطات لازم است برای اظهار درجه‌ی پیوستگی بین قطعات در نظر گرفته شوند. بنابراین فرض بر این است که ارتباطات مورد نظر، پیوستگی‌های متفاوتی بین قطعات متصل ایجاد می‌کنند. مخصوصاً برخی فرض‌های ویژه درباره‌ی روابط build، redirect، link و submit در نظر گرفته خواهند شد.

یک رابطه‌ی build، بین صفحه‌ی کارگزار و صفحه‌ی مشتری‌ایی که تولید کرده است، برای ایجاد درجه‌ی قویتری از پیوستگی بین قطعات برنامه‌ی کاربردی مبتنی بر وب در نظر گرفته می‌شود، زیرا که وجود صفحه‌ی مشتری به وجود صفحه‌ی کارگزار بستگی دارد. یک رابطه‌ی redirect بین دو صفحه درجه‌ی بالاتری از پیوستگی نسبت به ارتباط link برقرار می‌کند، زیرا یک عبارت redirect معمولاً بر اجرای یک Elaboration از طریق انتقال کنترل جریان از یک صفحه‌ی به صفحه‌ی بعد دلالت دارد. همچنین، یک ربطه‌ی submit بین یک صفحه‌ی مشتری و یک صفحه‌ی کارگزار درجه‌ی بالاتری از پیوستگی نسبت به ارتباط link، تولید می‌کند، زیرا یک عبارت submit معمولاً نمایشگر یک Elaboration و یک جریان داده بین صفحات است. علاوه بر این به خاطر جریان داده، یک رابطه‌ی submit با پیوستگی‌یی بیش از redirect متناظر می‌گردد.

با این فرض‌ها ارتباطاتِ link، redirect، Submit به مقادیر مثبت Wl، Wr و Ws منتسب شده‌اند و رابطه‌های زیر نیز قرارداد شده است:

$$Wrl = Wr/Wl$$
$$Wsl = Ws/Wl$$
$$1 < Wrl < Wsl$$

نسبت‌های Wrl و Wsl برمبنای تجربه نسبت داده شده‌اند. برای مثال در آزمایش‌ها، با Wl=1، Wrl=2.4 و Wsl=3 نتایج خوبی حاصل شده است. همچنین درجه‌ی پیوستگی $C_{A,B}$ بین دو جزء به نام‌های A و B، به شکل زیر عنوان می‌گردد:

$$C_{A,B} = C_{AB} + C_{BA}$$

که C_{AB} معیاری از پیوستگی تولید شده توسط لبه‌ی از A به B است و برعکس C_{BA} معیار پیوستگی تولید شده با لبه‌ی B به A است.

ساده‌ترین راه برای اندازه‌گیری $C_{A,B}$ شمارش لبه‌های وزن‌دار خارج شده از A و وارد شونده به B است و به‌طور مشابه برای $C_{B,A}$ هم. همچنین پیوستگی بین گره‌های A و B به‌طور بدیهی باید زمانی که A به‌طور مستقیم به B دسترسی دارد (یا B به‌طور مستقیم به A دسترسی دارد)، نسبت به حالتی که A به B از طریق سایر گره‌ها دسترسی دارد، قوی‌تر در نظر گرفته شود. به خاطر به‌حساب آوردنِ متمایز این

چیدمان‌های متفاوت، یک رویکرد[1] وزن‌گذاری اضافی را نیز که به هر گره به ازای لبه‌های خروجی‌اش W_X^{OUT} یک وزن W_X^{OUT} و به ازای لبه‌های ورودی‌اش یک وزن W_X^{IN} منتسب می‌کند، را نیز تطبیق داده‌ایم. W_X^{OUT} (به هر لبه از نوع x که از گره‌ی مورد نظر خارج می‌شود، نسبت داده می‌شود) به عنوان fan-out یک گره و W_X^{IN} (به هر لبه از نوع x که به گره‌ی مورد نظر وارد می‌شود، نسبت داده می‌شود) به عنوان fan-in آن به حساب می‌آید.

درجه‌ی پیوستگی $C_{A,B}$ دو قطعه به شکل زیر بیان می‌شود:

$$C_{A,B}= C_{AB} + C_{BA}= p_{A \to B}^{*} p_{B \leftarrow A} + p_{B \to A}^{*} p_{A \leftarrow B} \qquad (1)$$

ضرب اول $P_{A\to B} * P_{B\to A}$ یک نماینده از توان انباشته‌ی[2] ارتباطات از A به B است. در حالی که دومی نمایانگر توان انباشته‌ی ارتباطات از B به A است. در حالت کلی با داشتن دو گره‌ی X و Y، عبارت $p_{X\to Y}$ نمایانگر توان به‌هم پیوستگی[3] بر اساس لبه‌های خارج شونده‌ی وزن‌دار و $p_{Y\to X}$ بیانگر توان به‌هم پیوستگی بر اساس لبه‌های وزن‌دار ورودی از Y به X است.

عبارت $p_{X\to Y}$ به شکل زیر بیان می‌شود:

$$p_{X \to Y} = N_{LINK}(X \to Y) \cdot w_{LINK}^{OUT}(X) + N_{SUBMIT}(X \to Y) \cdot w_{SUBMIT}^{OUT}(X) + N_{REDIRECT}(X \to Y) \cdot w_{REDIRECT}^{OUT}(X)$$

که $N_{LINK}(X\ Y)$، $N_{SUBMIT}(X\ Y)$ و $N_{REDIRECT}(X\ Y)$ تعداد ارتباطات از نوع link، submit و redirect از X به Y هستند. به همین ترتیب عبارت را $Py \leftarrow x$ نیز می‌توان جمع لبه‌های ورودی از Y به X تفسیر نمود.

در عبارت فوق، اگر حاصلضرب $P_{A\to B} * P_{B\to A}$ یک باشد، تمام لبه‌های خارج شده از A به B رسیده‌اند و لبه‌ای داخل شده به B وجود ندارد که از A نیامده باشد. اگر شرایط متقارن برای ضرب $P_{B\to A} * P_{A\to B}$ نیز برقرار باشد، درجه‌ی پیوستگی $C_{A,B}$ حداکثر در نظر گرفته می‌شود، که برابر با عدد دو است. حداقل مقدار برای $C_{A,B}$ صفر است، که اشاره به زمانی دارد که گره‌ها به‌طور مستقیم به هم متصل نیستند.

---

[1] Strategy
[2] Cumulative strength
[3] Interconnection

۲-۴-۲-۴- الگوریتم خوشه‌بندی

همانطور که می‌دانیم الگوریتم‌های خوشه‌بندی سلسله‌مراتبی تجمعی، از موارد منفرد شروع می‌کنند و آن‌ها را به خوشه‌های کوچک گرد می‌آورند که به خوشه‌های بزرگتر جمع خواهند شد تا زمانی که به یک خوشه که مشتمل بر همه است، ختم گردد. نتیجه سلسله‌مراتبی از خوشه‌هاست.

الگوریتم سلسله‌مراتبی تکراری است و از یک خوشه‌بندی با n خوشه شروع می‌شود، که هر یک مشتمل بر یک قطعه از برنامه‌ی کاربردی مبتنی بر وب است، و خوشه‌ی جدید را براساس ۴ قاعده‌ی خوشه‌بندی به وجود می‌آورد:

- R1: خوشه‌ای که مشتمل بر صفحه‌ی مشتری ایجاد شده[1] باشد، با خوشه‌ای که صفحه‌ی کارگزار ایجاد کننده‌ی صفحه‌ی مقدم را در برمی‌گیرد، ادغام می‌شود.
- R2: اگر و تنها اگر تمام صفحات مورد ارجاع توسط پرچم <frame> از یک صفحه‌ی قاب‌دار مشتری به یک خوشه تعلق داشته باشند، خوشه‌ی مشتمل بر صفحه‌ی قاب‌دار، با خوشه‌ی نماینده‌ی صفحات ارجاعی، ادغام می‌شود.
- R3: اگر و تنها اگر تمام صفحه‌های مشتری (یا کارگزار) که یک مؤلفه‌ی مشتری (یا کارگزار) را دربر داشته باشند، در یک خوشه باشند، خوشه‌ی دربردارنده‌ی صفحه‌های مقدم با خوشه‌ای که مؤلفه‌ی مشتری (یا کارگزار) را شامل است، ادغام خواهد شد.
- R4: جفت خوشه‌هایی که مقدار پیوستگی آن‌ها بیشترین است، در خوشه‌ای جدید گرد می‌آیند.

الگوریتم مذکور به گراف پیوستگی WAG=(N,E) برنامه‌ی کاربردی تحت وب اعمال می‌شود. شرح الگوریتم در شکل زیر آمده است، که n کاردینالیتی مجموعه گره‌ی N، c یک خوشه‌ی عمومی در روش خوشه‌بندی داده شده است و x نماینده‌ی هر رابطه‌ی link، redirect و submit است.

الگوریتم به یک سلسله‌مراتب از خوشه‌ها منجر می‌شود که هر یک مشتمل بر یک خوشه است. همچنین برای اینکه علاوه بر یک سلسله‌مراتب، بخشی از قطعات برنامه نیز حاصل گردد، می‌توان سلسله‌مراتب را در یک سطح مناسب شکست و تنها به خوشه‌های سطح بالاتر توجه کرد. انتخاب سطح مناسب شکست می‌تواند براساس معیارهای کیفیت ویژه‌ای باشد. معیارهای ممکن آن‌هایی هستند که کیفیت یک خوشه‌بندی داده شده را عنوان می‌کنند. البته کیفیت یک خوشه‌بندی به اندازه‌ی خوب برآورد می‌شود که درک برنامه‌کاربردی تحت وب را میسر می‌سازد.

---

[1] Built client page

1. **begin** with *n* clusters each containing one WA component;
2. **define** the w_L, w_{RL} and w_{SL} values;
3. **for each** cluster containing a *built client page* component, apply rule R1;
4. **while** (there is at least a pair of connected clusters)
   do
   for each cluster containing a *client page with frame* component, apply rule R2;
   for each cluster containing a *client module* or an included *server page* component, apply rule R3;
   for each cluster *c*, and for each *x*, compute $w_x^{OUT}(c)$ and $w_x^{IN}(c)$;
   for each pair of clusters A and B, compute the $C_{A,B}$ coupling between them;
   apply rule R4;
   od

شکل ۲-۱۴ الگوریتم خوشه‌بندی ارائه شده برای رویکرد اول مبتنی بر خوشه‌بندی

در مرجع مذکور آمده است که یک خوشه‌بندی خوب، دارای خصیصه‌ی اتصالات داخلی قوی و اتصالات خارجی ضعیف است. اتصالات داخلی قوی بیانگر درجه‌ی انسجام بین موجودیت‌های یک برنامه کاربردی تحت وب است و اتصال خارجی، معیار پیوستگی بین موجودیت‌های یک برنامه کاربردی تحت وب را تفسیر می‌کند. بنابراین ما معیاری با نام *کیفیت یک خوشه‌بندی* یا QoC معرفی خواهیم کرد که می‌توان آن را به عنوان تفاوت بین اتصال داخلی و اتصال خارجی یک خوشه‌بندی ابراز نمود:

$$QoC = IntraConnectivity - InterConnectivity$$

که IntraConnectivity و InterConnectivity مانند آنچه در پایین آمده است محاسبه می‌شوند:

$$IntraConnectivity = \frac{1}{NC} \sum_{\substack{k \in CLUSTER \\ Card(k)>1}} \frac{\sum_{i,j \in CLUSTER_k} p^0_{i \to j} \cdot EOUT^0_i}{Card(k) \cdot (Card(k)-1)}$$

$$InterConnectivity = \begin{cases} \dfrac{1}{\dfrac{NC \cdot (NC-1)}{2}} \displaystyle\sum_{\substack{h,k \in CLUSTER \\ h \neq h}} \dfrac{\displaystyle\sum_{i \in CLUSTER_h} \sum_{j \in CLUSTER_k} p^0_{i \to j} \cdot EOUT^0_i}{2 \cdot Card(h) \cdot Card(k)} & \text{if } NC > 1 \\ 0 & \text{if } NC = 0,1 \end{cases}$$

شکل ۲-۱۵محاسبه‌ی IntraConnectivity و InterConnectivity

در عبارت شکل فوق، NC تعداد خوشه‌ها در پیکربندی خوشه‌های کنونی است، $P^0_{i\to j}$ همان $P_{i\to j}$ بین دو گره‌ی دلخواه i و jدر گراف پیوستگی اصلی برنامه‌ی کاربردی تحت وب، $CLUSTER_k$، Kامین خوشه در خوشه‌بندی مورد نظر، $EOUT_i^0$ تعداد گره‌های متمایز به‌طور مستقیم قابل دسترسی از گره‌ی iام در گراف پیوستگی اصلی برنامه‌ی کاربردی تحت وب و Card(x) کاردینالیتی خوشه‌ی xام از خوشه‌بندی مورد نظر است.

IntraConnectivity یک معیار وزن‌دار برای لبه‌های داخلی خوشه است. مقدار آن بین ۰ (زمانی که هیچ خوشه‌ای لبه‌های داخلی نداشته باشد) و ۱ (زمانی گره‌های داخلی خوشه‌ها به‌طور کامل به هم پیوسته‌اند) متغییر است. *InterConnectivity* معیار وزن‌داری برای لبه‌های بین خوشه‌هاست. مقدار آن بین ۰ (خوشه‌های توسط لبه‌ها به هم پیوسته نیستند) و ۱ (هر گره در خوشه به سایر گره‌های در خوشه‌های دیگر پیوسته است) متغییر است.

QoC برای یک خوشه‌بندی، مقادیری بین ۱- و ۱+ اختیار می‌کند. مقدار کمینه با خوشه‌بندی دارای یک گره در هر خوشه و متصل به تمام گره‌های دیگر، حاصل می‌شود. در عوض مقدار بیشینه با خوشه‌بندی دارای تنها یک خوشه با اتصالات کامل داخلی یا با خوشه‌بندی دارای خوشه‌های مجزای کاملاً پیوسته از داخل حاصل می‌گردد.

خوشه‌بندی حاصل شده در هر تکرار الگوریتم مذکور توسط مقداری داده شده، توسط یک مقدار داده شده‌ی معیار کیفیتQoC متمایز می‌گردد. خوشه‌بندی نمایانگر بیشترین مقدار QoC، گزینه‌ای برای پیاده‌سازی بهترین بخش از قطعات یک برنامه‌کاربردی تحت وب است. بنابراین ممکن است که سلسله‌مراتب را در سطح برش متناسب با بیشینه مقدار QoC شکست. خوشه‌بندی‌های براساس این پیکربندی با هدف انتساب یک شرح بامعنا برای هر کدام، جهت فرایند انتساب مفهوم انتخاب می‌شوند.

به این ترتیب مسأله‌ی شناخت یک برنامه‌ی کاربردی تحت وب، را می‌توان مانند رویکرد ساخت‌یافته‌ی زیر تعریف کرد:

۱. مهندسی معکوس برنامه‌ی کاربردی مبتنی بر وب و تولید گراف ارتباطیWAG
۲. انجام خوشه‌بندی براساس الگوریتم داده شده
۳. یافتن خوشه‌بندی C_{max} با بیشترین مقدار QoC
۴. انجام یک فرآیند انتساب مفهوم بر خوشه‌بندی C_{max}

البته فرایند انتساب معنا نیاز دارد که هر مرجع اطلاعات مرتبط با قطعات خوشه‌های مورد توجه، تحلیل شوند. علاوه بر آن عناصر داخلی یک قطعه (یعنی قطعات صفحات داخلی) و کدمبنای آن‌ها در طی فرایند، در نظر گرفته خواهند شد.

۲-۴-۲-۵- روش دوم

در این راهکار یک روش خوشه‌بندی صفحات وب بر اساس استخراج خودکار کلمات کلیدی از آن‌ها مورد بررسی قرار خواهد گرفت[۱۷]. کلمات کلیدی رایج جهت تصمیم‌گیری درباره‌ی زمان (نحوه‌ی) گروه‌بندی صفحات مورد استفاده قرار می‌گیرد. استفاده‌ی ثانویه‌ی کلمات کلیدی برچسب‌گذاری خودکار خوشه‌های بازیابی شده از صفحات وب است.

تولیدکننده‌ی وب می‌تواند پس از محاسبه خوشه‌بندی برای یک وب‌سایت، جهت درک سازمان کل سیستم، از آن بهره ببرد. اما جهت معنادار بودن، خوشه‌ها باید به درستی برچسب‌گذاری گردند. یک نمای سطح بالا که تنها نمایشگر جعبه‌های سیاهی (خوشه‌ها) است که به هم متصل شده‌اند، یا از سوی دیگر برچسب گذاری با نام تمام صفحاتی که مورد استفاده قرار گرفته‌اند، به هیچ وجه مفید نخواهد بود. از طرف دیگر یک فرآیند دستی برچسب‌گذاری فرم دشوار دیگر برای خوشه‌بندی در تحول وب‌سایت‌ها به شمار می‌رود.

در این‌جا به محتوی صفحات به عنوان موارد اصلی که برای خوشه‌بندی سایت‌ها مورد استفاده قرار می‌گیرد، توجه شده است. چکیده‌ی اطلاعات مربوط به صفحه، توسط استخراج کلمات‌کلیدی حاصل می‌شود. همان فن برای به‌کارگیری در حل مسأله‌ی برچسب‌گذاری خوشه‌ای نیز به‌کار خواهد رفت. کلمه‌کلیدی با بیشترین رتبه در هر خوشه به عنوان برچسب آن مورد استفاده خواهد گرفت. نتایج آزمایشی اولیه امکان‌پذیری این رویکرد را تایید می‌کنند.

رویکرد این مقاله یک بهبود اصلی نسبت به فنی که در بخش پیشین، مرجع [۱۰]، آمده است دارد: برچسب‌گذاری خودکار خوشه‌ها، و از یک جنبه کاملاً متمایز است: مورد اصلی که برای خوشه‌بندی مورد استفاده قرار می‌گیرد، همان محتوی صفحات است که در آن مرجع، ارتباط صفحات بود. در واقع محتوا نقش بسیار مهمی در برنامه‌های کاربردی تحت وب ایفا می‌کند و تصور می‌شود که نقطه‌ی شروع مهمی برای خوشه‌بندی است. از سوی دیگر مشکل اتصال برنامه‌های کاربردی تحت وب از یک مسأله ناشی

می‌شود، که توسط نویسندگان مرجع [۱۰] هم بدان اشاره شده است: پیوندهای ارجاعی محض[1] (مانند آن‌هایی که به صفحه‌ی اول ارجاع می‌دهند) و به سختی از آن‌هایی که مفهوماً غنی‌ترانـد، تشخیص داده می‌شوند. برچسب‌ها در مرجع [۱۰] (که مفهوم نامیده می‌شوند) به‌طور دستی به خوشه‌ها منتسب می‌شوند. در رویکرد کنونی انتساب برچسب‌ها به روشی کاملاً خودکار انجام می‌گیرد.

۲-۴-۲-۶- خوشه‌بندی

در این مقاله [۱۷] از الگوریتم خوشه‌بندی تجمعی سلسله‌مراتبی استفاده شده است. این الگوریتم با شروع از کف سلسله‌مراتب، سلسله‌مراتب خوشه‌بندی را ایجاد می‌نماید، یعنی در ابتدا هر موجودیت در یک خوشه‌ی مجزا قرار می‌گیرد. در هر مرحله، دو خوشه که بیشترین شباهت را با هم دارند، به هم متصل می‌شوند. پس از N-1 مرحله (که دارای N موجودیت است)، تمام موجودیت‌ها در یک خوشه قرار می‌گیرند. هر سطح در سلسله‌مراتب یک بخش‌بندی (خوشه‌بندی) از خوشه‌ها را تعیین می‌کند. برای انتخاب خوشه‌بندی حاصل، باید یک *نقطه‌ی برش* در نظر گرفته شود. در این مرجع الگوریتم سلسله‌مراتبی تجمعی به الگوریتم جانسون تطبیق داده شده است تا در جهت هدف مؤلفین عمل کند:

1. begin with N clusters, each containing one Web page (N is the number of Web pages in the Web site), and compute distance between pages (using the complement of the similarity measure).
2. **while** there are more than 1 cluster **do**
   (a) find the pair of clusters at least distance;
   (b) merge these clusters into a new cluster;
   (c) update the distance measures between each pair of clusters.

end while

شکل ۲-۱۶ الگوریتم خوشه‌بندی در رویکرد استخراج خودکار کلمات کلیدی

برای به‌هنگام کردن معیار فاصله بین خوشه‌ها از *قاعده‌ی پیوند کامل*[2] استفاده شده است. این قاعده می‌گوید که معیار فاصله‌ی بین یک خوشه‌ی C هم‌اکنون موجود و یک خوشه‌ی جدید D که از اتصال دو

[1] Purely navigational links
[2] Complete Linkage Rule

خوشه‌ی A و B تشکیل شده است، کمینه‌ی فاصله‌ی *dist(C,A)* بین و *dist(C,B)*. این شیوه انسجام را بر پیوستگی، تقدم می‌دهد.

۲-۴-۲-۷- استخراج کلمات کلیدی

برای دستیابی به هدف خوشه‌بندی صفحات وب مشابه، لازم است محتوی یک سند را با روشی ساده و از لحاظ کامپیوتری قابل ردگیری، متمایز نمود. همچنین نمایش محتویات متن بایستی برای محاسبه‌ی کارای معیار تشابه محتوایی بین متون مناسب باشد. لیستی از مترتبط‌ترین کلمات کلیدی در سند را می‌توان به این منظور مورد استفاده قرار داد.

ساده‌ترین روش استخراج کلمات کلیدی یک متن براساس یافتن پراستفاده‌ترین کلمات در متن است. شهودی‌ترین دفاع از این رویکرد آن است که مهمترین مفاهیم در متن معمولاً بسیار مورد استفاده قرار می‌گیرند، یا دست‌کم مفاهیم کم‌اهمیت‌تر، کمتر مورد استفاده قرار می‌گیرند. حتی اگر این مورد را به عنوان یک شهود قبول داشته باشیم، شمارش ساده‌ی پراستفاده‌ترین کلمات در یک متن برای برآورده شدن هدف ما کافی نیست. رویکرد پایه لازم است که به روش‌هایی اصلاح گردد، که در ادامه به آن‌ها پرداخته خواهد شد.

ابتدا باید به واحدهای لغوی که فراتر از کلمات ساده هستند بپردازیم. نه تنها یک مفهوم را می‌توان (در صورتی‌که یک عبارت[1] باشد) با اهمیتی بالاتر از یک کلمه مورد ارجاع قرار داد، بلکه اغلب مفهومی که یک متن را دقیقاً متمایز می‌کند، از ویژه‌ترین مفاهیم آن است که معمولاً با عبارت‌های پیچیده بیان می‌شود. بدین منظور به عنوان اولین گام در این راه، کلمات تکی و بیگرم‌ها[2] که دنباله‌ی پیوسته‌ی کلمات هستند، را به عنوان واحدها مورد استفاده به حساب آورده‌ایم. به هر یک از این دو واحد (bigram یا mono) با عنوان ترم ارجاع خواهیم داد.

دوم این‌که یک ترم می‌تواند مکرر در یک متن مورد استفاده قرار گیرد بدون آن‌که متن را در مقایسه با سایر متون متمایز سازد. این امر زمانی اتفاق می‌افتد که تمام متنی که ما سعی در خوشه‌بندی آن داریم، درباره‌ی یک موضوع فراگیر است. برای مثال اگر تمام متنی که مورد پردازش قرار می‌دهیم، به پورتال شرکت مخابرات تعلق داشته باشد، ممکن است ترم‌هایی مانند "خط تلفن" یا حتی "تلفن" را برای مشخص

[1] Phrase
[2] Bigrams

نمودن سند مناسب ندانیم، حتی اگر آن‌ها به صورت مکرر مورد استفاده قرار گیرند. برای اجتناب از چنین مشکلی تکرار یک ترم در یک سند باید با میانگین تکرار آن ترم در تمام وب‌سایت، برابر نباشد.

۲-٤-۲-۷-۱- انتخاب بیگرم‌ها

بر اساس هدف تعیین شده، همه‌ی بیگرم‌های متن به عنوان ترم به حساب نمی‌آید. ما بیشتر مایل به سه کلاس از ترم‌های پیچیده هستیم: موجودیت‌های نامدار، واحدهای لغوی پیچیده و عبارات آزاد تکراری[1]. موجودیت‌های نامدار نام‌های ارجاعی به افراد، سازمان‌ها و تاریخ‌ها هستند. واحدهای لغوی پیچیده عبارت‌های چندکلمه‌ای هستند، که در فرهنگ‌های لغت به سادگی یافت می‌شوند (مانند کلمات فعلی از قبیل "Put on"و عبارات اصطلاحی از قبیل "roller coaster"). در نهایت واژه‌ی عبارات آزاد تکراری برای ارجاع به ترکیبی آزاد از کلمات که به‌طور متناوب در حال استفاده برای انتساب به یک مفهوم هستند، مورد استفاده قرار می‌گیرد. این عبارات توسط استفاده‌ی بالا در متون مرجع (از قبیل " American Government")، درجه‌ی بالای پیوستگی[2] بین کلمات (برای مثال "First time")، یا تمایز بالا (مانند "international summit")، مشخص می‌شوند.

انتخاب بیگرم‌های متعلق به هر یک از این سه کلاس، یک امر پرمشغله و نیازمند به منابع است. البته این امر می‌تواند توسط بازیابی، به ترکیبی از یک مقیاس آماری ساده‌ی و یک دانش زبانی مقدماتی، تقریب زده شود. رویکرد پیشنهادی مبتنی بر انتخاب بیگرم‌ها به عنوان کلیدهای کاندید است که بیگرام‌های متداول ایتالیایی در مقاله‌ی کنونی مورد استفاده قرار گرفته‌اند. در این مقاله لیست بیگرام‌های متداول ایتالیایی با رویکرد زیر ایجاد شده است:

۱. انتخاب پرکاربردترین بیگرام‌ها در یک مرجع ۳۲ میلیون کلمه‌ای در یک روزنامه‌ی ایتالیایی
۲. حذف تمام بیگرام‌هایی که تعداد تکرارشان در مرجع کمتر از ۴ بار بوده است
۳. به‌کارگیری یک فیلتر برمبنای کلمات توقف

هدف مرحله‌ی فیلترینگ حذف بیگرام‌هایی مانند "with the" یا "the only" است که ممکن است مکرراً در متون ظاهر شوند، اما متعلق به هیچ‌یک از کلاس‌های ترم نشان داده شده در بالا نباشند. علاوه بر آن در این مرحله به سادگی تمام بیگرم‌هایی را که دست‌کم شامل یکی از کلمات موجود در لیست

[1] Recurrent free phrases
[2] Association

کلمات توقف است، حذف نموده‌ایم. کلمات توقف در حذف بیگرم‌های پراستفاده اما بی‌ربط بسیار مفیداند تاجاییکه کامپایل دستی لیستی از آن‌ها مناسب به نظر می‌رسد.

۲-٤-۲-۷-۲- بسامد[1] معکوس سند

جهت تمایز بهتر هر صفحه‌ی وب که بایستی خوشه‌بندی شود، برای کلمات کلیدی رتبه تعیین خواهد شد. یک رویکرد ساده می‌تواند شمارش تعداد تکرار ترم در صفحه باشد (فرکانس آن). اما این امر برای کلماتی که به‌طور یکنواخت در صفحات تکرار می‌شوند مناسب نیست، زیرا اسناد نسبتاً غیر مرتبط را هم در کنار هم قرار می‌دهد. به این ترتیب می‌توان کلمات کلیدی را بر اساس بسامد معکوس سند که در زیر تعریف شده است، رتبه‌بندی نمود:

$$W[k] = N[k] \log \frac{D}{d[k]}$$

که N[k] تعداد مطلق تکرار Kامین کلمه‌کلیدی، d[k] تعداد اسناد شامل آن کلمه‌ی کلیدی و D تعداد کل اسناد است. کلمات پراستفاده دارای بسامد معکوس سند بالاتری هستند (d[k] در مقایسه با D کوچک است). در عوض به کلمات کلیدی یکنواخت ارزش کمی معادل می‌گردد (مقدار $\log(D/d[k])$ در این حالت به سمت صفر میل می‌کند).

۲-۴-۲-۸- خوشه‌بندی و برچسب‌گذاری

فرض کنید KW بردار تمام کلمات کلیدی تعیین شده برای کل وب‌سایت (اتحاد کلمات کلیدی تعیین شده برای هر صفحه) باشد که هر کلمه‌ی کلیدی توسط یک مدخل یکتا نمایش داده شده باشد. یک بردار دیگر برای هر صفحه‌ی p می‌توان ساخت که N[k] تعداد وقوع kامین کلمه‌ی کلیدی KW[k] در p با اندیس k است. در زمانی که کلمه‌ی کلیدی در صفحه‌ی p نباشد، به‌ازای مدخل آن عدد صفر وارد خواهد شد. همچنین تعداد مطلق وقوع N[k] قابل جایگزینی با بسامد معکوس سند W[k] است.

با داشتن توصیف هر صفحه در قالب یک بردار، امکان به‌کارگیری تشابه یا معیار فاصله برای خوشه نمودن موجودیت‌ها در خوشه‌بندی وجود خواهد داشت. تشابه\فاصله بین خوشه‌ها با استفاده از *قاعده‌ی*

[1] Frequency

پیوند کامل از تشابه\فاصله بین موجودیت‌ها تعمیم یافته است. در این مقاله معیار تشابه بر معیار فاصله ترجیح داده شده است، زیرا دومی در معرض مشکلات شناخته شده‌ای مانند بردارهای تنک یا خالی قرار دارد: بر اثر کم شدن فاصله، مشکل شکل‌گیری خوشه‌های نامناسب بروز خواهد کرد. معیار تشابه با توجه به توصیفات بالا، از ضرب بردار نرمالیزه شده زیر حاصل می‌گردد:

$$sim(p,q) = \frac{< V_p, V_q >}{||V_p|| \quad ||V_q||}$$

که در آن V_p و V_q بردارهای تصویر از صفحات p و q هستند و علامت‌های < و > نشان‌دهنده‌ی ضرب برداری آن‌هاست، که به وسیله‌ی حاصلضرب نرم‌ها، نرمالیزه می‌شوند. همچنین محرز است که معیار تشابه محاسبه شده از طریق فوق بین ۰ و ۱ خواهد بود.

پس از اجرای الگوریتم تجمعی خوشه‌بندی فوق، یک نقطه‌ی برش مناسب به شکل دستی انتخاب می‌گردد. سهولت انتخاب میزان تجرید مناسب برای کاربر و سپس تطابق آن به طرف بالا یا کف سلسله‌مراتب یک امکان محاوره‌ای مهم به شمار می‌رود. در واقع انتخاب سطح تجرید مناسب برای ایجاد بهترین درک از برنامه از آغاز معین شده نیست و می‌توان با آزمون و خطا و حرکت در سلسله‌مراتب، مشخص گردد.

در نهایت برچسب‌ها، به صورت خودکار به هر خوشه منتسب می‌شوند. کلمه‌ی کلیدی با بیشترین وقوع (بیشترین بسامد معکوس سند) در خوشه‌ی C، به آن به عنوان برچسب منتسب خواهد شد. لازم به توجه است، در حالی که تعداد کلی وقوع‌های کلمه‌ی کلیدی KW[k] در یک خوشه‌ی C، جمع تعداد وقوع‌های صفحات دربرگرفته شده است، اما بسامد معکوس سند از به کارگیری معادله‌ی مرتبط عرضه شده بر تمام خوشه‌ها حاصل می‌شود; D تعداد خوشه‌ها به حساب می‌آید و d[k] تعداد خوشه‌هایی است که مشتمل بر KW[k] هستند.

۲-۵- نتیجه‌گیری

در رابطه با هدف پایان‌نامه که طراحی و پیاده‌سازی محیطی برای مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب است، این فصل مقدمه‌ای بر مهندسی معکوس نرم‌افزار پیش از پرداختن به مهندسی معکوس برنامه‌های کاربردی تحت وب را عرضه نمود و مفهوم کلی مهندسی معکوس نرم‌افزار، که نرم‌افزارهای کاربردی تحت‌وب نیز بخشی از آن است، مرور گردید. ابتدا اهداف آن بیان شده تا در آینده بتوان میزان موفقیت روش‌های جدید را با کمک آن درک کرد و سپس مشکلات مهندسی معکوس تحت عنوان نگاشت بین عرصه‌های مختلف پیاده‌سازی و سطوح طراحی و مشکلات ذاتی آن عنوان گردید. پس از آن به بیان ماهیت مهندسی معکوس نرم‌افزار و سطوح مختلف آن پرداخته شد و این امر با استفاده از نمودارهای متنوع تفهیم گردید.

همچنین پس از ارائه‌ی یک تعریف برای برنامه‌های کاربردی مبتنی بر وب و تشریح علت روی‌آوردن تولیدکنندگان نرم‌افزار به این نوع برنامه‌ها، یک دسته‌بندی برای انواع مختلف آن، همراه با مثال عرضه گردید. پس از قطعات برنامه‌های کاربردی مبتنی بر وب که در منبع [۵] یک دسته‌بندی مناسب از آن‌ها صورت گرفته است، نام برده شد و از وجوه تمایز برنامه‌های کاربردی مبتنی بر وب و برنامه‌های کاربردی عادی پرده برداشته شد. همچنین از مقایسه‌های صورت پذیرفته، این نتیجه به دست آمد که به خاطر مشکلات و محدودیت‌های تولید نرم‌افزارهای کاربردی تحت وب کنونی، نگه‌داری آن‌ها درحال حاضر امر ساده‌ایی نیست[۲۰].

در ادامه با توجه به تعدد رویکردهای ارائه شده برای مهندسی معکوس برنامه‌های کاربردی تحت وب، ابتدا برای یافتن نوعی دسته‌بندی آن‌ها تلاش نمودیم تا بتوان با دیدن هر رویکردی آن را در گروه مشخص خود قرار داد و در هر دسته به مقالات متنوعی پرداخته شد. به این ترتیب گرچه تعداد کمی از مقالات ارائه شده در زمینه‌ی مهندسی معکوس بررسی شده است، اما سعی در انتخاب مقالاتی بوده است که جنبه‌های مختلف دسته‌ها را پوشش دهند و حالت جهت‌دهی و راه‌گشایی داشته باشند.

آنچه در بررسی رویکردها قابل توجه است، این است که تقریباً تمام روش‌ها ابتدا به ارائه‌ی مدلی از برنامه‌ی کاربردی مورد بررسی پرداخته‌اند و مدل‌های ارائه شده بیشتر به صورت نمایش مبتنی بر گراف است که در آن، گره‌ها، قطعات نرم‌افزاری هستند که مورد بررسی قرار خواهند گرفت و یال‌ها ارتباطات بین آن‌ها هستند.به عنوان مثال در ۲-۴-۱-۲ از الحاق جی کنالن[1] بر UML، برای مدل‌سازی و تولید برنامه‌های کاربردی تحت وب [۱۳,۳۹]، استفاده شده است، یا در روش بررسی شده در قسمت۲-٤-۲-۱

[1] J. Conallen

گره‌های گراف را صفحات برنامه‌کاربردی مبتنی بر وب تشکیل می‌دهند و یال‌های آن را ارتباطات مختلف بین صفحات. این مطلب درباره‌ی برنامه‌های کاربردی غیر مبتنی بر وب نیز صادق است، به عنوان مثال در برنامه‌های کاربردی متعددی از مؤلفه‌های[1] نرم‌افزاری (مانند فایل‌ها، کلاس‌ها و بسته‌ها[2]) به عنوان گزینه‌ای برای گره‌ها استفاده شده و MDF[3] براساس آن تشکیل گردیده است[۲۳]، [۲۴]، [۲۵] و ....

دقت در این نحوه‌ی مدل‌سازی برنامه، برای شروع عملیات مهندسی معکوس، این ایده را در ذهن متبلور می‌سازد که می‌توان رویکردی عمومی برای مهندسی معکوس نرم‌افزارهای کاربردی مبتنی بر وب ارائه داد که بر ساختارهای گرافی عمل نماید و به عبارت بهتر توانایی انجام عملیات مهندسی معکوس را بر هر ساختار گرافی داشته باشد و صورت مسأله را از مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب به انجام عملیات خوشه‌بندی بر یک گراف تبدیل نماید.

نکته‌ی مهم دیگر عدم وجود رویکردهایی است که بتوان آن را به طیف وسیعی از برنامه‌های کاربردی مبتنی بر وب اعمال نمود و این مشکل هم از ماهیت متفاوت زبان‌های برنامه‌نویسی نرم‌افزارهای کاربردی مبتنی بر وب سرچشمه می‌گیرد و هم از گوناگونی آن‌ها. که این موضوعات را می‌توان در راستای ارائه‌ی رویکردی که در این پایان نامه ارائه خواهد شد، مورد توجه قرار داد. قابل توجه است که هیچ‌یک از منابع بررسی شده، ادعای عمومی بودن رویکرد خود در سطح وسیعی را نکرده است.

همچنین ماهیت سلسله‌مراتبی برنامه‌های کاربردی مبتنی بر وب که به وضوح در روش‌های بررسی شده به چشم می‌خورد، آن‌ها را برای اعمال الگوریتم‌های مختلف خوشه‌بندی سلسله‌مراتبی مستعد ساخته است و به نظر می‌رسد می‌توان از خوشه‌بندی، در مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب استفاده‌های بیشتری نمود.

در روش‌های مبتنی بر خوشه‌بندی آن‌چه مسلم است، این است که پیش از اقدام به خوشه‌بندی باید با پیش‌فرض‌های اعمال آن آشنا بود، که انتخاب مدلی که قطعات برنامه‌های کاربردی تحت‌وب را به‌نحوکافی[4] توصیف کند، تعریف معیار تعیین کننده‌ی زمان خوشه‌بندی دوقطعه به یک واحد منسجم و تعریف الگوریتم خوشه‌بندیی که باید به کار گرفته شوند، از جمله موارد مهم در این زمینه‌اند که باید مشخص شوند. الگوریتم‌های خوشه‌بندی در این زمینه معمولاً به سه دسته تقسیم می‌شوند که عبارتند از: الگوریتم‌های مبتنی بر ساختار، اتصال و کلمات کلیدی.

---

[1] Modules
[2] Packages
[3] Module Dependency Graph
[4] Adequately

فصل سوم

توزیع بارکاری کارگزار وب

۳-۱- مقدمه

توزیع بارکاری[1] در حالت کلی به پخش وظیفه‌ی یک ماشین بین چند ماشین، برای جلوگیری از اشباع ظرفیت خدمت‌دهی آن، اشاره دارد. این واژه اصطلاحی عام است که ممکن است در زمینه‌های مختلفی مطرح شود. برخی از این زمینه‌ها عبارتند از: توزیع داده در بین حافظه‌های مختلف، توزیع فایل‌ها در دیسک‌ها، توزیع وظیفه‌ها[2] بین پردازنده‌ها، توزیع بسته بین رابط‌های شبکه و توزیع درخواست‌ها بین کارگزارها. منظور از توزیع بارکاری در اینجا همان توزیع درخواست‌ها بین کارگزاران وب است.

در این فصل ابتدا به اهمیت توزیع بارکاری کارگزاران وب پرداخته خواهد شد. سپس انواع توزیع بار با یک دید اجمالی مورد بررسی قرار خواهد گرفت و نحوه و شرایط استفاده از هر یک تشریح می‌گردد. در نهایت مشکلاتی که در زمینه‌ی توزیع نرم‌افزاری بار کاری ممکن است بروز کند یادآوری می‌گردد و راه‌کارهایی که تاکنون برای آن‌ها ارائه شده است نیز معرفی می‌شود.

---

[1] Load Balancing
[2] Task

۳-۲- چرا توزیع بارکاری کارگزارها لازم است؟

اگر تنها یک کارگزار وب برای پاسخ‌گویی به تمام درخواست‌های HTTP یک برنامه‌کاربردی مبتنی بر وب موجود باشد، پس از معروف شدن وب‌سایت، کارگزار وب ممکن است دیگر توانایی رسیدگی به حجم‌های بالای ترافیک ورودی را نداشته باشد. صفحات وب ممکن است به کندی بارگذاری گردند و کاربران مجبور شوند برای خلوت شدن بارکاری کارگزار منتظر بمانند. افزایش ترافیک و ارتباطات به وب‌سایت ممکن است به حدی برسد که ارتقای سخت‌افزاری کارگزار نیز پاسخ‌گو نباشد.

برای دستیابی به قابلیت توسعه‌ی کارگزارن وب، لازم است تعداد بیشتری کارگزار اضافه شود تا بارکاری را بین گروهی از کارگزاران توزیع نمود، که این گروه با عنوان *خوشه‌ی کارگزاران*[1] نیز نام برده می‌شود. توزیع بار بین این کارگزاران به *توزیع بارکاری*[2] معروف است. توزیع بارکاری به همه‌ی انواع کارگزارها، اعم از کارگزار برنامه‌کاربردی یا کارگزار پایگاه‌داده، قابل اعمال است. اما در این فصل تنها به توزیع بارکاری کارگزاران وب پرداخته خواهد شد.

۳-۳- روش‌های توزیع بارکاری

یک سیستم‌کارگزار وب توزیع شده، مشتمل بر چندین کارگزار وب یا یک طرح توزیع درخواست‌های مشتریان بین آن‌هاست. معماری‌های متعدد مبتنی بر خوشه برای این‌کار پیشنهاد و پیاده‌سازی شده‌اند. که در زیر به آن‌ها اشاره می‌شود[۳۰]:

۳-۳-۱- توزیع در جانب کاربر

یکی از مزایای این رویکرد آن است که لازم نیست تغییری در جانب کارگزار ایجاد گردد. این امر با افزودن برخی قطعات[3] به نرم‌افزار جانب مشتری (از قبیل مرورگر وب) انجام می‌شود. یکی از مثال‌های جانب در این زمینه مرورگر نت‌اسکیپ[4] است. وب‌سایت نت‌اسکیپ توسط بیش از ۱۰۰ کارگزار وب حمایت می‌شود. هر زمان مرورگری سایت نت‌اسکیپ را درخواست کند، آن مرورگر به صورت تصادفی یکی از آن کارگزارهای را انتخاب می‌کند. این امر در واقع به نوعی توزیع بارکاری را نیز شامل می‌شود.

---

[1] Server Cluster
[2] Load Balancing
[3] Component
[4] Netscape

اما هر وب‌سایتی امکان و اجازه‌ی تغییر نرم‌افزار مشتری را ندارد. علاوه بر آن ممکن است برخی تغییرات درجانب مشتری ضروری باشد و این در حالی است که مشتری از تغییرات پویای جانب کارگزار نیز خبر ندارد. بدین ترتیب معماری مبتنی بر مشتری جذابیت زیادی ندارد.

۳-۳-۲- توزیع در کارگزار DNS

در حالت ایده‌آل تمام درخواست‌های کاربران بایستی فرایند تبدیل آدرس را پیش از ارسال به کارگزار مقصد طی کند. بنابراین کارگزار DNS گزینه‌ای مناسب برای توزیع درخواست‌ها به شمار می‌آید. زیرا کارگزار DNS می‌تواند کارگزاری مناسب انتخاب نماید و آدرس IP آن را به مشتری برگرداند. بدین ترتیب برخی کارگزاران پیشین از این رویکرد استفاده می‌کردند.

متأسفانه باتوجه به سطوح مختلف نهان‌نگاری[1]، بیشتر نگاشت‌های نام به آدرس IP در مرورگرهای مشتریان، کارگزاران حایل[2] و کارگزاران DNS میانی نهان‌نگاری می‌شوند. بنابراین DNS اصلی اغلب اوقات مورد دسترسی قرار نخواهد گرفت. این امر از دسترسی مشتریان به آخرین اطلاعات مرتبط با کارگزار جلوگیری می‌کند. به این ترتیب کاربران در خلال زمان اعتبار نهان‌نگاری، به همان کارگزار دسترسی خواهند داشت. نتایج شبیه‌سازی نشان‌دهنده‌ی آن است که میانگین زمان اعتبار نهان‌نگاری یک ساعت می‌تواند به میزان ۴۰ درصد در کل بارکاری خوشه نوسان ایجاد نماید[۳۰].

علاوه بر این مشکلات عمده‌ی دیگری در رابطه با این رویکرد مطرح هستند[۳۲] که از آن جمله می‌توان به عدم حمایت مناسب از دسترس‌پذیری کارگزارها اشاره کرد، یعنی اینکه با توجه به توزیع معمولاً نوبت گردشی[3] درخواست‌ها، اگر یکی از کارگزاران نهایی از کار بیافتد هنوز هم درصدی از درخواست‌ها به آن ارجاع داده می‌شود. پس لازم است در بازه‌های زمانی مکرر، وجود تمام کارگزاران نهایی وارسی گردد. همچنین الگوریتم توزیع بار به شیوه‌ی نوبت گردشی ممکن است، عادلانه نباشد، زیرا تضمینی در طول برابر از لحاظ زمانی برای جلسات کاربران با کارگزاران وجود ندارد و چون در این رویکرد به بارکاری کنونی کارگزار توجهی نمی‌شود، ممکن است بارکاری برخی کارگزارها نسبت به سایرین نوسان زیادی داشته باشد.

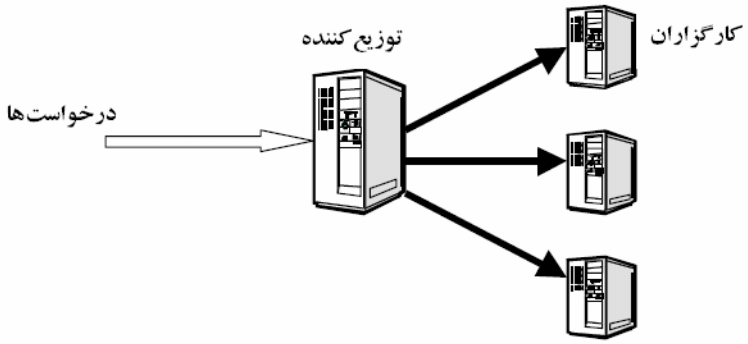
---

[1] Caching
[2] Proxy Servers
[3] Round robin

۳-۳-۳- توزیع توسط توزیع‌کننده[1]

معماری مبتنی بر توزیع‌کننده، یکی از رایج‌ترین انتخاب‌ها برای کارگزاران دارای چند خوشه به شمار می‌رود. این معماری از یک موجودیت مرکزی به نام توزیع‌کننده که مسوول توزیع تمام درخواست‌های مشتریان به کارگزاران نهایی[2] است. تصویری از این معماری در ادامه آمده است.

اکنون موارد علمی و صنعتی زیادی برای این معماری وجود دارد. پیاده‌سازی‌های خوشه‌های وب این نوع معماری را می‌توان بر اساس فنون شبکه که توزیع‌کننده از آن برای توزیع درخواست کاربران استفاده می‌کند، طبقه‌بندی کرد:



شکل ۳-۱ نمای سطح بالا از خوشه‌های کارگزار در معماری مبتنی بر توزیع‌کننده

۴. سوییچینگ در لایه‌ی چهار به همراه هدایت[3] بسته در لایه‌ی دو (L4/2)

۵. سوییچینگ در لایه‌ی چهار به همراه هدایت بسته در لایه‌ی سه (L4/3)

۶. سوییچینگ در لایه‌ی هفت به همراه هدایت بسته در لایه‌ی دو یا سه (L7)

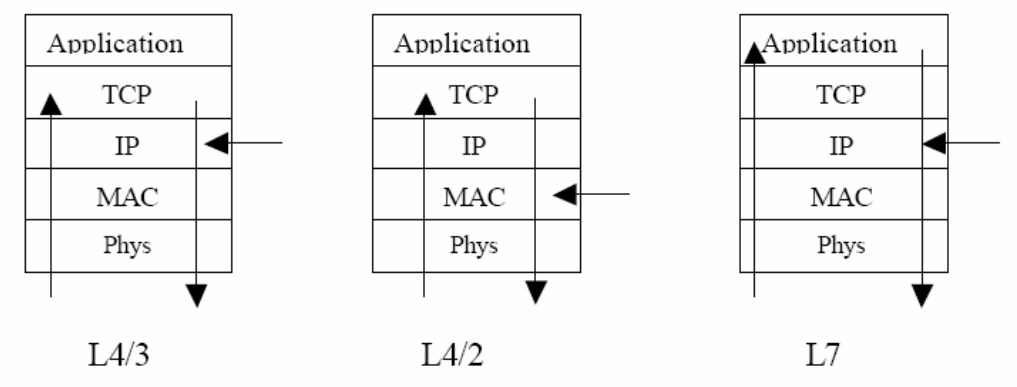
ایده‌ی نهفته در طرح طبقه‌بندی فوق آن است که بسته‌های درخواستی مشتری، پیش از ارسال به کارگزار نهایی، دست‌کم دوبار از پشته‌ی پروتکلی شبکه‌ی توزیع‌کننده رد خواهند شد (یک بار در جهت بالا و دیگر بار در جهت پایین).

- یک درخواست ابتدا از پشته‌ی پروتکلی بالا خواهد رفت. بسته دست‌کم باید به لایه‌ی TCP (لایه‌ی چهار) برسد، زیرا توزیع کننده به وارسی اطلاعات جلسه در لایه‌ی TCP،

[1] Dispatcher
[2] Backend
[3] Forwarding

جهت هدایت بسته، نیاز دارد. با توجه به فنون مختلف، بسته ممکن است به لایه‌های مختلفی بالا رود (چهار یا بالاتر). ما این فرایند را سوییچینگ در لایه‌ی x نام نهاده‌ایم.

- سپس توزیع کننده، بسته را دوباره بسته‌بندی می‌کند و آدرس کارگزار نهایی را به آن می‌افزاید و آن را به کارگزار نهایی ارسال می‌کند. آدرس افزوده شده ممکن است یک آدرس لایه‌ی دو (آدرس MAC) یا آدرس لایه‌ی ۳ (آدرس IP) باشد. بسته به نوع آدرس، این امر را هدایت در لایه‌ی x می‌نامیم.



شکل ۳-۲ طبقه‌بندی معماری‌های مبتنی بر توزیع‌کننده‌ی کارگزاران وب

در شکل فوق، پیکان‌های افقی، بیانگر لایه‌ای هستند که نگاشت مجدد در آن‌ها صورت می‌پذیرد. فنون بیان شده با جزییات بیشتر در ادامه خواهند آمد.

رویکرد L4/3:

همانطور که شکل فوق نشان می‌دهد، رویکرد L4/3 درخواست کاربر را با تغییر آدرس IP (آدرس لایه‌ی ۳) در سرآیند مربوطه، به آدرس IP کارگزار نهایی، هدایت می‌کند. در این مورد فرض بر این است که هر کارگزار دارای یک آدرس IP یکتاست.

زمانی که کارگزاران نهایی بسته‌ی پاسخ را به مشتری بر می‌گردانند، لازم است که آدرس IP مبدأ را به آدرس IP توزیع کننده تغییر دهند. این امر می‌تواند با برگرداندن بسته به توزیع‌کننده یا تغییر محلی آدرس IP صورت پذیرد. رویکرد اول بار اضافه بر توزیع‌کننده تحمیل می‌کند، زیرا تمام بسته‌های ورودی و خروجی باید از توزیع‌کننده عبور کنند. رویکرد دوم را می‌توان با استفاده از فنونی مانند IP-Tunneling یا تغییر مستقیم پیکربندی کارگزار نهایی، انجام داد.

L4/3 دارای این مزیت است که کارگزاران می‌توانند بر شبکه‌های مختلفی پخش باشند، زیرا توزیع‌کننده آن‌ها را با آدرس IP می‌شناسد. این امر زمانی که نرخ درخواست‌ها به حدی برسد که یک شبکه نتواند به تنهایی پاسخ‌گوی حجم انبوه انتقال بسته باشد، مفید است.

رویکرد L4/2:

این رویکرد، درخواست‌های کاربران را با تغییر آدرس MAC در سرآیند لایه‌ی دوی بسته، به آدرس MAC کارگزار نهایی، به کارگزاران وب نهایی ارسال می‌کند. این فن از نوشتن مجدد سرآیند لایه‌ی IP بسته‌ی درخواست اجتناب می‌کند و به کارگزاران نهایی امکان می‌دهد تا یک آدرس IP را با توزیع‌کننده به اشتراک گذارند. بسته‌های خروجی از کارگزاران نهایی باردیگر از توزیع‌کننده رد نخواهند شد، زیرا بسته‌های پاسخ، دارای همان آدرس IP هستند که توزیع‌کننده داراست.

عیب این طرح آن است که تمام کارگزاران باید در همان زیرشبکه‌ی[1] توزیع‌کننده قرار داشته باشند، زیرا یک آدرس MAC را نمی‌توان در شبکه‌های مختلف مورد استفاده قرار داد.

رویکرد L7:

برخلاف دو معماری گذشته، در این روش لازم است درخواست کاربر تمام پشته‌ی پروتکلی شبکه را طی کند، یعنی در لایه کاربرد با آن کار خواهد شد. این گذر اضافی اطلاعات مهم دیگری نیز درباره‌ی درخواست در اختیار قرار می‌دهد، یعنی اینکه در این مرحله می‌توان از محتوی درخواست کاربر نیز مطلع شد. از این امر می‌توان در زمینه‌ی یکی از فنون جذاب توزیع درخواست‌ها بر کارگزارهای وب استفاده نمود، که قطعه‌بندی داده[2] نام دارد.

به‌جای تکرار[3] یا تکثیر[4] داده برتمام کارگزارهای نهایی، که هم‌اکنون تنها راه‌کار در ایجاد رویکردهای L4/2 و L4/3 به حساب می‌آیند، فایل‌های وب‌سایت تقسیم می‌شود و بر کارگزارهای مختلفی نهاده می‌شود و توزیع کننده امکان انتقال درخواست به کارگزاری که فایل درخواستی را پردازش می‌کند، دارد.

یک مزیت این رویکرد آن است که در هنگام نیاز به تغییر محتوی یک سایت، از به‌هنگام‌سازی داده بر تمام کارگزاران انتهایی اجتناب می‌شود. اما این رویکرد سربار پردازشی اضافی به همراه دارد، زیرا لایه‌ی کاربرد خارج از هسته‌ی سیستم‌عامل است و نیز مشکل تقسیم‌مجدد پویای داده هم در اینجا مطرح می‌شود.

---

[1] Subnet
[2] Data Partition
[3] Replication
[4] Cloning

۳-۳-۱- الگوریتم‌های توزیع رویکرد مبتنی بر توزیع‌کننده

روش‌های متنوعی برای توزیع درخواست‌ها در روش مبتنی بر توزیع‌کننده وجود دارد، اما این‌که هر یک از این روش‌ها از چه الگوریتمی برای تقسیم درخواست‌ها بین کارگزارن استفاده می‌کنند، نیز باید بررسی گردد تا در هنگام ارائه‌ی رویکردی جدید، مقایسه و نتیجه‌گیری ساده‌تر شود. عمده‌ترین الگوریتم‌هایی که این نوع توزیع بارکاری از آن‌ها برای پخش درخواست‌ها استفاده می‌کند، عبارتند از[۳۴]:

- الگوریتم نوبت گردشی[1]

در این طرح، درخواست‌ها به شیوه‌ی نوبت گردشی به کارگزارهای مختلف ارسال می‌شوند. در این شیوه به بارکاری کنونی کارگزارها توجهی نمی‌شود.

- الگوریتم کمترین ارتباط[2]

درخواست بعدی به کارگزاری ارسال می‌گردد که دارای کمترین تعداد ارتباطات HTTP است. این الگوریتم، یک الگوریتم زمان‌بندی پویا به حساب می‌آید زیرا که لازم است تعداد ارتباطات باز هر کارگزار را در نظر داشته باشد. این امر را می‌توان با نگهداری یک جدول ارتباط-کارگزار بر کارگزار توزیع کننده، پیاده‌سازی نمود.

- الگوریتم کمترین بارکاری[3]

در این شیوه، کارگزار توزیع کننده، درخواست بعدی را به کارگزاری که دارای کمترین بارکاری است (بارکاری کارگزار به عنوان مجموع زمان خدمت تمام درخواست‌های واگذار شده به آن در نظر گرفته می‌شود) ارسال می‌کند. این الگوریتم لازم است اطلاعات زمان خدمت تمام درخواست‌های کاربران را داشته باشد. این اطلاعات اغلب در زمان رسیدن درخواست، نامشخص هستند و به این ترتیب دشواری‌های عمده‌ای در ارتباط با پیاده‌سازی این الگورتیم وجود دارد.

۳-۳-۴- توزیع در جانب کارگزاران

یک ویژگی قابل توجه در رویکردهای مبتنی بر DNS و توزیع‌کننده آن است که تنها یک موجودیت مرکزی، مسوول توزیع بارکاری است. رویکرد مبتنی بر کارگزار تمام کارگزاران را از طریق ارتباطات بین کارگزاری، در فرآیند توزیع بار دخالت می‌دهد.

---

[1] Round Robin
[2] Least Connection
[3] Least Loaded

یک مثال در این زمینه، دوباره‌نویسی بسته‌ی به سبک توزیع‌شده[1] (DPR) است[۳۱]. در این طرح، در ابتدا یکی از کارگزاران در خوشه که آدرس IP آن به همه اعلام شده است، برای پذیرش درخواست‌های کاربران برگزیده می‌شود. زمانی که بارکاری آن بیش از توانش گردید، کارگزار مناسبی در خوشه می‌یابد و درخواست‌ها را به آن تغییر جهت می‌دهد. همچنین جهت عادلانه بودن این امر برای تمام کارگزاران، IP آن‌ها توسط DNSشان به صورت نوبت گردشی به همه اعلام می‌شود. بنابراین تمام کارگزاران با این مکانیزم تغییر درخواست، در پردازش درخواست‌های کاربران شریک خواهند بود.

ممکن است در این رویکرد، سربار ارتباطات بین جفت کارگزارها زیاد گردد. همچنین تغییر در هسته‌ی پروتکل شبکه‌ی تمام کارگزارها نیز لازم است. بدین ترتیب نگهداری و پیاده‌سازی آن، نسبت به سایر رویکردها دشوارتر است.

[1] Distributed Packet Rewriting

۳-۴- مشکل متغییر سراسری مشترک

ماهیت توزیع بارکاری بیشتر در زمینه‌ی کارگزارانی بروز پیدا می‌کند که داده‌های بیشتر خواندنی دارند، تا نوشتنی. به عبارت بهتر به علت توزیع محتوی برنامه‌کاربردی مبتنی بر وب یا وب‌سایت، بر چند کارگزار، پویایی وب‌سایت یا برنامه‌کاربردی مشکلاتی را در زمینه‌ی هماهنگی و یکسان بودن داده به وجود می‌آورد.

اگر تغییر داده در یک کارگزار نیاز سریع به انعکاس در سطح سایر کارگزارها نداشته باشد، می‌توان یک مکانیزم تکرار[1] داده‌های مشترک پویا را در بازه‌های زمانی مشخص انجام داد، تا هماهنگی در خواندن داده‌ها به صورت یکنواخت وجود داشته باشد، اما در صورتی که داده‌ی مشترکی با میزان ارجاع بالا بین کارگزاران وجود داشته باشد، باید پیش از توزیع آن بر کارگزاران، برایش تصمیمی اتخاذ شده باشد. برای این امر دو رویکرد به صورت عملی وجود دارد:

۱. ذخیره‌ی متغییر مشترک در پایگاه داده (یا یک فایل XML)
۲. نگه‌داری متغییر مشترک در حافظه‌ی اصلی، اما با اعمال سیاست‌های لازم

همانطور که می‌دانیم، نگهداری متغییر مشترک در پایگاه داده (یا یک فایل XML) مدیریت آن را تسهیل می‌کند، اما به ازای هر کاربر (دسترسی) جدید، با ارجاع به فایل یا پایگاه داده روبرو هستیم و در صورت ارجاع‌های فراوان، کارایی پایینی در این زمینه نسبت به مورد دوم وجود دارد. همچنین ایجاد تغییرات محتمل در برنامه‌های کاربردی برای این زمینه بسیار ساده است (آدرس فایل یا پایگاه داده باید در کد برنامه عوض شود). اما مورد دوم، در مرجع [۳۳] به صورت زیر مورد بررسی قرار گرفته است:

مشکلی که در گذر از توزیع بارکاری یک کارگزار وب بین چندکارگزار، امکان وقوع دارد، به وابستگی بین صفحه‌ها ارتباط دارد. این امر از طریق تعریف متغییر سراسری در برنامه‌ی کاربردی مبتنی بر وب ایجاد می‌شود و برای زبان‌های مختلف دستورات مخصوص به خود را دارد. برای مثال در زبان ASP با استفاده از شیء Application می‌توان متغییری تعریف کرد که برای تمام جلسه‌ها یکی باشد و در هر جلسه بتوان به آن دست یافت. برای زبان JSP چند سطح تعریف متغییر وجود دارد که متغییر از سطح Application همان معنی‌ایی را دارد که برای زبان JSP داشت، یا اینکه می‌توان کلاس‌های ایستایی تعریف نمود که در تمام طول برنامه تنها یک نمونه از آن‌ها بتوان ایجاد کرد.

[1] Replication

برای مثال اگر متغییر A در یک صفحه مقدار داده شده باشد، این مقدار می‌تواند در صفحه‌های بعدی مورد استفاده قرار گیرد. به این ترتیب مقداری که در صفحه‌ی p1 به یک متغییر داده شده است در صفحه‌ی p2 قابل دسترسی و استفاده است. مشکل در دسترسی همزمان به یک متغییر سراسری از طریق چند کپی از یک صفحه‌ی وب (مانند صفحات ASP یا JSP) بروز می‌کند. به نحوی که چند کاربر ممکن است بخواهند به طور همزمان از طریق کپی‌های متفاوت صفحه‌ی p1 به متغییر A دسترسی پیدا کنند و هر کدام بخواهند مقدار آن را تغییر دهند.

یک متغییر سراسری به خودی خود، مشکلی ایجاد نمی‌کند و راه‌کارهایی برای هماهنگی استفاده از آن در هر زبان برنامه‌نویسی لحاظ شده است. اما زمانی که برنامه‌ی کاربردی مبتنی بر وب بر چند کارگزار قرار داده شود تا بارکاری آن توزیع و ترافیک کنترل گردد، چند کارگزار داریم که هر یک، یک متغییر سراسری دارد و این متغییرها هیچ ارتباطی با هم ندارند. اما مطلوب ما این است که تمام این متغییرها یکی باشند و دسترسی سریع و امن به آن‌ها برای هر کارگزار امکان‌پذیر باشد.

یک مثال عملی از این مورد می‌تواند یک سیستم مزایده‌ی الکترونیکی باشد[۳۰]، که ممکن است چندین نفر بخواهند همزمان وارد مزایده شوند. هر کدام یک کپی از صفحه‌ی مربوط به مزایده را از طریق مرورگر خود دریافت می‌کنند و می‌خواهند آخرین قیمت پیشنهادی توسط سایرین را مشاهده نمایند و در ضمن قیمت پیشنهادی خود را ارائه دهند. این امر با یک کارگزار به خوبی کار می‌کند، اما در صورت توزیع محتوی برنامه‌کاربردی مبتنی بر وب، بر چند کارگزار باید فکری برای متغییر مشترک کرد.

روش‌های ذکر شده برای توزیع بارکاری، چون کاری با ماهیت برنامه‌ی کاربردی مبتنی بر وب ندارند هدف را توزیع درخواست می‌دانند، چاره‌ای برای این کار نیندیشده‌اند[۳۰]. آنچه در مرجع [۳۰] برای این امر پیشنهاد شده طراحی و پیاده‌سازی رویکردی است که استفاده از متغییرهای سراسری را در این حالت ممنوع کرده و دو کلاس با نام‌های Data Client و Data Server ایجاد نموده است که Data Server بر یک کارگزار با عنوان "وب‌سرور مخزن" قرار می‌گیرد و نگهداری متغییر و هماهنگی دسترسی به آن را بر عهده دارد و بقیه‌ی کارگزارهای نهایی با استفاده از Data Client به آن دسترسی می‌یابند.

۳-۵- نتیجه‌گیری

در این فصل ابتدا به اهمیت توزیع بار بین چند کارگزار برای کاهش بارکاری ارجاعی به یک کارگزار منفرد در زمان توسعه‌ی خدمات پرداخته شد. همانطور که موارد متنوعی برای توزیع بین چند کارگزار وجود دارد (از قبیل محاسبات، درخواست‌ها و ...)، روش‌های متنوعی نیز برای توزیع این موارد وجود دارد.

بر اساس آنچه در این فصل بیان گردید، رویکرد مناسب برای توزیع بارکاری، که بتواند پس از انجام مهندسی معکوس بر برنامه‌ی کاربردی مبتنی بر وب و تقسیم شدن برنامه به چند بخش با پیوستگی داخلی قوی، درخواست‌ها را به نحو مناسب بین قطعات تقسیم نماید، استفاده از توزیع درخواست‌ها با کمک توزیع‌کننده است. زیرا مشکلات مرتبط با نهان‌نگاری در گره‌های میانی شبکه که در توزیع در DNS مطرح بود را ندارد و همچنین پیچیدگی آن از توزیع در جانب کارگزار کمتر است و عمومیت آن از توزیع در جانب کاربر. اما از بین رویکردهای موجود در توزیع توسط توزیع کننده، رویکردی برای به‌کارگیری در این پایان‌نامه مفید خواهد بود که تا لایه‌ی ۷ در پشته‌ی پروتکلی شبکه بالا برود، زیرا پس از اعمال خوشه‌بندی بر صفحات یک برنامه‌کاربردی مبتنی بر وب و تعیین نام صفحات موجود در هر خوشه، لازم است توزیع درخواست‌ها بر اساس نام صفحات مورد ارجاع صورت پذیرد، که استخراج نام صفحه‌ی مورد ارجاع در سرایند بسته‌های HTTP که در لایه‌ی ۷ مطرح است، وجود دارد. به این ترتیب بهتر است از رویکرد L7 استفاده نمود.

همچنین تشریح گردید که یک بعد مهم توزیع بارکاری بین کارگزاران وب، که ممکن است دور از ذهن نیز به نظر برسد، مسأله‌ی متغییر مشترک سراسری است که زمانی بروز می‌کند که بخواهیم یک برنامه‌کاربردی مبتنی بر وب با داده‌ی مشترک سراسری را بر چند کارگزار توزیع کنیم. به روش‌مسأله متغییر سراسری نیز اشاره گردید و مشخص شد که این امر تا حد زیادی به زبانی که برنامه‌ی کاربردی مبتنی بر وب با آن پیاده‌سازی شده است نیز بستگی دارد. بدین ترتیب باید در زمان تقسیم برنامه‌کاربردی مبتنی بر وب بر چند کارگزار بایستی نحوه‌ی رفع مسأله‌ی متغییرهای مشترک سراسری نیز بیان گردد.

فصل چهارم

رویکرد پیشنهادی

۴-۱- مقدمه

در فصول گذشته به بررسی رویکردهای مختلف مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب و توزیع بارکاری کارگزاران وب پرداخته شد. در این فصل بر اساس راهکارهای بررسی شده و نقاط قوت و ضعف هر یک، ابتدا به معرفی رویکردی جدید که معماری برنامه‌های کاربردی مبتنی بر وب را بازیابی خواهد کرد، پرداخته خواهد شد و سپس جزییات آن رویکرد که مبتنی بر خوشه‌بندی است، بررسی می‌گردد و در نهایت روش توزیع بارکاری که بر مبنای رویکرد مهندسی معکوس انجام پذیرفته است، تشریح می‌گردد.

روش ارائه شده مبتنی بر گام‌های مختلفی است که ابتدا کد مبنای برنامه‌ی کاربردی مبتنی بر وب را به یک گراف چندگانه تبدیل می‌نماید و پس از تبدیل نمودن آن به گراف ساده، الگوریتم خوشه‌بندی را بر آن انجام می‌دهد، سپس بر اساس نتیجه‌ی خوشه‌بندی، پیکربندی جدیدی برای توزیع درخواست‌های ارجاعی به برنامه‌کاربردی مبتنی بر وب، پیشنهاد می‌کند.

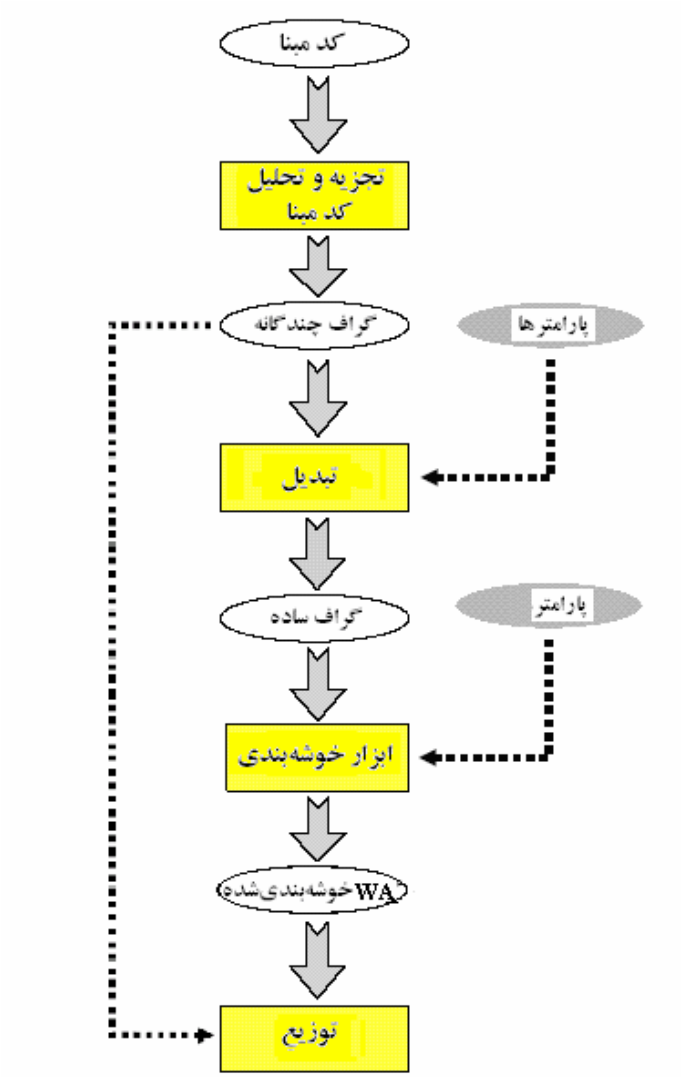
فرآیند مورد نظر در چهار مرحله به شرح زیر صورت می‌پذیرد که نمودار جریان[1] آن در شکل ۴-۱ آمده است:

[1] Flowchart

- *تجزیه و تحلیل کد مبنا:* برنامه‌ی کاربردی مبتنی بر وب تجزیه می‌شود و ساختاری در قالب گراف چندگانه ایجاد می‌گردد. ابزار این قسمت به صورت جداگانه ایجاد خواهد شد.
- *تبدیل:* این مرحله گراف چندگانه را به عنوان ورودی در قالب مجموعه‌ای از پارامترها از یک فایل پیکربندی مجزا می‌پذیرد. بر اساس این پارامترها، *معیار تشابه*(Similarity Measure) بین گره‌های متصل به هم در گراف چندگانه محاسبه خواهد شد و ساختار گراف ساده به عنوان خروجی تولید می‌گردد.
- *خوشه‌بندی:* در این مرحله، گراف وابستگی خوشه‌بندی خواهد شد (یک الگوریتم خوشه‌بندی بر گراف خوشه‌بندی نشده اجرا می‌گردد) و پارامتری که مرتبط با تعداد کارگزاران موجود در مرحله‌ی بعد است به عنوان نقطه‌ی برش به الگوریتم اعمال می‌گردد، خروجی یک گراف خوشه‌بندی شده خواهد بود که هر خوشه‌ی آن مجموعه‌ای از صفحات و قطعات برنامه‌کاربردی اولیه است.
- *توزیع:* کپی‌هایی از برنامه‌ی کاربردی مبتنی بر وب، بر کارگزاران موجود در خوشه‌ی کارگزاران قرار داده می‌شود و یک کارگزار اصلی نیز توزیع بارکاری آن‌ها بر اساس ارجاعات به خوشه‌های تعیین شده را بر عهده می‌گیرد و تغییرات لازم در کد مبنا برای این امر لحاظ می‌گردد، از برخی یافته‌های مرحله‌ی اول نیز برای تغییر در کد استفاده می‌شود.

لازم به یادآوری است که شکل مذکور بیان‌گر ترتیب زمانی مراحل انجام فرآیند مهندسی معکوس بر اساس فرآیند مورد نظر است و ورودی و خروجی‌های هر مرحله را نیز مشخص می‌کند، اما این بدان معنی نیست که هر مرحله تنها یک‌بار رخ خواهد داد و ممکن است یکی از مراحل، مثلاً خوشه‌بندی، چندین بار انجام پذیرد تا کنترل به مرحله‌ی بعد برود.

به این ترتیب در ادامه‌ی این فصل، ابتدا برای مرحله‌ی اول، یک مدل نمایش برنامه‌کاربردی مبتنی بر وب که نقش ورودی فرایند پیشنهادی مهندسی معکوس را دارد، عرضه می‌شود. در این مدل که مبتنی بر گراف است، لبه‌ها و گره‌ها به دقت مشخص شده‌اند و سعی بر آن بوده است تا با هرچه عمومی‌تر کردن آن، قابلیت اعمال بر برنامه‌های کاربردی تحت وب متنوعی (با پیاده‌سازی‌های متفاوت) را داشته باشد.



شکل ٤-١ مراحل بازیابی معماری برنامه‌کاربردی مبتنی بروب و توزیع آن بر کارگزاران

در مرحله‌ی دوم نحوه‌ی تبدیل گراف چندگانه، که ورودی مرحله‌ی دوم است، به گراف ساده بیان شده است، این امر بدان خاطر است که مرحله‌ی سوم که عملیات خوشه‌بندی را صورت خواهد داد، این امر را بر یک گراف ساده انجام می‌دهد. الگوریتم خوشه‌بندی انتخاب شده، سلسله‌مراتبی تجمعی است. در نهایت برای مرحله‌ی چهارم نیز جزییات عملیات توزیع درخواست‌ها تشریح شده است که این امر به نتیجه‌ی خوشه‌بندی در مرحله قبل وابسته است.

۴-۲- مرحله‌ی یکم: تجزیه و تحلیل

هدف در این مرحله تحلیل کد مبنا، ردیابی و کمی نمودن انواع ارتباطات بین قطعات معماری[1] (که در زیر نحوه‌ی انتخاب آن‌ها تشریح می‌شود) در نرم‌افزار کاربردی مبتنی بر وب است. ورودی این گام کد مبنای سیستم است و خروجی آن یک ساختار گراف چندگانه از صفحات وب و روابط بین آن‌هاست.

این مرحله به وضوح زمان‌گیرترین مرحله در فرایند مهندسی معکوس است، زیرا در آن، کد مبنای سیستم خوانده و تجزیه و تحلیل می‌شود. سپس قطعات معماری مورد پردازش قرار می‌گیرد و انواع مختلف ارتباطات بین آن‌ها محاسبه می‌شود. نتیجه، یک گراف چندگانه[2] خواهد بود که تعمیمی از گراف‌هاست با این مشخصه که تعداد دلخواهی از لبه‌ها بین دو گره، و حلقه (که عبارت است از اتصال یک گره به خودش) را اجازه می‌دهد. در نحوه‌ی انتخاب گره‌ها و لبه‌های این گراف که تشکیل دهنده‌ی قطعات و روابط بین آن‌ها در نرم‌افزار به شمار می‌آید، سعی بر لحاظ نمودن حداکثر عمومیت بوده است، یعنی مدلی در نظر گرفته شده است که به سادگی برای تمام برنامه‌های کاربردی مبتنی بر وب قابل استفاده باشد و اجزای آن در همه‌ی برنامه‌های کاربردی مبتنی بر وب مشترک باشد.

۴-۲-۱- خصوصیات گره‌ها

یک جزء مهم از مهندسی معکوس موفق، نمایش مناسب برای درک بهتر از کد مبنای بررسی شده است. انتخاب گره‌ها به صورت‌های مختلفی می‌تواند صورت پذیرد و این امر به موارد متعددی از جمله سطح مهندسی معکوس و بستر و نحوه‌ی پیاده‌سازی آن وابسته است. یعنی بسته به اینکه بازیابی در کدام سطح از بخش ۲-۲-۳ صورت پذیرد و برچه موجودیت‌هایی اعمال گردد، یا برنامه‌ی تحت تحلیل از چه روش‌شناسی‌ایی[3] استفاده کرده باشد، می‌توان برای انتخاب گره‌ها روش‌های مختلفی پیشنهاد کرد. بدین ترتیب در انتخاب گره‌های گراف لازم است بدانیم که چگونه مورد استفاده قرار خواهند گرفت.

این امر که کدام مدل به نتایج بهتری منجر خواهد شد، به طور دقیق مشخص نیست و نیازمند به بررسی فراتری از دامنه‌ی این مستند است، اما محرز است که مدل‌هایی که به تفاوت ماهیتی برنامه‌های مبتنی بر وب توجه بیشتری داشته باشند در اولویت هستند و از طرفی عمومیت فرایند نیز نباید خدشه‌دار شود.

---

[1] Architectural Components
[2] Multigraph
[3] Methodology

بر این اساس، گره‌ها در گراف‌چندگانه مشابه با آن‌چه در مراجع [۳] و [۸] آمده است و در بخش ۲-۴-۲-۱ نیز ذکر گردید، قطعات اصلی برنامه‌ی کاربردی مبتنی بر وب یا همان صفحات آن درنظر گرفته شده‌اند، که می‌توانند به *صفحات* کارگزار و *صفحات* مشتری تقسیم شوند. یک صفحه‌ی مشتری می‌تواند مبتنی بر قاب باشد، یعنی توسط یک ساختار صفحه‌ی مخصوص که *مجموعه قاب*[1] نامیده می‌شود، به *قاب‌هایی*[2] تقسیم شود. نیز یک برنامه‌ی کاربردی مبتنی بر وب، می‌تواند مشتمل بر مؤلفه‌های مشتری یا اشیای وب از قبیل عکس‌ها، اشیای چندرسانه‌ای (صدا و فیلم)، اپلت‌ها و غیره هم .... باشد که با توجه به رویکرد کلی که در ادامه می‌آید، در مدلی که در اینجا در نظر گرفته شده است، از در نظر گرفتن آن‌ها صرفنظر شده است (شکل ۴-۲).

نکته‌ی نهایی این‌که هر انتخابی برای گره‌های گراف مورد نظر، قطعاً انتخاب یال‌های آن را نیز تحت تأثیر قرار خواهد داد.

۴-۲-۲- خصوصیات لبه‌ها

در این بخش، پیش از مشخص نمودن انواع لبه‌ها، خصیصه‌هایی را که هر یک بایستی دارا باشند، بیان خواهیم نمود. لبه‌های بیانگر وابستگی بایستی دست کم سه نوع خصیصه را داشته باشند:

- *مبدأ*: بیانگر گره‌ی آغاز لبه است
- *مقصد*: بیانگر گره‌ی انتهای لبه است
- *نوع*: بیانگر نوع وابستگی ایجاد شده توسط لبه است، که سبب اختصاص وزن به‌خصوص به آن نیز خواهد شد

۴-۲-۳- انواع وابستگی‌ها

بر اساس انتخاب انجام پذیرفته برای گره‌ها، لبه‌ها که بیانگر وابستگی بین صفحات هستند، انواع مختلفی از وابستگی‌ها از جمله Link، Submit، Redirect و Include را شامل خواهند شد.

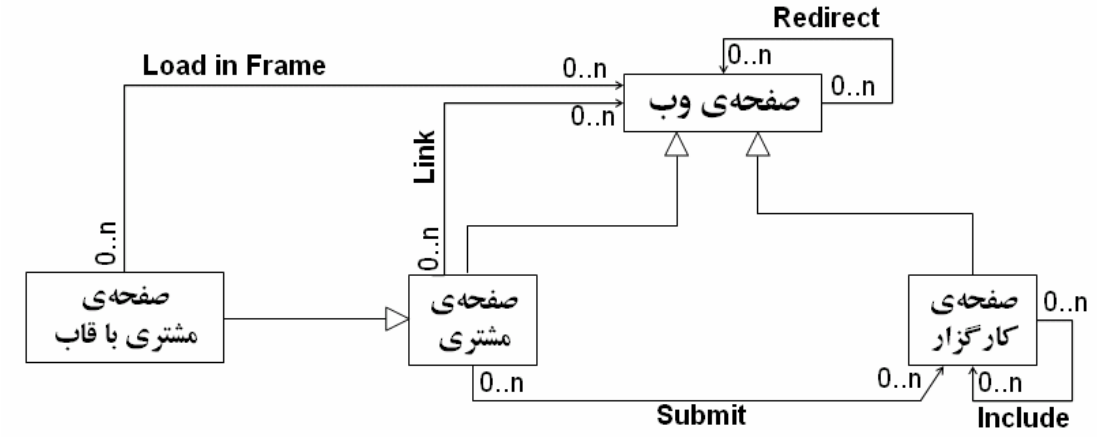
زمانی که یک صفحه (یا یک قطعه از صفحه) مشتمل بر پیوندهای ابرمتنی به خودش یا سایر صفحات باشد، رابطه‌ای که Link نام دارد بین صفحه (یا قطعات صفحه) و سایر صفحات، در مدل، نمایش داده خواهد شد. همچنین رابطه‌ی Submit[3] بین یک فرم و صفحه‌ی کارگزار که فرم داده را ایجاد[1] می‌کند،

[1] Frameset
[2] Frames
[3] Submit

رابطه‌ی Redirect بین یک اسکریپت و یک صفحه‌ی وب، رابطه‌ی Include بین یک اسکریپت مشتری و مؤلفه‌ی مشتری (یا بین دو صفحه‌ی کارگزار) نیز در مدل در نظر گرفته می‌شوند. به همین ترتیب رابطه‌ی بارگذاری در قاب[۲] که بین یک صفحه‌ی مشتری دارای قاب و مجموعه صفحاتی که با علامت[۳] <frame> به آن‌ها ارجاع می‌دهد، ایجاد می‌گردد.

هر یک از این روابط، با نوع متفاوتی از لبه‌ها در گراف‌چندگانه نمایش داده می‌شود. همچنین به هر لبه یک وزن منتسب می‌شود، که متناظر با قوت وابستگی است.

ابهامی که در این‌جا به ذهن خطور می‌کند آن است که چه تعداد و چه نوعی از وابستگی‌ها باید مورد توجه قرار گیرند. لیست انتخاب‌های موجود می‌تواند خیلی گسترده باشد. هنوز کاملاً مشخص نیست که کدامیک از این‌ها سودمندتراند و انتخاب‌های صورت گرفته تاکنون نیز توجیه علمی نداشته‌اند و اغلب بر اساس تجربه بوده‌اند[۳ و ۲۶]. اما در این‌جا، با درنظر گرفتن عمومیت مسأله و با توجه به اهمیت و تکرار بیشتر و نیز انتخاب‌های مراجع پیشین[۳]، سه نوع رابطه‌ی Link، Submit و Redirect به عنوان روابط اصلی در نظر گرفته شده‌اند.



شکل ٤-۲ مدل مفهومی انتخابی برای برنامه‌های کاربردی تحت وب

بقیه‌ی روابط دارای خصیصه‌هایی هستند که جهت تسهیل درک و نیز هماهنگی بیشتر با مرحله‌ی بعد، به صورت محلی در همین مرحله حل می‌شوند. برای مثال از آن‌جا که رابطه‌ی Include را با قرار دادن فایل

[1] Elaborate
[2] Load_in_frames
[3] tag

مشخص شده در آرگومان آن، در فایل مقصد، می‌توان حل نمود و گرهی و یالی اضافه به نمودار نیفزود، این مورد، در پیاده‌سازی مرحله‌ی اول حل شده است همچنین برای رابطه‌ی بارگذاری در قاب[1] نیز می‌توان قاب‌های مرتبط با یک صفحه را با Link جایگزین نمود. جزییات بیشتر در تشریح پیاده‌سازی آمده است.

مدل ایجاد شده، بر اساس آن‌چه تاکنون بیان گردید، با نمودارهای مشابه به UML ترسیم شده و در شکل ۴-۲ آمده است.

۴-۲-۴- محاسبه‌ی وابستگی لبه‌ها

سه مورد اصلی در زیر بررسی خواهند شد: Link، Submit و Redirect، که به ترتیب به مقادیر مثبت W_S، W_r و W_l را به آن‌ها منتسب خواهیم نمود. همانطور که می‌دانیم یک رابطه‌ی Redirect بین دو صفحه، درجه‌ی بالاتری از پیوستگی نسبت به ارتباط Link برقرار می‌کند، زیرا ارتباط از طریق Link بیانگر انتقال از یک صفحه به صفحه‌ی دیگر توسط محاوره با کاربر است، در حالی که یک عبارت Redirect معمولاً بر انتقال کنترل به صورت خودکار از یک صفحه‌ی وب به صفحه‌ی دیگر دلالت دارد. همچنین، یک رابطه‌ی Submit بین یک صفحه‌ی مشتری و یک صفحه‌ی کارگزار درجه‌ی بالاتری از پیوستگی نسبت به ارتباط Redirect، تولید می‌کند، زیرا معمولاً علاوه بر انتقال کنترل، با یک جریان داده بین صفحات نیز همراه است. به این ترتیب یک رابطه‌ی Submit از یک رابطه‌ی Link نیز قوی‌تر است. باید در انتساب مقادیر به متغییرهای Ws، Wr و Wl رابطه‌ی زیر را در نظر داشته باشیم[۳]:

$$W_l < W_r < W_s$$

انتخاب اعداد مناسب برای هر یک از موارد فوق نیز نیاز به بررسی جدگانه و خارج از این مستند دارد. علت آن است که برای یافتن عدد مناسب متناظر با هر یک بایستی روند کل فرآیند را پیاده‌سازی نمود و با آزمودن انتخاب‌های متعدد، اعداد مناسب را یافت که این امر در مرجع [۳] صورت پذیرفته است و نتایج زیر حاصل شده است: با در نظر گرفتن وزن ۱ برای W_l، عدد ۲/۴ برای W_rو ۳ برای W_S پیشنهاد می‌گردد. به این ترتیب در صورتی‌که از قطعه‌ی A به قطعه‌ی B یا برعکس، یک ارتباط از نوع Link داشته باشیم، یک برچسب با وزن ۱ به گراف چندگانه افزوده خواهد شد و به همین ترتیب برای سایر ارتباطات هم انتساب انجام خواهد گرفت.

[1] Load in Frame

۴-۳- مرحله‌ی دوم: تبدیل

ورودی این مرحله ساختار گراف‌چندگانه از برنامه‌ی کاربردی مبتنی بروب است که به صورت مجموعه‌ای از پارامترها به آن عرضه می‌شود و خروجی آن یک گراف جهت‌دار یا بی‌جهت از همان برنامه است که عملیات تبدیل بر آن انجام گرفته است و انواع مختلف وابستگی بین گره‌های گراف‌چندگانه‌ی قبلی به یک *معیار تشابه* تکی که به الگوریتم می‌گوید که دو موجودیت تا چه اندازه مرتبط (مشابه) یا متفاوت (نامشابه) هستند، خلاصه شده است و به این ترتیب نمایش کنونی این گراف برای انجام خوشه‌بندی مناسب گشته است. به این ترتیب از این مرحله به بعد دیگر کاری با برنامه‌ی کاربردی مبتنی بر وب نخواهیم داشت و تنها با مفاهیم خوشه‌بندی سروکار خواهیم داشت. لبه‌های گراف ساده‌ی حاصل شده از این مرحله، به شکل غیر جهت‌دار در نظر گرفته خواهد شد، زیرا که بیشتر الگوریتم‌های خوشه‌بندی جهت را در ملاحظات خود به شمار نمی‌آورند[۹] و تنها به یک معیار تشابه بین هر زوج از گره‌ها نیازمندند.

برای تبدیل ساختار گراف لازم است که نحوه‌ی محاسبه‌ی معیار تشابه جدید بیان گردد، این معیار بر اساس معادله زیر است [۷] که در آن N تعداد انواع مختلف وابستگی (یال) بین گره‌هاست، $w(A,B)_i$ وزن لبه‌ای است که دارای نوع i می‌باشد و A را به B متصل کرده است (تعداد تکرارهای آن لبه)، همچنین p_i پارامتری است که وزن متناظر با iامین جمله در جمع‌وزن‌دار را نشان می‌دهد.

$$sim(A,B) = \sum_{i=1}^{N} w(A,B)_i * p_i \qquad (۱)$$

مجموعه پارامترهای P_i در عبارت فوق نقش بسیار مهمی در خوشه‌بندی نتیجه خواهد داشت. این پارامترها اهمیت نسبی انواع مختلف وابستگی‌ها را در مقدار تشابه نهایی معین می‌کنند. که مقدار آن پیش از این تعیین گردید. آنچه در این چهارچوب لازم به توجه است این است که مقدار نسبت داده شده به پارامتر P_i، بایستی قابل تغییر باشد و نباید پیاده‌سازی آن فاقد انعطاف باشد.

۴-۴- مرحله‌ی سوم: خوشه‌بندی

ورودی این گام، گراف یا گراف جهت‌دار نمایانگر سیستم است. این گراف همچنین ممکن است ساختاری پیچیده داشته باشد و از اجرای پیشین همین مرحله حاصل شده باشد. خروجی این مرحله نیز ساختار گرافی خوشه‌بندی شده‌ی سیستم است که بر اثر اعمال خوشه‌بندی بر قطعات قویاً پیوسته[1] ایجاد شده است و در واقع نمایی سطح بالا با سطح تجرید دلخواه کاربر است. این مرحله که در واقع خوشه‌بندی اصلی در آن صورت می‌گیرد، گزینه‌های منتخب برای مؤلفه‌ها[2]، زیرسیستم‌ها و یا قطعاتی را که می‌توانند از برنامه‌ی کاربردی مبتنی بر وب کنونی استخراج شوند، نشان می‌دهد و عنصر مرکزی آن در واقع همان موتور خوشه‌بندی است.

گرچه فرآیند اصلی باید کاملاً خودکار باشد، اما انواع مختلف خوشه‌بندی ممکن است بر اساس نیاز به پارامترهای ورودی، به نظارت نیز احتیاج داشته باشند. همچنین در روندی که ما برای توزیع برنامه‌کاربردی مبتنی بر وب بر چند کارگزار تشریح خواهیم کرد، از تعداد یا ضریبی از تعداد کارگزارها برای تعیین نقطه‌ی برش الگوریتم خوشه‌بندی استفاده خواهیم کرد.

با توجه به اینکه خوشه‌بندی رویکردی بسیار عمومی است، در هنگام انتقال آن به زمینه‌ی کنونی، باید لحاظ گردد که در مجموع همیشه برای سیستم‌های کامپیوتری، تمایل به سلسله‌مراتبی از خوشه‌ها که اجازه تودرتو بودن آن‌ها را نیز می‌دهد، بیشتر است تا خوشه‌بندی ساده، و این مطلب از ماهیت سلسله‌مراتبی سیستم‌های نرم‌افزار نشأت می‌گیرد. زبان‌های متنوع برنامه‌نویسی مخصوص برنامه‌های کاربردی مبتنی بر وب، این مورد را به صورت‌های مختلفی مورد حمایت قرار داده‌اند، اما همگی برسر وجود آن اشتراک دارند. به این ترتیب در رویکرد ما خوشه‌بندی سلسله‌مراتبی مطلوب نظر خواهد بود.

این الگوریتم از دسته‌ی تجمعی[3] است و با شروع از تمام موجودیت‌ها به عنوان خوشه‌های متمایز و انتخاب مکرر جفت خوشه‌های با بیشترین تشابه و پیوند آن‌ها با هم، عمل خوشه‌بندی را به انجام می‌رساند که شبه‌کدی از آن در ادامه آمده است.

با دقت در الگوریتم مذکور می‌توان دریافت که از سه مرحله‌ی تکراری تشکیل شده است. ابتدا دو خوشه‌ی مشابه انتخاب می‌شوند، میزان تشابه یا عدم تشابه در آغاز برای هر زوج از خوشه‌ها محاسبه می‌شود. سپس دو خوشه با بیشترین تشابه، با هم ادغام می‌شوند و در نهایت محاسبه‌ی ضریب تشابه بین

---

[1] Strongly connected
[2] Modules
[3] Agglomerative

خوشه‌ی تازه تشکیل شده و سایرین، صورت می‌پذیرد. و این امر تا زمانی که یک خوشه باقی بماند، قابل انجام است.

منظور از Sim(R,C) در خط هفتم همان قاعده‌ی به‌هنگام‌سازی تشابه[1] است. قاعده انتخابی در این الگوریتم، پیوند ساده[2] انتخاب شده است. بر طبق این قاعده Sim(R,C)=max(Sim(A,C), Sim(B,C)) که R اجتماع A و B است و C یک خوشه‌ی دلخواه است.

```
Set each Page a separate cluster
Add all Clusters to Θ (the set of top level clusters)
While (There is more than one cluster left in Θ)
     Let A and B be the two most similar clusers (A,B ∈Θ)
     Create cluster R=A∪B
     Θ=Θ-{A,B}
     Compute sim(R,C),  ∀C∈Θ
     Θ=Θ+{R}
End While
```

همانطور که در الگوریتم‌های خوشه‌بندی سلسله‌مراتبی رایج است، یک مشکل اساسی تعیین نقطه‌ی برش[3] یا زمانی است که خوشه‌ها در آن در سطح تجرید مناسب قرار دارند و درخت حاصل، درخت نتیجه است. یافتن این نقطه کلیدی است و از معیارهای نرم‌افزار[4] می‌توان برای یافتن آن استفاده نمود[۷] ، به عنوان مثال می‌توان برای زیرسیستم‌های ایجاد شده، اندازه و میزان پیچیدگی تعیین نمود.

به علت آنکه خوشه‌های حاصل شده از قطعات برنامه‌کاربردی بیشترین تشابه داخلی را با یکدیگر دارند همچنین تشابه این خوشه‌ها با خوشه‌های دیگر کمتر است، احتمال ارجاع یک کاربر به اجزای داخل یک خوشه بیشتر از ارجاع به خارج از آن است. به این ترتیب با توجه به مرحله‌ی بعد که توزیع بارکاری بر چند کارگزار وب انجام خواهد پذیرفت، بهتر است پاسخ‌گویی به درخواست‌های داخل هر خوشه به یک کارگزار سپرده شود. در نتیجه پیشنهاد می‌شود، تعداد خوشه‌ها برابر با تعداد کارگزارها یا ضریبی از آن در نظر گرفته شود.

---

[1] Similarity Update Rule
[2] Simple Linkage
[3] Cut point
[4] Software metric

۴-۵- مرحله‌ی چهارم: توزیع

پس از تقسیم برنامه‌کاربردی مبتنی بر وب به قطعات با پیوستگی داخلی بالا، نوبت به توزیع بارکاری ارجاعات به آن فرا می‌رسد. با توجه به رویکردهای پیشین بررسی شده، درباره‌ی کارگزاران وب، دو چیز برای توزیع وجود دارد:

۱. توزیع مستندات

۲. توزیع درخواست‌های رسیده از مشتری‌ها

براساس نتایج حاصل شده از مرحله‌ی قبل، مجموعه‌ی از خوشه‌ها در دسترس خواهیم داشت که با لحاظ نمودن انسجام داخلی آن‌ها، می‌توان هر یک را بر یک کارگزار در مجموعه‌ی کارگزارها قرار داد و انتظار داشت با تشخیص درخواست کاربر و ارسال آن به خوشه‌ی مطلوب، به رویکردی مفید در زمینه‌ی توزیع بار دست یافته شود. اما در این راستا مسایل متعددی ممکن است مطرح شوند که عملکرد صحیح برنامه‌ی کاربردی مبتنی بر وب را دستخوش تغییر کنند. به عنوان مثال بازسازی ارتباطات خارجی خوشه‌ها یکی از مواردی است که باید به درستی رسیدگی شود. به این ترتیب لازم است با در نظر گرفتن این موارد، رویکردی مناسب برای توزیع بارکاری بین کارگزارها، عرضه نمود.

۴-۵-۱- شیوه‌ی مناسب توزیع

دو نکته‌ی مهم در این زمینه عبارتند از: نحوه‌ی برخورد با مسأله‌ی متغییرهای مشترک سراسری و نحوه‌ی توزیع. بر اساس آنچه در قسمت ۳-۴ بیان شد، زمانی که بارکاری یک برنامه‌کاربردی مبتنی بروب (وب‌سایت پویا) بین چند کارگزار وب توزیع می‌گردد، پاسخ مسأله‌ی متغییر مشترک سراسری ضروری است. در این رابطه، در صورت نیاز از پروتکل طراحی شده در مرجع [۳۳] استفاده خواهد شد که با کمی تغییر، بر سایر بسترها نیز قابل اعمال است.

نکته‌ی مهم دیگر، پاسخ به این پرسش است که آیا قرار دادن هر خوشه بر یک کارگزار امر مناسبی است؟ پاسخ به این پرسش ما را در انتخاب شیوه‌ی صحیح توزیع رهنمون خواهد کرد. از آن‌جا که هدف ما از توزیع قطعات بر کارگزاران مختلف، افزایش سرعت دسترسی کاربران و کاهش بار کاری کل برای هر کارگزار است، جای‌دهی هر خوشه بر یک کارگزار و شبیه‌سازی روند کلی برنامه‌کاربردی بین کارگزارها برای ارتباطات بین خوشه‌ای لزوماً به هدف مذکور منجر نخواهد شد، یا دست‌کم رویکردی کاراتر از آن وجود دارد.

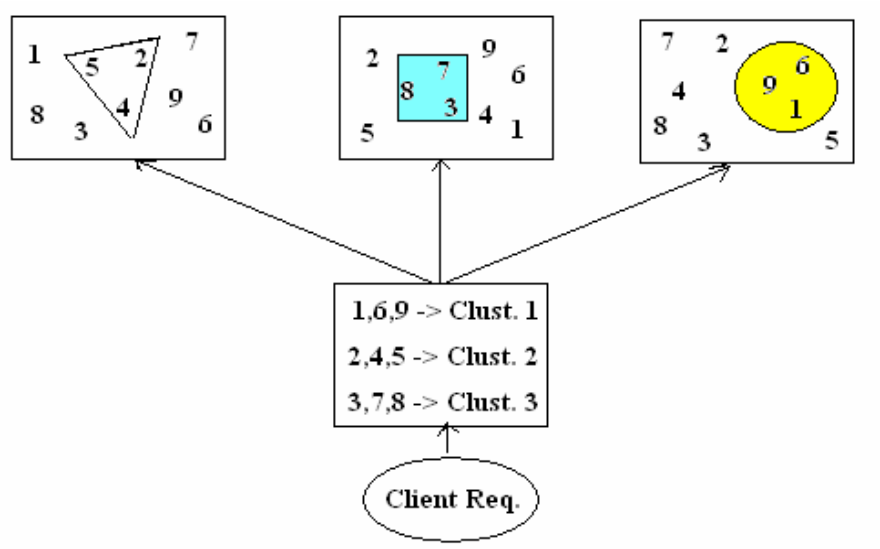
با دقت در حالات ممکن برای روابط بین خوشه‌ها، پاسخ به این امر تسهیل می‌گردد؛ با فرض ایستا بودن قطعات و عدم تبادل داده‌ی آن‌ها با یکدیگر رویکرد توزیع محض خوشه‌ها(جای‌دهی هر خوشه بر یک کارگزار و شبیه‌سازی روند کلی برنامه‌کاربردی بین کارگزارها برای ارتباطات بین خوشه‌ای)، بسیار سودمند نیز خواهد بود، زیرا انتقال کنترل از یک خوشه به یک خوشه‌ی دیگر (یا به عبارت دیگر از یک کارگزار به کارگزار دیگر) تنها با دستورات ساده‌ی Redirect امکان‌پذیر است و مشکلی برای جلسه‌ی[1] کاربر به وجود نخواهد آمد.

اما در حالت پویایی ارتباط (به عنوان مثال Login نمودن کاربر در یک خوشه)، جلسه کاربر در زمان انتقال از یک کارگزار به کارگزار دیگر، بایستی با همان مشخصات حفظ گردد، به عبارت دیگر امنیت و سطح دسترسی و تبادلات پیشین بایستی برای خوشه‌ی جدید نیز هماهنگ گردد و جهت نیل به این امر لازم است پروتکلی طراحی گردد که آن را شبیه‌سازی نماید. گرچه احتمال روابط بین قطعات کم است(زیرا خوشه‌ها دارای درجه انسجام داخلی بالا و پیوستگی خارجی کم هستند)، اما سربار ناشی از تبادلات خوشه‌ها بر اثر انتقال کنترل ممکن است، مزایای سرعت دسترسی و کاهش بارکاری را متأثر نماید. با توجه به آن‌چه بیان گردید، رویکرد زیر پیشنهاد می‌شود.

۴-۵-۲- رویکرد توزیع بارکاری

شکل زیر بیانگر نحوه‌ی منطقی توزیع بارکاری بین کارگزاران است. این نمودار تفاوت اساسی با درک شهودی ما، از تقسیم بار بر اساس خوشه‌ها دارد. در این نحوه‌ی توزیع بار، لازم نیست که هر خوشه را به تنهایی بر یک کارگزار قرار داد، بلکه کافیست یک کپی از برنامه‌ی کاربردی مبتنی بر وب را بر هر کارگزار نهاد و ارجاع‌های مربوط به هر خوشه را توسط کارگزار توزیع‌کننده به یک کارگزار به‌خصوص هدایت نمود. این امر سبب می‌شود که ترافیک عمده‌ی مربوط به هر خوشه، تنها به یک کارگزار هدایت شود و قرار گرفتن کامل برنامه‌کاربردی بر هر کارگزار سبب جلوگیری از ارجاع به خوشه‌های دیگر می‌شود و پاسخ‌گویی به شیوه‌ی محلی صورت می‌گیرد و به این ترتیب زمان عمده‌ای در پاسخ به کاربر صرفه‌جویی می‌شود و هدف اصلی که توزیع بار واقعی و کاهش زمان پاسخ بود، برآورده می‌شود.

[1] Session



شکل ٤-٣ شمای منطقی توزیع بار بین کارگزارها

کارگزار دیگری که در شکل نشان داده نشده است، کارگزار ذخیره‌ی داده‌هاست که در واقع نگهداری بانک اطلاعاتی نرم‌افزار کاربردی مبتنی بر وب و ارجاعات به آن را برعهده دارد. همچنین برای مسأله‌ی متغییرها مشترک سراسری نیز از آن کمک گرفته خواهد شد.

با توجه به اینکه به جای توزیع خوشه‌ها بر کارگزاران، تمام برنامه‌کاربردی را روی هر کارگزار قرار می‌دهیم، ممکن است این پرسش به ذهن خطور کند که آیا هدر رفتن فضای اضافی در این زمینه نسبت به افزایش کارایی ناشی از کاهش ارجاعات بین کارگزارها، قابل توجیه است؟ از آن‌جا که فضای اختصاصی در مرورگرها امروزه بسیار ارزان شده است و مسأله‌ی کارایی هنوز به عنوان یک گلوگاه مطرح است، پاسخ به این امر بدیهی است.

اما ایراد دیگر وارد بر تکرار کل برنامه‌ی کاربردی مبتنی بر وب در هر خوشه، افزونگی و امکان به وجود آمدن تناقض در داده است که درباره‌ی رویکرد پیشنهاد شده، این امر منتفی است، زیرا برای نگهداری داده‌ها یک کارگزار مستقل در نظر گرفته شده است و تنها قسمت مرتبط با کد برنامه برکارگزارها توزیع شده است. اما امکان بروز مشکلی به نام مسأله‌ی متغییرهای سراسری وجود دارد که در بخش پیشین به آن پرداخته شد.

همچنین لازم است ذکر گردد که کارگزار توزیع کننده، تنها مشتمل بر صفحه‌ی اول برنامه‌کاربردی است و بار ترافیکی آن تنها توزیع درخواست‌هاست. در این کارگزار یک جدول نگاشت درخواست‌ها وجود دارد که بر اساس خوشه‌بندی انجام پذیرفته تنظیم شده است و توزیع درخواست‌ها بر مبنای آن

صورت می‌گیرد و این امر مزیت توزیع هدایت شده را نسبت به الگوریتم‌هایی مانند نوبت گردشی دی-ان-اس نشان می‌دهد. زیرا به علت آن‌که رویکرد پیشنهادی در لایه کاربرد (لایه ۷ شبکه در مدل OSI) کار می‌کند، از مزایای عمده‌ی آن می‌توان به امکان آمارگیری[1] از درخواست‌های رسیده اشاره کرد، که این امر سبب می‌شود پس از مدتی از کار سیستم، از بازخورد[2] آن استفاده نمود و خوشه‌های شلوغ‌تر را تشخیص داد و بر تعداد کارگزارهای آن‌ها افزود و خوشه‌های خلوت‌تر را نیز یافت و با هم ترکیب نمود. این‌بار توزیع بارکاری بین خوشه‌های مشابه را می‌توان بر اساس الگوریتم نوبت گردشی انجام داد.

در نهایت باید دقت داشت که چون توزیع درخواست‌ها در لایه کاربرد (لایه ۷ شبکه در مدل OSI) صورت می‌پذیرد، برای هر ارجاع عمل نگاشت صورت نمی‌گیرد، بلکه برای هر جلسه این امر انجام می‌شود، زیرا پس از شناسایی خوشه‌ی مورد درخواست و هدایت درخواست به آن، دیگر ارتباطات در قالب این جلسه به صورت مستقیم با همان خوشه خواهد بود و نیازی به توزیع‌کننده نیست. این امر در پیاده‌سازی صریح‌تر تشریح شده است. به این ترتیب می‌توان این توزیع بار را از نوع توزیع‌بار مبتنی بر توزیع‌کننده به شیوه‌ی L7 که در فصل پیش به آن اشاره شد، ذکر کرد. بدیهی است در این روش باید به تعداد کارگزارها آدرس IP عمومی[3] داشت و این امر امکان استفاده از این نوع خوشه‌بندی بر بستر اینترنت فراهم می‌کند.

۴-۶- نتیجه‌گیری

در این فصل با توجه به بررسی‌های مشروح انجام پذیرفته در فصول پیش، یک روش عمومی مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب در قالب فرآیندی چهار مرحله‌ای که در مرحله‌ی آخر توزیع بارکاری بین کارگزاران وب را نیز انجام می‌دهد، ارائه گردید. روش ارائه شده مبتنی بر گام‌های مختلفی است که ابتدا کد مبنای برنامه‌ی کاربردی مبتنی بر وب را به یک گراف چندگانه تبدیل می‌نماید و پس از تبدیل نمودن آن به گراف ساده، اعمال الگوریتم‌های خوشه‌بندی بر آن میسر می‌گردد و سپس بر اساس خوشه‌بندی انجام شده و اختصاص یک کارگزار به هر خوشه، توزیع بارکاری نیز صورت می‌گیرد.

---

[1] Logging
[2] Feedback
[3] Public IP

تقسیم فرایند مهندسی معکوس به مراحل چهارگانه‌ی تجزیه و تحلیل کد مبنا و ایجاد گراف چندگانه از سیستم کنونی، تبدیل ساختار گراف چندگانه به گراف ساده، خوشه‌بندی و توزیع درخواست‌ها براساس نتیجه‌ی خوشه‌بندی، یک چهارچوب عام برای بازیابی معماری برنامه‌های کاربردی مبتنی بر وب عرضه نموده است.

غیر از مرحله‌ی اول که تجزیه و تحلیل کد مبنای برنامه است، سایر موارد مستقل از برنامه هستند و تا حد امکان خودکار. این امر سبب می‌شود انعطاف فوق‌العاده‌ی روش‌ها و ابزار طراحی شده برای مراحل بعد خواهد شد و به رویکرد پیشنهادی، عمومیت بالایی نسبت به سایر روش‌های ارائه شده برای مهندسی معکوس نرم‌افزارهای کاربردی مبتنی بر وب داده است. به‌علاوه مدل انتخابی برای نمایش گرافی ورودی مرحله‌ی اول، روشی است که در بین تمام برنامه‌های کاربردی مبتنی بر وب مشترک است و با تغییرات کمی به تمام زبان‌های پیاده‌سازی این برنامه‌ها قابل اعمال می‌باشد. همچنین انتخاب الگوریتم خوشه‌بندی برنامه‌ی کاربردی مبتنی بر وب از نوع سلسله‌مراتبی نیز، هماهنگی آن با ماهیت سلسله‌مراتبی سیستم‌های نرم‌افزاری را تسهیل بیشتری بخشید.

در ادامه رویکرد جدید توزیع‌بار کاری بین کارگزاران وب، تشریح گردید که توزیع درخواست‌ها بر اساس ارجاع به خوشه‌ها صورت می‌گرفت. نیز مشکل بالقوه‌ایی که در این شیوه‌ی توزیع‌بار، امکان بروز دارد مطرح شد، که عبارت است از مشکل متغییرهای مشترک سراسری.

رویکرد کنونی توزیع‌بار، امکان آمارگیری از درخواست‌ها و تشخیص خوشه‌های پردرخواست و کم‌درخواست را میسر کرده است و براساس نتایج آن، پس از مدتی از کار سیستم، می‌توان با افزایش کارگزاران اختصاصی به خوشه‌های پردرخواست و ادغام خوشه‌های کم درخواست، بارکاری را به نحو هوشمندانه‌تری کنترل نمود. همچنین کپی تمام برنامه‌ی کاربردی مبتنی بر وب در هر خوشه سبب جلوگیری از ارجاع بین خوشه‌ای و پیچیدگی‌های مرتبط با آن می‌گردد و به این ترتیب با پاسخ‌گویی محلی، روند سریع‌تری در دسترسی ایجاد شده است.

فصل پنجم

جزییات پیاده‌سازی

۵-۱- مقدمه

پیاده‌سازی در پایان‌نامه‌ای که موضوع آن مهندسی معکوس برنامه‌های کاربردی باشد، بایستی دقیق و با رعایت اصول نگهداری و توسعه انجام پذیرفته باشد تا درک این سیستم نیز خود به مهندسی معکوس نیازمند نباشد! بدین ترتیب، پیش از هر چیز بایستی برخی مفاهیم سطح پایین به عنوان مطالب پیش‌نیاز برای درک روند پیاده‌سازی بیان گردد و بعد چهارچوب کلی محیط ایجاد شده تشریح شود، تا درنهایت بتوان به‌خوبی از جزییات ضمیمه‌هایی که حاوی کد مبنای نهایی هستند و در انتهای پایان نامه عرضه شده‌اند، آگاه شد.

در این راستا ابزاری با نام RELB (Reverse Engineering and Load Balancing) به زبان جاوا ایجاد گردیده است که بر مراحل چهارگانه‌ی ذکر شده در فصل پیش منطبق است و برای هر مرحله از فرآیند مهندسی معکوس که در شکل ۴-۱ آمده است، ابتدا مفاهیم پیش‌نیاز ضروری بیان گردیده است و سپس به بیان جزییات پیاده‌سازی و روند آن پرداخته شده است. در حد امکان نیز تلاش بر آن بوده است که دید مناسبی از ابزار تهیه شده ایجاد گردد و از بیان مطالب کم‌اهمیت‌تر اجتناب شود.

۵-۲- تعیین معیارهای ارزیابی

برای ارزیابی میزان کارایی رویکرد بایستی معیارهایی را از ابتدا مشخص نمود تا در نهایت، بر اساس میزان حصول آن‌ها درباره‌ی نتایج حاصل قضاوت کرد. معیارهای زیر برای ارزیابی نهایی روش پیشنهادی در نظر گرفته شده‌اند[۲۲]:

عمومیت[1]: فرآیند باید تا جایی که ممکن است فارغ از زبان پیاده‌سازی باشد و به بیان دیگر کمترین وابستگی را به زبان پیاده‌سازی داشته باشد. به این ترتیب می‌توان آن را به هر سیستمی اعمال نمود، یا دست‌کم برای تطابق آن با سیستم‌های مختلف، کار کمی لازم است صورت پذیرد. برای میسر شدن این امر دو راه‌حل لحاظ شده‌اند:

- تجزیه‌ی مناسب در مرحله‌ی اول

ایجاد گراف چندگانه با استفاده از تجزیه‌ی کد مبنا بایستی به گونه‌ای باشد که یک مدل عمومی از وب‌سایت کنونی به‌دست آید. در انتخاب چنین مدلی سعی بر عدم وابستگی به جزییات پیاده‌سازی یک بستر به‌خصوص است و در عوض خصوصیات مشترک برنامه‌ی کاربردی مبتنی بر وب بیشتر مد نظر خواهند بود.

- استقلال مراحل بعد، از مرحله‌ی اول

ساختار گراف چندگانه‌ی ایجاد شده از مرحله‌ی اول به یک شکل میانی تعیین شده ذخیره می‌شود، از این مرحله به بعد کاملاً از دامنه‌ی زبان پیاده‌سازی آن جدا می‌شود و می‌توان هر رویکردی را بر آن اعمال نمود، زیرا اکنون دیگر در قالبی است که از مجموعه‌ای یال‌ها و رئوس تشکیل یافته است.

خودکارسازی[2]: هر چند نمی‌توان دخالت عامل انسانی را در فرآیند مهندسی معکوس به صفر رساند[۴]، اما کمترین مبادلات با عامل انسانی در این میان، به خاطر صرفه‌جویی در هزینه، زمان و تلاش، مطلوب است. این امر ممکن است از کارایی الگوریتم بکاهد، زیرا عدم دخالت نیروی انسانی در زمان مناسب ممکن است سبب جهت‌یافتن الگوریتم به سوی انحراف گردد. به این ترتیب لازم است بین محاوره‌ای بودن و خودکار بودن الگوریتم تعادلی برقرار گردد، به نحوی که تا حد امکان ابزار خودکار باشد.

---

[1] Generalization
[2] Automation

قابلیت توسعه[1]: بدیهی است که فرآیند مهندسی معکوس برای برنامه‌های کاربردی با اندازه‌ی کوچک کارایی کمی دارد، زیرا سیستم کوچک پیچیدگی کمی دارد. اما باید تا جایی‌که امکان دارد، فرایند مهندسی معکوس را بتوان بر سیستم‌های بزرگ و بزرگتری اعمال نمود. بنابراین در ارائه‌ی رویکرد و پیاده‌سازی ابزار متناظر با آن بایستی این امر را لحاظ نمود. از راه‌کارهای مناسب در این زمینه، ارائه‌ی ابزار در چند مرحله است، که سبب می‌شود، تا به جای یک سیستم بزرگ یکپارچه، چند سیستم کوچک‌تر اما با استقلال نسبی از هم عرضه شوند و تغییرات آینده‌ی سیستم کل را تسهیل بخشند.

۵-۳- مرحله‌ی اول

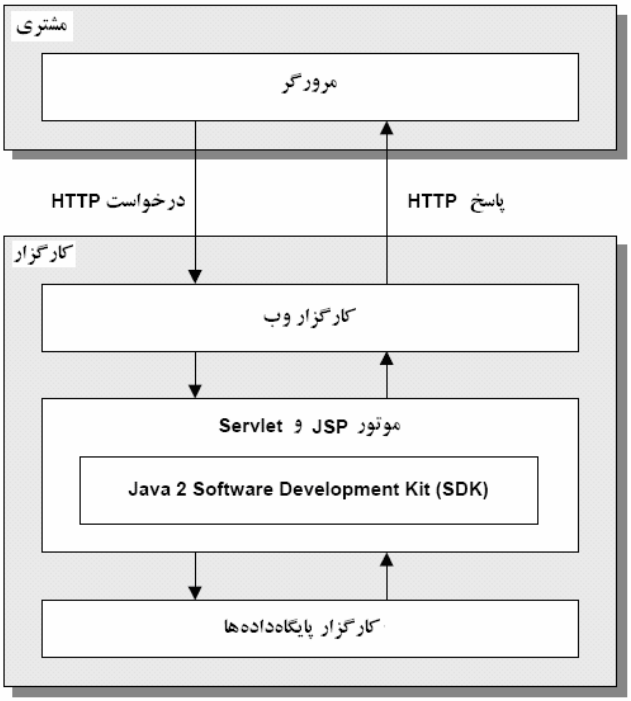
در این مرحله کد مبنای سیستم خوانده و تجزیه و تحلیل می‌شود. سپس قطعات استخراجی مورد پردازش قرار می‌گیرد و انواع مختلف ارتباطات بین آن‌ها محاسبه می‌شود و در یک قالب میانی ذخیره می‌شود که در واقع بیانگر گراف چندگانه‌ای است که در مرحله‌ی بعد مورد استفاده قرار خواهد گرفت.

این مرحله در واقع یک تجزیه‌گر جامع می‌سازد که فایل‌های متنوع موجود در برنامه‌کاربردی را برای استخراج قالب میانی بررسی می‌کند. امر بی‌شک بدون وابستگی از زبان و بستر پیاده‌سازی نخواهد بود. در این مستند، زبان JSP به عنوان گزینه‌ی مورد بررسی، انتخاب گردیده است.

۵-۳-۱- مقدمه‌ای بر زبان JSP

در اوایل عرضه شدن جاوا، قابلیت اپلت‌ها که توانایی اجرا در مرورگرهای مختلف را داشتند، سبب شد تا فعالیت زیادی برای ایجاد برنامه‌های کاربردی مبتنی بر وب انجام گیرد. که این امر منجر به ایجاد JSP[2] و سرولت‌ها گردیدکه در عمل، به واسطه‌ی پیچیدگی کمتر JSP، برنامه‌نویسان به سمت آن تمایل پیدا کردند.

[1] Scalability
[2] Java Server Page



شکل ۵-۱ اجزای یک برنامه‌کاربردی مبتنی بر وب با بستر Java

برنامه‌های کاربردی مبتنی بر وب با فن‌آوری جاوا برای اجرای خود نیاز به برنامه‌هایی با نام موتور[1] JSP و سرولت دارند تا تبادل آن‌ها با کارگزار وب را میسر و مفهوم سازد. نحوه‌ی رفتار کارگزار وب با موتور JSP توسط J2EE[2] تعیین می‌گردد. اجزای ذکر شده برای یک برنامه‌ی کاربردی مبتنی بر وب با مشخصات فوق در شکل ۵-۱ آمده است. تام‌کت[3] یکی از موتورهای کدباز سرولت و JSP است، که که محصولی از پروژه‌ی جاکارتا[4] از شرکت آپاچی[5] است و برنامه‌های تحت وب مورد آزمایش در این مستند نیز بر اساس آن تهیه شده‌اند.

در رابطه با سرولت نیز ذکر این نکته کافیست که روشی برای نوشتن برنامه‌های Javaی جانب کارگزار است که یکی از استفاده‌های معمول آن ایجاد پویای صفحات وب است. سرولت‌های بر کارگزار قرار می‌گیرند و منتظر می‌شوند تا درخواستی فرابرسد، سپس بر اساس آن کاری انجام می‌دهند و نتیجه را در

[1] Engine
[2] Sun's Java 2 Platform, Enterprise Edition
[3] Tomcat
[4] Jakarta
[5] Apache

قالب یک صفحه‌ی وب، برمی‌گردانند. پیچیدگی زیاد سرولت‌ها که درک و ایجاد آن‌ها را دشوار می‌نماید، سبب شد تا JSP به عرصه وارد گردد.

نحوه‌ی تبادل موتور JSP با کارگزار را می‌توان در نه مرحله‌ی زیر خلاصه نمود:

۱. کاربر به یک وب‌سایت می‌رود و استفاده از آن را توسط یک صفحه‌ی JSP (دارای پسوند jsp.) شروع می‌کند. مرورگر وب وی، درخواست را از طریق پروتکل HTTP بر بستر اینترنت ارسال می‌نماید.

۲. درخواست JSP به کارگزار وب داده می‌شود.

۳. کارگزار وب تشخیص می‌دهد که فایل مورد درخواست از نوع JSP است، پس آن را به موتور JSP و سرولت ارجاع می‌دهد.

۴. اگر فایل JSP برای اولین بار صدا زده شده باشد، بایستی تجزیه (Pars) گردد، وگرنه کنترل به مرحله ۷ انتقال داده شود.

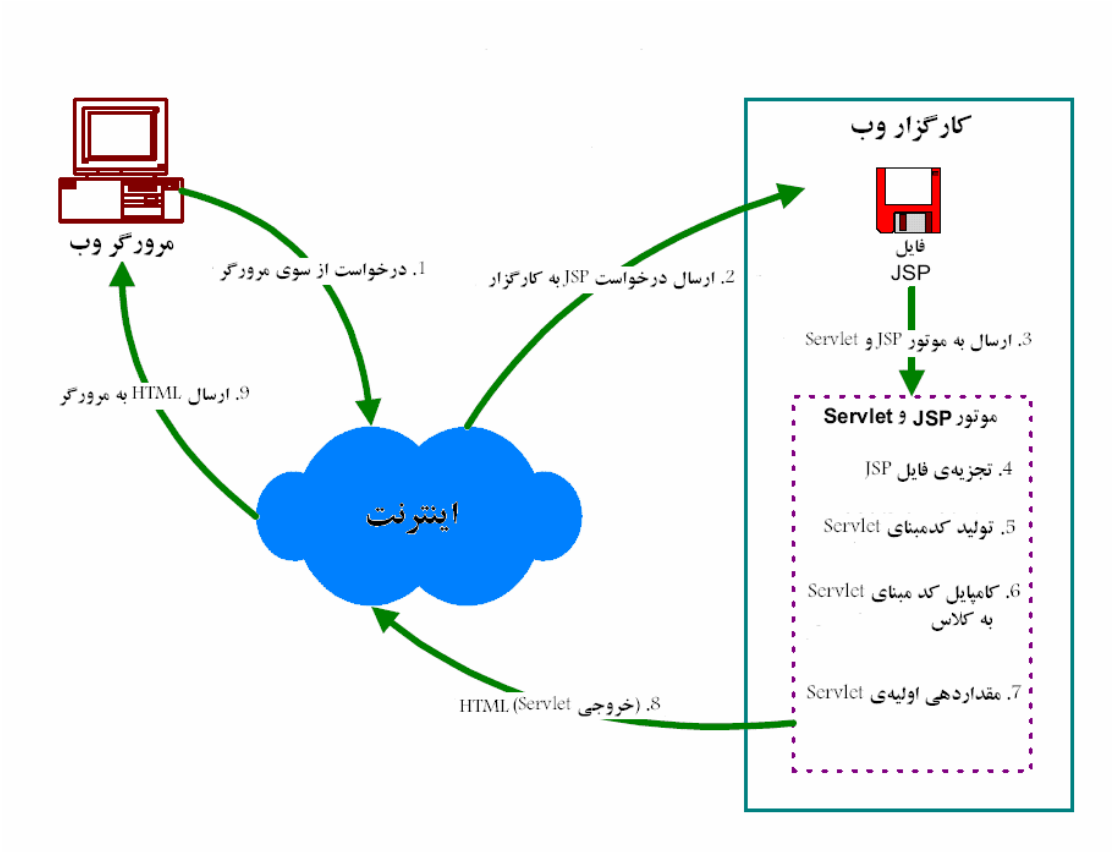
۵. تولید یک سرولت ویژه از فایل JSP، عملکرد مرحله‌ی کنونی است. تمام HTMLهای لازم به دستورات println تبدیل می‌شوند.

۶. کدمبنای سرولت به یک کلاس کامپایل می‌شود.

۷. با استفاده از فراخوانی متدهای init و service، نمونه‌ای از سرولت ایجاد می‌گردد.

۸. خروجی سرولت که یک HTML است، از طریق اینترنت به مرورگر درخواست کننده ارسال می‌شود.

۹. نتایج HTMLیی بر مرورگر کاربر نمایش داده می‌شوند.



شکل ۵-۲ معماری JSP

با توجه به اینکه در اصل بستر جاوا برای سیستم‌فایل‌های سازگار با یونیکس تطابق یافته است، نحوه‌ی جای‌دهی فایل‌های یک برنامه‌کاربردی مبتنی بر وب، در کارگزار، در حالت کلی، به شکل زیر است:

Directory	Contents
appName	JSP, HTML, and image files
appName/WEB-INF/classes	classes accessed by JSP files

شکل ۵-۳ نحوه‌ی استقرار فایل‌های یک برنامه‌ی کاربردی تحت وب مبتنی بر JSP

یعنی پس از نصب Tomcat در پوشه‌ی webapp آن که دارای آدرس TOMCAT_HOME/webapps است، پوشه‌ای به نام *appName* ایجاد می‌نماییم و فایل‌های برنامه‌ی کاربردی مبتنی بر وب را در آنجا خواهیم گذاشت.

دستورات JSP در قالب برچسب‌هایی، مشابه با برچسب‌های HTML و همراه با آن‌ها، در فایل‌هایی با پسوند، jsp عرضه می‌شوند. برای آشنایی بیشتر، یک نمونه صفحه‌ی برنامه‌ی ساده‌ی JSP در زیر آمده است:

```
<%@ page import="hello.NameHandler" %>
<jsp:useBean id="abean" scope="page" class="hello.NameHandler" />
<jsp:setProperty name="abean" property="*" />

<html>
        <head>
                <title>Hello, User</title>
        </head>

<%@ include file="dukebanner.html" %>

<table border="0" width="700">
<tr>
<td width="150">   </td>
</tr>
<tr>
<td width="150"   </td>
<td width="550">

<form method="get">
 <input type="text" name="username" size="25"> <br>
 <input type="submit" value="Submit">
 <input type="reset" value="Reset">
</td>
</tr>
</form>

</table>
        <%  if ( request.getParameter("username") != null ) {    %>
        <%@ include file="response.jsp" %>
        <%
                }
        %>
</body>
</html>
```

شکل ۵-۴ یک قطعه کد نمونه‌ی JSP

۵-۳-۲- استخراج روابط

استخراج روابط بر اساس مدلی که برای برنامه‌کاربردی مبتنی بر وب انتخاب گردید، صورت خواهد پذیرفت. با توجه به اینکه زبان JSP به عنوان گزینه‌ی مورد بررسی، انتخاب گردیده شده است بایستی

روابط را از فایل‌های JSP و HTML استخراج کرد. البته باید توجه داشت که براساس چهارچوب‌های متعددی که برای برنامه‌های کاربردی مبتنی بر JSP عرضه می‌گردد، حالات مختلفی ممکن است حتی برای روابط ساده‌ی بین آن‌ها، پیش بیاید. برای مثال، در چهارچوب کد بازی[1] به نام Struts که محصولی از پروژه‌ی جاکارتا از شرکت آپاچی است و به صورت فراگیری برای ایجاد برنامه‌های کاربردی مبتنی بر وب مورد استفاده قرار می‌گیرد، برچسب‌هایی[2] مانند زیر به عنوان Link به کار می‌روند:

```
<html:link page="/logon.jsp">
```

لحاظ نمودن چنین پیچیدگی‌هایی علاوه بر اینکه عمومیت مسأله را کمرنگ‌تر می‌کند، نیاز به زمان و دانش بسیار زیادی دارد. بدین ترتیب، سعی شده است تا قالب ساده و با پیچیدگی‌های کمتر و عمومیت بیشتری برای این مرحله در نظر گرفته شود. زیرا همانطور که می‌دانیم، در این مرحله که وابسته با زبان و بستر پیاده‌سازی است، زمان بسیار زیادی می‌تواند صرف گردد و تجزیه‌گر جامعی می‌تواند ایجاد شود. بر این اساس، رئوس استخراج روابط در زیر بیان شده است [۲۸,۲۷] :

Link: رابطه‌ی Link معمولاً در صفحات به صورت زیر ظاهر می‌شود:

```
<a href="argument"> text </a>
```

که در آن argument به صفحه‌ای دیگر (یا به بیان کلی‌تر به فایل،http, news, gopher, telnet و یا ftp) اشاره دارد و text متنی است که به ازای آن صفحه نمایش داده می‌شود.

البته فرم دیگری از پیوند وجود دارد که استفاده‌ی کمتری دارد و تنها مجاز است، در بالای مستند بیاید و یک مثال از آن به شکل زیر است:

```
<HEAD>
...other head information...
<TITLE>Chapter 5</TITLE>
    <LINK rel="prev" href="chapter4.html">
    <LINK rel="next" href="chapter6.html">
</HEAD>
```

Submit: این رابطه در HTML در برچسب <form> واقع می‌گردد. یک فرم در HTML دارای سه قسمت عمده است: برچسب‌های باز و بسته‌ی <form>، عناصر ورود اطلاعات و کلید submit که داده را به کارگزار ارسال می‌نماید. در یک فرم معمول HTML، برچسب <form> ورودی، شکلی مشابه به زیر دارد:

---

[1] Open Source
[2] Tags

<form method=get **action**=*"someURL"*>

اما در برنامه‌های JSP، خصیصه‌ی action، تعیین کننده‌ی قطعه یا فایل JSPیی است که داده‌ی ورودی کاربر در فرم را دریافت می‌کند. اگر برنامه‌نویس خواسته باشد، داده توسط شئیی که در برچسب <jsp:useBean> قرار دارد، پردازش شود، ممکن است خصیصه‌ی action ذکر نگردد. سایر بخش‌های فرم در درست مانند HTML استاندارد ایجاد می‌گردد، به عنوان مثال می‌توان ورودی زیر را در نظر گرفت:

<input type = "text" name="username">

Include: این برچسب امکان استفاده از سایر فایل‌ها در صفحه‌ی JSP را میسر می‌سازد. نحو آن مشابه به یکی از موارد زیر است. با این شرح که مورد اول شمول را در زمان کامپایل انجام می‌دهد، اما دومی شمول را در زمان اجرا عملی می‌کند.

```
<%@ include file="myFile.html" %>
<jsp:include page="relative URL" flush="true" />
```

Redirect: این برچسب امکان ارسال[1] درخواست به صفحه‌ی دیگری را میسر می‌سازد و در JSP تنها دارای یک خصیصه به نام page است، که آدرس مقصد را دربردارد. این آدرس ممکن است یک مقدار ایستا باشد، یا اینکه در زمان اجرا ایجاد شود. دو مثال از این مورد در زیر آمده است:

```
<jsp:forward page="/utils/errorReporter.jsp" />
<jsp:forward page="<%= someJavaExpression %>" />
```

اما در HTML ممکن است از بر چسب META http-equiv برای این امر استفاده شود. یعنی به عنوان مثال، خط زیر سبب خواهد شد که مرورگر صفحه‌ی کنونی را پس از ۲ ثانیه به آدرس www.yahoo.com تغییر دهد.

```
<META http-equiv="refresh" content="2,http://www.yahoo.com">
```

Frameset: قاب‌ها که اجازه‌ی تقسیم صفحه به ناحیه‌های مجزا را می‌دهند، اجازه‌ی بار شدن صفحه در یک قاب بدون تغییر محتوی سایر قاب‌ها را می‌دهند.

```
<frameset rows="30%,*">
    <frame name="banner" src="banner.html" marginwidth"5" marginheight="5">
    <frameset cols="30,*">
        <frame name="NAV" src="navigation.html" frameborder="0">
        <frame name="MAIN" src="main.html" scrolling="yes" noresize>
    </frameset>
</frameset>
```

[1] Forward

برچسب مشابهی برای بارگذاری آدرسی دیگر در یک قاب در صفحه‌ی کنونی وجود دارد، که به صورت زیر نوشته می‌شود:

```
<iframe name="localqute" src="http:\\www.yahoo.com">
```

۵-۳-۳- پیاده‌سازی

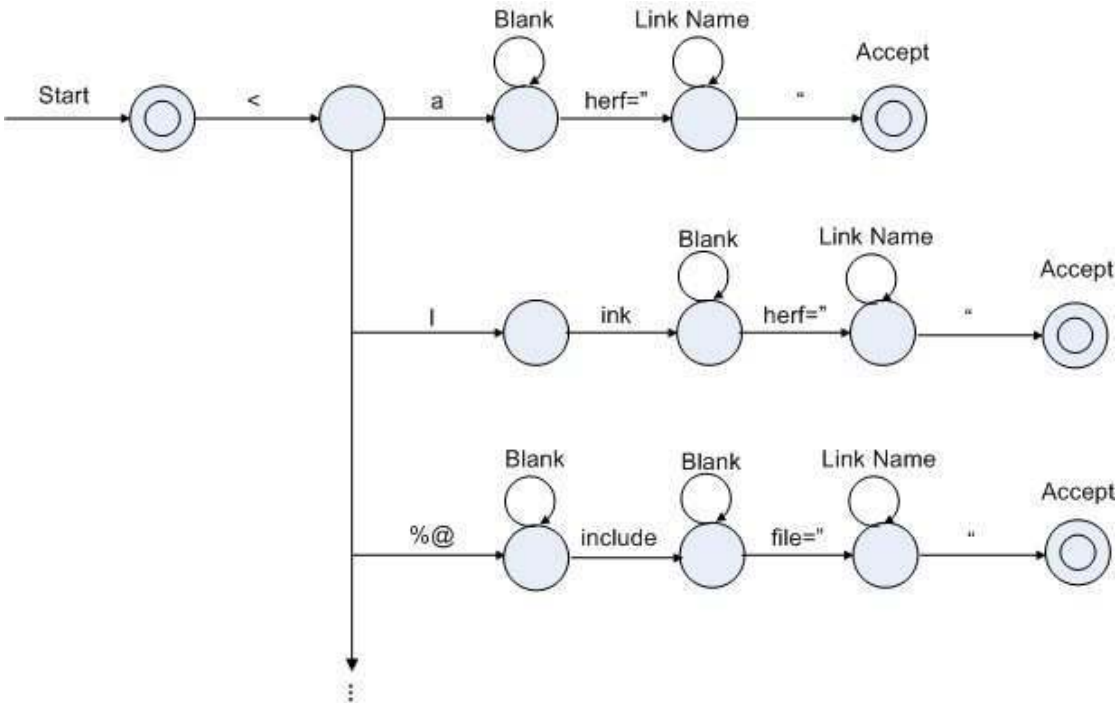
ابزار مورد نظر در اینجا در کلاس FirstStep پیاده‌سازی شده است. این کلاس مسوول ایجاد گراف‌چندگانه از یک برنامه‌کاربردی مورد نظر است. برای این منظور بایستی تمام فایل‌های موجود در برنامه‌کاربردی مبتنی بر وب، یک بار خوانده شوند و روابط موجود در هر یک از آن‌ها، با توجه به نکاتی که در بخش پیشین درباره‌ی تشخیص انواع روابط بیان شد، تشخیص داده شود و ثبت گردد. فایل‌های ایجاد شده و مورد استفاده در این بخش عبارتند از:

جدول ۵-۱ فایل‌های مورد استفاده در مراحل مختلف فرایند مهندسی معکوس

نام فایل	Files.txt	MultipleGraph.txt	Clusters.txt
شرح استفاده	انتساب یک شماره به هر خوشه (فایل) ابتدایی	نگهداری ساختار گراف چندگانه	نگهداری شماره‌های متناظر با فایل‌های هر خوشه در یک سطر
ساختار تقریبی داخلی	index.htm login.jsp aboutus.jsp …	4-s 14-L 2-L 3-L 9-S 9-L 11-L 1-L …	1,3,5 17,86,14 4 …

جهت تسهیل پیچیدگی و افزایش سرعت در تشخیص کلمات کلیدی تشریح شده در بخش پیش، از نمودار انتقال معین[1] که در ساخت تحلیل گر لغوی برای کامپایلرها کاربرد دارد[۳۵]، استفاده شده است. نمودار مفهومی الگوریتم تشخیص پیوندها و یکی از شمول‌ها برای نمونه در زیر آمده است:

[1] Deterministic Transition Diagram



شکل ۵-۵ نمودار انتقال تشخیص روابط

بر این اساس شبه‌کد[1] عملکرد اصلی کلاس FirstStep در زیر آمده است:

```
String FileName;
Int FileNumber=0;
Character char;

For each file with (.html or .htm or .jsp) extension
    Write (FileNumber,FileName) in File.txt
    Convert_to_LowerCase(File.txt);
    Char = read_a_char(FileName);

    While (char != End_of_File){

          While (Char != "<");
          Char = read_next_char(FileName);

          If (Char="a")
                Anchor();

          Else if (Char= "l")
                Link();

          Else if (Char= "m")
                HTMLRedirect();

          Else if (Char = "i")
                Iframe();

          Else if (Char= "f")
```

[1] Pseudocode

```
                Char = read_next_char(FileName);
                If (Char= "o")
                      HTMLForm();
                Else if (Char= "r")
                      FrameSet();

          Else if …

          …
    }
```

که هر یک از متدها مانند Anchor()، Link()، Iframe() و ... براساس نحو بیان شده در ۷-۲-۲ و نمودار انتقال تصویر ۷-۵، رابطه‌ی موجود و فایل مرتبط به آن را می‌یابند و به فایلی به نام MulipleGraph.txt می‌افزایند، تا برای استفاده‌ی مرحله ی بعد، آماده گردد. موجودیت‌ها در فایل MulipleGraph.txt به شکل زوج‌های FileNumber-RelationType که با فاصله از هم جدا شده‌اند، ذخیره می‌شوند. هر سطر در این فایل نماینده‌ی یک فایل بررسی شده‌ی برنامه‌ی کاربردی مبتنی بر وب است. به عبارت دیگر شماره سطرها در این گراف متناظر با شماره‌هایی هستند که به نام فایل‌های برنامه‌ی کاربردی تحت وب بررسی شده در فایل File.txt منتسب شده‌اند. فایل MulipleGraph.txt همانطور که از نامش پیداست، ساختار گراف چندگانه را نگهداری می‌کند و به عنوان ورودی مرحله‌ی پیش به شمار می‌آید.

همچنین لازم به ذکر است، جهت کاهش خطاهای تشخیص کلمات کلیدی، در همان ابتدای ریختن محتوی فایل‌های برنامه‌ی کاربردی مبتنی بر وب در File.txt، تمام حروف آن‌ها را به کمک کلاس Convert_to_LowerCase حروف کوچک تبدیل می‌کنیم.

Relation Type تنها همان سه نوع عمده‌ای است که در فصل پیش در نظر گرفته شده بود(Submit, Redirect و Link) برای سایر روابط به علت بدیهی بودن، در همین مرحله خوشه‌بندی انجام می‌گیرد، شبه‌کد آن در زیر آمده است:

```
For Each Include Relation
   Cluster Included file with this file

For Each Frame in A FrameSet
   Set a Link Relation between MainPage and Page related to the Frame
```

بدین ترتیب فایل Clusters.txt نیز برای نگهداری شماره‌ی فایل‌های موجود در هر خوشه، در یک سطر و برای استفاده در مراحل آخر (از جمله مرحله‌ی توزیع)، لازم است.

نکته‌ی آخری که ذکر آن ضروری است، آن است که در مرحله‌ی آخر از فرایند مهندسی معکوس که توزیع بار برنامه‌ی کاربردی مبتنی بر وب براساس خوشه‌های ایجاد شده است، لازم است برخی متغییرهای

سراسری در کد مبنا یافت شود، که می‌توان با افزودن بخش کوچکی به فرایند این مرحله این امر را امکان‌پذیر کرد. آنچه باید یافت شود، کلاس‌های ایستایی است که احتمالاً در کد JSP تعریف شده باشند. این موضوع نیز باتوجه به نیازهای مرحله‌ی آخر، در پیاده‌سازی لحاظ شده است.

کد مبنای این مرحله در پیوست ۲ آمده است.

۵-۴- مرحله‌ی دوم

در این مرحله ابتدا باید ساختار گراف چندگانه به ساختار گراف ساده تبدیل گردد، بدین ترتیب لازم است که فایل بیانگر گراف چندگانه با استفاده از الگوریتم زیر تبدیل گردد و ساختاری ایجاد شود که بتوان آن را در یک ماتریس دو بعدی ریخت که توسط الگوریتم خوشه‌بندی مورد استفاده قرار گیرد. به این ترتیب یک فایل متنی به نام SimpleGraph.txt که در آن فقط عدد وجود دارد و سطرها و ستون‌ها بیانگر خوشه‌ها هستند و ستون‌ها با علامت فاصله از هم جدا شده‌اند، ایجاد خواهد شد.

در این مرحله دو کلاس به نام‌های Get_Value و ComputeSimpleGraph برای مقداردهی به روابط Link، Submit و Redirect مورد استفاده قرار می‌گیرند. تابع Get_Value در راستای عمومی نمودن فرآیند، امکان تغییر مقدارهای انتخابی برای روابط مذکور را فراهم می‌کند و کد آن به سادگی الگوریتم زیر است:

```
For Each L in MulipleGraph.txt Replace L with 1
For Each R in MulipleGraph.txt Replace R with 2.4
For Each S in MulipleGraph.txt Replace S with 3
```

الگوریتم ComputeSimpleGraph نیز که مقدار برچسب ساده‌ی بین گره‌ها در گراف ساده را محاسبه می‌نماید (رابطه‌ی ۱ در بخش ۶-۴) به شرح زیر است:

```
For each Line in MulipleGraph.txt

    For any (FileNumber,RelationType ) that FileNumber is the Same

        NewRelationType =  Sum of RelationTypes

            Input NewRelationType in its proper place
            in a Temporary Two Dimantial Array

Write Temporary Array in SimpleGraph.txt File
```

کد مبنای این مرحله در پیوست ۳ آمده است.

۵-۵- مرحله‌ی سوم

از الگوریتم خوشه‌بندی سلسله‌مراتبی در مرحله‌ی سوم برای ایجاد خوشه‌ها استفاده خواهد شد. در این‌جا به ازای هر بار اجرای خوشه‌بندی، تعداد خوشه‌ها یکی کمتر خواهد شد. نتایج هر مرحله خوشه‌بندی در فایلی به نام $SimpleGraph_i.txt$ که i تعداد خوشه‌های موجود در فایل کنونی است و این فایل برای خوشه‌بندی مرحله‌ی بعد به کار می‌رود. الگوریتم سطح بالای خوشه‌بندی در زیر آمده است:

```
1: Convert SimpleGraph.txt Name to SimpleGraph_n.txt that n is Current Number of Clusters
2: Get Num_Of_Clusters from User
3: For (i=n; Num_of_Clusters;i--)
4:       Write SimpleGraph.txt in a 2D Array named Temp
5:       Find Greatest Number in Temp
6:       Merge Two Nearest Cluster Based on previous step
7:       Update Temp Array
8:       Update Clusters.txt File
9:       Write Temp Array in SimpleGraph_i-1.txt
10: Based On Clusters.txt and Files.txt
         Print Each Cluster's included Files
```

در سطر ۱ تنها یک تغییر نام برای هماهنگی با الگوریتم صورت می‌گیرد.

در سطر ۲ تعداد خوشه‌ها که در واقع تعداد کارگزارنی است که قرار است، برنامه‌کاربردی ما بر آن‌ها توزیع گردد، از کاربر درخواست می‌شود. این یکی از مراحلی است که به تبادل با کاربر نیاز دارد.

در سطر ۴ برای تسهیل انجام محاسبات، فایل با تعداد i خوشه در یک آرایه iتایی دو بعدی ریخته می‌شود

در سطر ۵ دو خوشه که بیش از همه به هم شبیه‌اند، یافت می‌شود

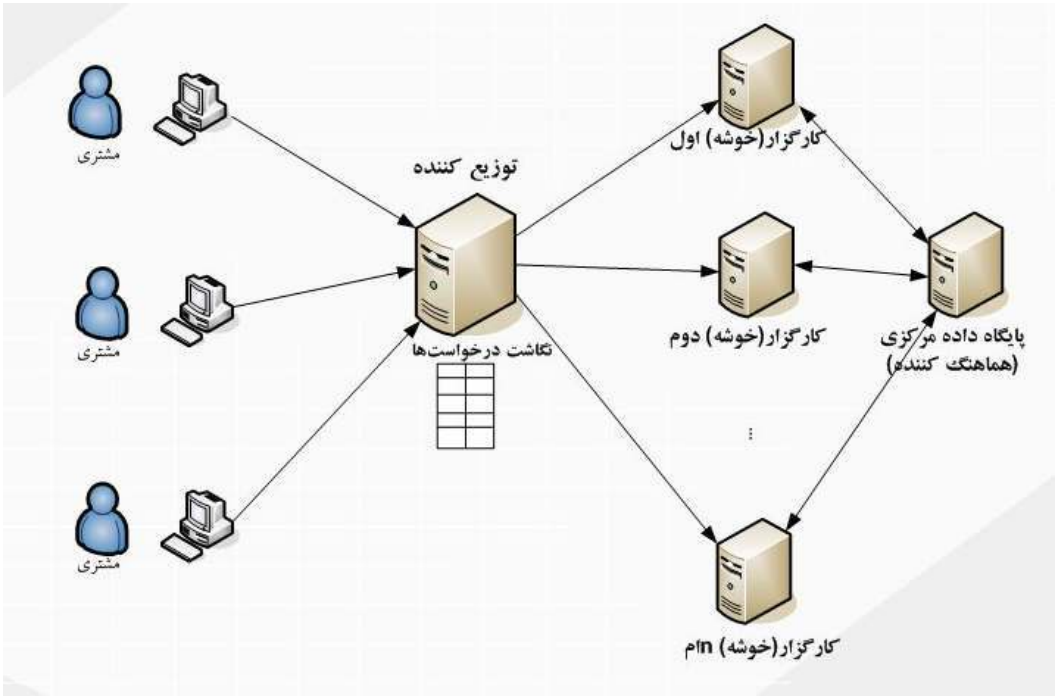
در سطرهای ۶ و ۷ براساس قاعده‌ی به‌هنگام‌سازی که ما آن را از نوع پیوند ساده انتخاب کرده‌ایم، یک واحد از تعداد سطر و ستون‌های آرایه کم می‌شود و مقادیر سایر سطر و ستون‌ها بر اساس ادغام دو خوشه‌ی شبیه به هم، به‌هنگام‌سازی می‌شوند

در سطر ۸ فایل `Clusters.txt` که شماره‌ی فایل‌های موجود در هر خوشه را نگهداری می‌کند، به هنگام می‌شود

در سطر۹ لیست نهایی فایل‌های موجود در هر خوشه، بر اساس اطلاعات موجود در فایل‌های مرحله‌ی اول چاپ می‌شود. کد مبنای این مرحله نیز در پیوست ۴ آمده است.

۵-۶- مرحله‌ی چهارم

پیاده‌سازی در این مرحله بر دو بخش مبتنی است، اول پیاده‌سازی بخشی از کد، که برای توزیع کننده[1] در نظر گرفته شده است و با استفاده از نتایج مرحله‌ی پیش، درخواست‌ها به صفحات وب را تشخیص می‌دهد و به کارگزار (خوشه) مربوطه ارسال می‌دارد. دوم آن قسمت از کد مبنا که بایستی به جای کدی که هم‌اکنون به متغییر سراسری دسترسی دارد، جایگزین گردد. که این موضوع خود مشتمل بر پیاده‌سازی جانب کارگزاران (خوشه‌ها) و کارگزار هماهنگ‌کننده (که متغییر سراسری در حافظه‌ی اصلی آن نگهداری می‌شود) است، که ما در اینجا مکان آن را از لحاظ فیزیکی، همان کارگزار پایگاه داده‌ها تعیین کرده‌ایم. براین اساس پیکربندی فیزیکی جدید پیشنهادی برای توزیع بارکاری برنامه‌کاربردی مورد بررسی در زیر آمده است:



شکل ۵-۶ پیکربندی فیزیکی بستر توزیع بارکاری

همانطور که در فصل پیش بیان شد، لازم است که یک نسخه‌ی کامل از کد مبنا بر روی هر یک از کارگزاران وب موجود در مزرعه‌ی کارگزاران کپی شود و در صورت مجزا نبودن پایگاه داده‌ها، آن را به کارگزار مجزایی منتقل نمود.

[1] Dispatcher

۵-۶-۱- توزیع درخواست‌ها

با توجه به زبان انتخابی برای بررسی برنامه‌های کاربردی مبتنی بر وب، در این فصل، از امکانات همان زبان برای توزیع درخواست‌ها به کارگزاران استفاده شده است. در زبان JSP امکانی به نام صافی[1] برای تغییر درخواست‌های HTTP لحاظ گردیده است که با کمک آن می‌توان درخواست‌ها را به کارگزار دیگری منتقل نمود. بدین منظور ابتدا لازم است که صفحه‌ی اول وب‌سایت بر کارگزار توزیع‌کننده قرار گیرد و سپس براساس سرآیند درخواست‌های HTTP و داشتن جدول نگاشت خوشه‌ها به کارگزارها و با کمک امکان فیلتر در زبان JSP، درخواست‌ها را توزیع نمود. علت قرار دادن صفحه‌ی اول در کارگزار توزیع کننده آن است که به طور معمول بیشتر درخواست‌ها برای صفحات یک وب‌سایت، از صفحه‌ی اول آن آغاز می‌گردد و ما مایل نیستیم که ترافیک صفحه‌ی اول بر خوشه‌ای تحمیل گردد که مشتمل بر آن است، پس تنها صفحه‌ی اول را بر کارگزار توزیع کننده قرار خواهیم داد.

دستیابی به اطلاعات یک درخواست از طریق دسترسی به سرآیندهای بسته‌ی HTTP میسر می‌شود. این سرآیندها متشکل از نام سرآیند و مقدار آن هستند. سرآیندها پیش از بدنه‌ی پیامی که مشتری به کاربر ارسال می‌کند، فرستاده می‌شوند. می‌توان از یک صافی برای تغییر سرآیند درخواست استفاده نمود.

برای این منظور از کلاس javax.servlet.http.HttpServletRequestWrapper استفاده خواهیم کرد که قابلیت‌های اضافه‌ای برای کار با درخواست‌های HTTP در اختیار می‌گذارد. مراحل کار به شرح زیر است[۲۹]:

۱. کلاسی ایجاد خواهیم کرد که HttpServletRequestWrapper را Extened می‌کند.

۲. کلاس را در فهرست *WEB-INF/classes* کپی می‌کنیم.

۳. کلاسی ایجاد خواهیم کرد که از javax.servlet.Filter استفاده می‌کند. این کلاس از کلاس Request Wrapper استفاده خواهد کرد تا پارامتر ServletRequest از متد Filter.doFilter( ) را استفاده نماید.

۴. ثبت صافی در web.xml مرحله‌ی آخر کار است. به این ترتیب صافی به تمام درخواست‌های برنامه‌کاربردی مبتنی بر وب اعمال خواهد شد.

کد نمونه‌ی صافی در زیر آمده است:

```
package com;

import javax.servlet.*;
import javax.servlet.http.*;
```

[1] Filter

```
  public class RequestFilter implements Filter {

      private FilterConfig config;

      /** Creates new RequestFilter */
    public RequestFilter( ) {}

    public void  init(FilterConfig filterConfig)  throws
ServletException{

        this.config = filterConfig;
    }

  public void  doFilter(ServletRequest request,
    ServletResponse response, FilterChain chain) throws
java.io.IOException,
      ServletException {

      ReqWrapper wrapper = null;
      ServletContext context = null;

      //create the request wrapper object, an instance of the
      //ReqWrapper class. The client request is passed into
      //ReqWrapper's constructor

      if (request instanceof HttpServletRequest)
          wrapper = new ReqWrapper((HttpServletRequest)request);

      //use the ServletContext.log method to log param names/values

      if (wrapper != null){
          context = config.getServletContext( );

      context.log("Query: " + wrapper.getQueryString( ));}


      //continue the request, response to next filter or servlet
      //destination

      if (wrapper != null)
          chain.doFilter(wrapper,response);
      else
          chain.doFilter(request,response);

    }//doFilter

    public void destroy( ){

    /*called before the Filter instance is removed
          from service by the web container*/
     }//destroy
  }
```

اعمال فیلتر نیز توسط کدی مشابه به زیر انجام می‌گیرد، که البته باید این کد را به فایل web.xml اضافه نمود:

```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <!DOCTYPE web-app
      PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
```

```
            "http://java.sun.com/dtd/web-application_2_3.dtd"
>

<web-app>

<!-- register the filter -->

<filter>
    <filter-name>LogFilter</filter-name>
    <filter-class>com.LogFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>LogFilter</filter-name>
    <url-pattern>/requestheaders</url-pattern>
</filter-mapping>

<!-- register the servlet to which the filter is mapped -->

<servlet>
    <servlet-name>requestheaders</servlet-name>
    <servlet-class>com.RequestHeaderView</servlet-class>
</servlet>

<!-- Here is the URL mapping for the requestheaders servlet -->

<servlet-mapping>
    <servlet-name>requestheaders</servlet-name>
    <url-pattern>/requestheaders</url-pattern>
</servlet-mapping>

</web-app>
<!-- top of web.xml deployment descriptor -->

<filter>
    <filter-name>LogFilter</filter-name>
    <filter-class>com.LogFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>LogFilter</filter-name>
    <url-pattern>/displayHeaders.jsp</url-pattern>
</filter-mapping>

<!-- rest of deployment descriptor -->
```

۵-۶-۲- حل مسأله‌ی متغییرهای مشترک سراسری

برای این منظور بایستی از تمامی متغییرهای سراسری که در برنامه کاربردی مبتنی بر وب وجود دارد، مطلع بود. شیوه‌هایی که می‌توان متغییرهای سراسری تعریف نمود، به زبان برنامه‌نویسی نیز وابسته است. این امر در JSP و سرولت به دو نحو امکان‌پذیر است:

۱. متغییر Static

۲. متغییر Application که توسط تمام سرولت‌ها و صفحات JSP در برنامه‌کاربردی مبتنی بر وب به اشتراک گذاشته می‌شود. موسوم به ServletContext است که می‌توان توسط getServletConfig().getContext() به آن دست یافت.

گزینه‌ی دوم مورد نظر ما نخواهد بود. زیرا ما برنامه‌های JSP را برای بررسی انتخاب کرده‌ایم، اما راه‌کار آن نیز مشابه به راه‌کار مورد اول است. برای ایجاد هماهنگی در دسترسی به متغییرهای مشترک سراسری ابتدا باید آن‌ها را یافت. ممکن است، موارد فوق هم به صورت صریح و هم به صورت ضمنی در کدمبنای برنامه‌ی کاربردی مبتنی بر وب ظاهر شوند، شیوه‌ی صریح یعنی این‌که برنامه‌نویس تعریف کلاس ایستا را مستقیماً در کد JSP انجام دهد، کمتر محتمل است و شیوه‌ی ضمنی تعریف متغییرهای ایستا یعنی آن‌که برنامه‌نویس این امر را در یک Java Bean که مجموعه‌ای از کلاس‌های کامپایل شده است و توسط صفحات JSP مورد استفاده قرار می‌گیرد، انجام داده باشد، بیشتر کاربرد دارد. به این ترتیب با توجه به ساختار تجزیه‌گری که در مرحله‌ی اول ایجاد گردید، بهتر است یافتن آن را به برنامه‌نویس یا نگهدارنده‌ی سیستم ارجاع داد. با این مقدمه، دو رویکرد برای یافتن متغییرهای مشترک سراسری پیشنهاد می‌شود:

۱. افزودن بخشی به تجزیه‌گر طراحی شده در مرحله‌ی اول فرآیند مهندسی معکوس، برای یافتن عبارات Static

۲. استفاده از دانش قلمرو برنامه و برنامه‌نویسان و نگهداران و کاربران، برای یافتن متغییرهای سراسری

حال با فرض اینکه متغییر یا متغییرهای سراسری در کدمبنا یافت شد، فرآیند دو مرحله‌ای زیر برای پیاده‌سازی انجام خواهد پذیرفت:

۱. قرار دادن کدی برای هماهنگی استفاده از متغییر سراسری بر کارگزار هماهنگ کننده

۲. جایگزینی کدمبنای کارگزاران ارجاع دهنده به متغییر سراسری با کد جدید

جزییات پیاده‌سازی این بخش به زبان ASP در مرجع [۳۳] آمده است و با توجه به اینکه نحوه‌ی توزیع بارکاری با رویکرد جدید کاملاً تشریح و پیاده‌سازی شده است، بازسازی آن برنامه به زبان JSP ضروری دیده نمی‌شود، اما با توجه به امکانات خاص زبان JSP در این‌باره، به ذکر چند نکته اکتفا می‌گردد. اول اینکه ساختاری که بر کارگزار مبنا قرار داده‌می‌شود بایستی امکان دسترسی همزمان و امن را فراهم سازد، بدین منظور لازم است خط زیر به بالای صفحه‌ای که دسترسی‌ها به آن ارجاع داده می‌شود، اضافه شود:

```
<%@ page isThreadSafe="true" %>
```

این امر سبب می‌گردد، تا برای هر دسترسی یک نخ[1] اختصاص یابد و مسأله‌ی هماهنگی بین نخ‌ها به برنامه‌کارگزار وب واگذار گردد و نیاز به دخالت برنامه‌نویس نداشته باشد.

اما متغییری که اکنون در کارگزار وب توزیع‌کننده جزوه متغییرهای سراسری است با ایده‌ای مشابه به زیر، به سادگی قابل تعریف و استفاده است:

```
<%
synchronized (application) {
SharedObject CommonVariable = (SharedObject)
application.getAttribute("sharedObject");
CommonVariable.update(someValue);
application.setAttribute("sharedObject", CommonVariable);
}
%>
```

حال برنامه‌های مشتری با دسترسی به شیء مشترک کار پیشین خود را از سر خواهند گرفت. کدی نیز که در برنامه‌ی مشتری قرار می‌گیرد و امکان دسترسی به کارگزار را فراهم می‌سازد از Socket Programming استفاده خواهد نمود که به دلیل وابستگی به پشته‌ی پروتکلی شبکه برای تمام زبان‌های پیاده‌سازی مشابه است[۳۳].

[1] Thread

۵-۷- بررسی معیارهای ارزیابی

با توجه به موارد ذکر شده در بخش دوم فصل پیش برای ارزیابی میزان کارایی رویکرد بایستی به معیارهایی که از ابتدای ایجاد آن در نظر گرفته شده بود، رجوع کرد. این معیارها عبارت بودند از عمومیت، خودکار سازی و قابلیت توسعه.

- **عمومیت:** درباب عمومیت رویکرد و ابزار پیاده‌سازی توجه به این دو نکته ضروری است که
  - ✓ تنها بخش فرآیند که وابسته به قلمرو مسأله و زبان پیاده‌سازی است، همان مرحله‌ی اول است که عملیات تجزیه و تحلیل کد مبنا را بر عهده دارد و تنها مرحله توزیع درخواست‌ها، آن‌هم به میزان بسیار کمی (در حد یافتن متغیرهای مشترک سراسری) مجدداً به کد مبنا ارجاع خواهد داد، که آن نتایج هم از همین مرحله‌ی اول برایش فراهم می‌گردد
  - ✓ با توجه به بیان پیاده‌سازی هر مرحله به‌صورت شبه‌کد امکان تغییر آن به هر زبان یا بستر دیگر نیز فراهم است
- **خودکارسازی:** تنها مرحله‌ای که غیر خودکار است و نیاز به دخالت صریح کاربر دارد، مرحله‌ی چهارم است، که امکان خودکارتر کردن آن نیز وجود دارد و غیر از آن تنها تعیین پارامترهای خوشه‌بندی است که نیاز به دخالت عامل انسانی دارد، که این امر نیز غیر قابل اجتناب است
- **قابلیت توسعه:** این قابلیت را بدون پیاده‌سازی نیز در ساختار ارائه شده می‌توان مشاهده کرد، علت آن است که
  - ✓ ساختار ارائه شده در واقع به شکل یک چهارچوب است و جداسازی منطقی مراحل آن، سبب می‌شود هر مدل نمایشی از ساختار یک برنامه‌ی کاربردی مبتنی بر وب که به صورت گرافی باشد را بتوان به عنوان ورودی به سیستم عرضه کرد.
  - ✓ همچنین نحوه‌ی انتخاب رابطه‌ی بین صفحات سبب می‌شود که این رویکرد با تغییرات جزیی در مرحله‌ی اول، به تمام برنامه‌های کاربردی تحت وب پیاده‌سازی شده با هر زبان قابل اعمال باشد و این مزیت‌های رویکرد کنونی نسبت به رویکردهای مشابه مانند مرجع [۳] است.

۵-۸- نتیجه‌گیری

در این فصل راه‌کارهای ارائه‌ی ابزاری که یک فرآیند چهار مرحله‌ای را حمایت کند ولی در عین حال عمومیت، خودکار بودن و قابلیت توسعه را نیز همراه داشته باشد، مطرح شد. سعی بر آن بود که با تشریح دقیق فرآیند پیاده‌سازی امکان استفاده‌ی مجدد و انتقال بستر برای چنین ابزاری فراهم گردد.

در راه پیاده‌سازی مرحله‌ی اول فرآیند که بسیار با زبانی که برنامه‌ی کاربردی مبتنی بر وب با آن پیاده‌سازی شده است در ارتباط است، ابتدا مقدمه‌ای بر JSP، به عنوان زبان مورد بررسی ما، ارائه گردید و سپس نحوه‌ی استخراج قطعات و روابط برنامه‌های کاربردی مبتنی بر وب پیاده‌سازی شده به این زبان بررسی شد و در نهایت با استفاده از یکی از روش‌های ساخت تحلیل‌گر نحوی در کامپایلر، شبه‌کد فرایند استخراج روابط عرضه گردید. برای تسهیل پیاده‌سازی مراحل دوم و سوم نیز، سعی شد قالب میانی ساده‌ای برای خروجی و ورودی هر مرحله تهیه شود، که در عین سادگی، کارایی لازم را نیز داشته باشد. الگوریتم خوشه‌بندی مرحله‌ی سوم نیز مرحله به مرحله تشریح گردید.

در نهایت، به پیاده‌سازی مرحله‌ی چهارم، یعنی مرحله‌ی توزیع بار کاری بین کارگزارها، براساس خروجی مرحله‌ی سوم، یعنی خوشه‌های تعیین شده، پرداخته شد. در این گذار بررسی مسأله‌ی متغییرهای مشترک سراسری قبلاً انجام پذیرفته بود، ولی برای نحوه‌ی توزیع بار با استفاده از امکانات زبان JSP الگوریتم مشخص همراه با شبه‌کد عرضه گردید، که از امکانی به اسم صافی برای پالایش و تغییر درخواست‌های HTTP در زبان JSP استفاده می‌کرد.

فصل هفتم

نتیجه‌گیری و کارهای آتی

۷-۱- نتیجه‌گیری

در این پایان‌نامه رویکردی چندمرحله‌ای برای مهندسی معکوس برنامه‌های کاربردی مبتنی بر وب عرضه گردید. نتیجه‌ی این رویکرد بازیابی معماری برنامه‌کاربردی و توزیع آن بر چند کارگزار وب برای کاهش ترافیک برنامه‌ی کاربردی مبتنی بر وب بود. با استفاده از این روش، رویت معماری برنامه‌ی کاربردی مبتنی بر وب در سطوح مختلف امکان‌پذیر گردید. طراحی فرایند به صورت چند مرحله‌ای هم عمومیت آن در قالب یک چهارچوب را سبب شده است و هم پیاده‌سازی آن به صورت یک سیستم لایه‌ای را.

از مزایای عمده‌ی رویکرد پیاده‌سازی شده، نسبت به روش‌های مشابه، می‌توان به عمومیت، نسبتاً خودکار بودن و قابلیت توسعه‌اش اشاره نمود. زیرا تنها مرحله‌ای از فرایند که به زبان و قلمرو برنامه وابسته است، مرحله‌ی اول است و نیز رویکرد مورد نظر را می‌توان تقریباً بر تمام برنامه‌های کاربردی تحت وب اعمال نمود، زیرا به استخراج روابطی از آن‌ها می‌پردازد که در تمام زبان‌های پیاده‌سازی برنامه‌های کاربردی مبتنی بر وب مشترک است. همچنین حجم برنامه نیز هرگز مانعی برای استفاده از این رویکرد نیست و رویکرد قادر است بر برنامه‌ی کاربردی مبتنی بر وب با هر اندازه‌ای اعمال شود.

از دلایل خودکار بودن آن می‌توان نیاز بسیار کم به اطلاعات و مستندات حیطه‌ی نرم‌افزار اشاره کرد و نیز مراحلی که با ارتباط با عامل انسانی برای تسهیل رویکرد انجام می‌گیرد، معدود است. گرچه بدیهی است که نمی‌توان نیاز به عامل انسانی را به طور مطلق از مراحل بازیابی معماری یا طراحی نرم‌افزار حذف نمود[۴].

ابزاری هم با نام RELB (Reverse Engineering and Load Balancing) به زبان جاوا برای این رویکرد پیاده‌سازی شده است که با توجه به مطالب ذکر شده در جزییات پیاده‌سازی و رهنمودها و شبه‌کدهای مرتبط می‌توان این ابزار یا بخشی از آن را با صرف زمان کوتاهی به زبان‌های دیگر نیز پیاده‌سازی نمود. ساختار لایه‌ای ابزار و استقلال نسبی لایه‌های مختلف آن از هم انعطاف‌پذیری بالایی برای آن فراهم نموده است و امکان تغییرات در آن را تسهیل بخشیده است.

در خاتمه الگوریتمی برای توزیع درخواست‌های HTTP به کارگزاران توسط یک کارگزار توزیع‌کننده، ایجاد گردید که از نتیجه مرحله‌ی خوشه‌بندی برای ارسال درخواست‌ها به صورت هدایت‌شده استفاده می‌کند.

معماری و بستر پیشنهادی برای پیاده‌سازی این الگوریتم توزیع نیز عرضه و پیاده‌سازی شد و با توجه به قابلیت گزارش‌گیری از ارجاعات توزیع شده در این الگوریتم، می‌توان از بازخورد[1] آن برای بهبود عملکرد سیستم توزیع شده و افزایش یا کاهش تعداد کارگزارهای اختصاص داده شده به هر خوشه استفاده کرد.

نتایج تحقیقات جانبی در کنار این پایان‌نامه را نیز نباید دست کم گرفت، از آن جمله می‌توان به استخراج تفاوت‌های فنی و غیر فنی برنامه‌های کاربردی غیر مبتنی بر وب و مبتنی بر وب پرداخت که سبب خواهد شد، با بررسی رویکردهای مهندسی معکوس برنامه‌های کاربردی غیر مبتنی بر وب و دانستن تفاوت‌های آن‌ها با برنامه‌های مبتنی بر وب، بتوان از زمینه‌ی غنی مهندسی معکوس نرم‌افزارهای غیر مبتنی بر وب در زمینه‌ی وب نیز بهره برد.

۷-۲- کارهای آتی

برخی جهت‌های مهم‌تر آینده‌ی ادامه این پایان نامه و جهت‌های مرتبط تحقیق، به اختصار در زیر آمده‌اند:

[1] Feedback

- چهارچوب پیشنهادی برای مهندسی معکوس (تا مرحله‌ی ۳) را می‌توان بر هر مدل‌سازی از برنامه‌های کاربردی مبتنی بر وب که توسط یک گراف قابل نمایش باشد، اعمال کرد. پس یکی از کارهایی که در آینده در ادامه‌ی این مقاله می‌تواند مطرح باشد، آن است که بیابیم رویکردهایی را که برنامه‌های کاربردی مبتنی بروب را می‌توان توسط گراف‌ها مدل‌سازی نمود. به این ترتیب می‌توان به ازای هر روش مدل‌سازی برنامه‌های کاربردی مبتنی بر وب یک خروجی داشت و با اتکا به چنین چهارچوبی درپی مقایسه‌ی این خروجی‌ها برآمد و حتی از آن نتیجه گرفت که برای بازیابی معماری یک برنامه‌کاربردی تحت وب بهتر است که چه نوع مدلسازی مبتنی بر گرافی برای آن در نظر گرفت.
- به علت این‌که ارتباطات در بین برنامه‌های کاربردی مبتنی بر وب متنوع‌تر است از برنامه‌های کاربردی غیر مبتنی بروب، رویکرد فعلی برای برنامه‌های کاربردی مبتنی بروب بهتر کارخواهد کرد. اما همانگونه که در بخش ۲-۳-۴ ذکر گردید، با درک وجوه تمایز برنامه‌های کاربردی مبتنی بر وب و برنامه‌های کاربردی غیر مبتنی بر وب، از زمینه‌ی مبسوط مهندسی معکوس برنامه‌های کاربردی غیر مبتنی بر وب نیز می‌توان کمک گرفت و با نگاشت مدل‌های نمایش‌دهنده نرم‌افزار، مدل‌های جدید در عرصه‌ی برنامه‌های کاربردی تحت وب عرضه نمود که مبتنی بر نمایش گرافی باشند.
- چون رویکرد کنونی به گونه‌ای عمومی ارائه شده است که وابستگی به بستر یا زبان خاصی ندارد با استفاده از یک سری ابزارهای متداول می‌توان آن را برای هر بستر یا زبانی قابل استفاده کرد.
- از مزایای خوشه‌بندی سلسله‌مراتبی می‌توان به تنظیم سطح تجرید در آن اشاره نمود، که با توجه به این مورد و نیز با توجه به آن‌که خروجی محصولاتی که از این فرآیند ایجاد می‌شوند در قالب نمایش گراف است، می‌توان با سطوح تجرید مختلف سیستم را مورد بررسی قرار داد. اما رویکردهای انتخاب نقطه‌ی برش مناسب برای این سلسله‌مراتب هنوز مورد بررسی قرار نگرفته‌اند.
- در مرحله‌ی سوم از رویکرد و ابزار، تنها یک الگوریتم خوشه‌بندی سلسله‌مراتبی تجمعی مورد استفاده قرار گرفته است و قاعده‌ی به‌هنگام‌سازی تشابه نیز در آن، پیوند ساده انتخاب شده است، اما با توجه به تنوع در رویکردهای خوشه‌بندی و قواعد به‌هنگام‌سازی تشابه و با توجه به استقلال پیاده‌سازی این مرحله از مراحل پیشین و پسین، می‌توان اثر سایر الگوریتم‌های خوشه‌بندی را نیز بر آن سنجید و با مقایسه، رویکرد خوشه‌بندی مناسب را برای آن انتخاب نمود.

- بر این اساس که روند توزیع بارکاری بین کارگزارها بر اساس نتایج خوشه‌بندی صورت می‌گیرد و رویکرد توزیع کنونی، قابلیت آمارگیری درخواست‌ها را دارد، می‌توان روشی چندسطحی برای توزیع بار کاری بین کارگزاران وب ارائه داد، که در سطح اول از رویکرد کنونی استفاده می‌کند و با استفاده از نتایج آماری ارجاع به خوشه‌ها و یافتن خوشه‌های پر ارجاع و کم ارجاع و تغییر تعداد کارگزارهای مرتبط با هر خوشه، درسطح دیگری بین خوشه‌های مشابه، استفاده از الگوریتم‌های دیگر توزیع بررسی شود

مراجع

[۱] M. L. Nelson, "A Survey of Reverse Engineering and Program Comprehension", http://arxiv.org/ftp/cs/papers/0503/0503068.pdf, April 1996.

[۲] A. E. Hassan, "Architecture Recovery of Web Applications", MSc Thesis, Waterloo Univ. 2001.

[۳] G. A. D. Lucca, A. R. Fasolino, U. D. Carlini, F. Pace, and P. Tramontana. "Comprehending web applications by a clustering based approach", In Proc. of the 10th International Workshop on Program Comprehension (IWPC), pages 261– 270 Paris, France, June 2002.

[۴] P. Grubb, A. A. Takang, Software Maintenance Concepts and Practice, 2nd edition, World Scientific Publishing, Singapore, 2003.

[۵] A. E. Hassan and R. C. Holt. "Towards a better understanding of web applications", 3rd International Workshop on Web Site, Evolution (WSE 2001), pages 112–116, November 2001.

[۶] ADELE team, IMAG Institute – CNRS, "Reverse-Engineering and Configuration Management: Concepts and perspectives", Laboratoire LSR, University of Grenoble I, 1997.

[۷] Sven Ziemer, "An Architecture for Web Applications", DIF 8914 Distributed Information Systems, November 2002.

[۸] G. A. Di Lucca, M. Di Penta, G. Antoniol, G. Casazza, “An approach for reverse engineering of web-based applications”, Proc. of 8th Working Conference on Reverse Engineering, IEEE CS Press, Los Alamitos, CA, 2001, pp. 231 240.

[۹] T. Wiggerts, "Using clustering algorithms in legacy systems remodularization", In FourthWorking Conference on Reverse Engineering (WCRE ’97), 1997.

[۱۰] G. A. D. Lucca, A. R. Fasolino, U. D. Carlini, F. Pace, and P. Tramontana. "Comprehending web applications by a clustering based approach". In Proc. of the 10th International Workshop on Program Comprehension (IWPC), pages 261– 270 Paris, France, June 2002. IEEE Computer Society.

[۱۱] P. Tonella1, F. Ricca, E. Pianta, C. Girardi, G. Di Lucca, A. R. Fasolino, P. Tramontana, "Evaluation Methods for Web Application Clustering", 5th International Workshop on Web Site Evolution, pages 33 – 41 , 2004,

[۱۲] M. Lanza, "Object Oriented Reverse Engineering", Phd Thesis, Bern Univ. 2003.

[۱۳] J. Conallen, Building Web Applications with UML, 2nd Edition, Addison Wesley, 2002.

[۱۴] R. C. Holt. An Introduction to TA: the Tuple-Attribute Language, http://plg.uwaterloo.ca/~holt/papers/ta.html, Mar. 1997.

[١٥] R. C. Holt. Structural manipulations of software architecture using Tarski relational algebra. In Proceedings of WCRE'98, Oct. 1998.

[١٦] G. Di Lucca, A. Fasolino, P. Tramontana, and U. De Carlini, "Abstracting business level UML diagrams from web applications," in Proceedings of International Workshop on Web Site Evolution, (Amsterdam, The Netherlands), pp. 12–19, Oct 2003

[١٧] P. Tonella, F. Ricca, E. Pianta, C. Girardi, "Using keyword extraction for web site clustering", 5th International Workshop on Web Site Evolution, IEEE CS Press, Los Alamitos, CA, 2003, pp. 41-48

[١٨] R. S. Pressman. What a Tangled Web We Weave. IEEE Software, 17(1):18–21, Jan. 2000.

[١٩] http://plg.uwaterloo.ca/~aeehassa/home/pubs/msthesis.ppt

[٢٠] G. A. Di Lucca, A. R. Fasolino, P. Tramontana, U. D. Carlini, " Supporting Concept Assignment in the Comprehension of Web Applications", Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), IEEE CS Press, 2004.

[٢١] F. Ricca and P. Tonella. Using clustering to support the migration from static to dynamic web pages. In Proc. of the International Workshop on Program Comprehension (IWPC), pages 207–216, Portland, Oregon, USA, May 2003. IEEE Computer Society.

[٢٢] V. C. Garcia, D. L.edio, A. F. do Prado, " Towards an effective approach for Reverse Engineering", Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04), IEEE CS Press, 2004.

[٢٣] Mancoridis, S., Mitchell, B., Chen, Y. and Gansner, E. (1999). "Bunch: A Clustering Tool for the Recovery and Maintenance of Software System Organizations of Source Code", Proc. of Int'l Workshop on Program Comprehension.

[٢٤] Mancoridis, S., Mitchell, B. S., Rorres, C., Chen, Y. and Gansner, E. R. (1998). "Using Automatic Clustering to Produce High-Level System Organizations of Source Code", Proc. of the 6th Int'l Workshop on Program Comprehension, pp.45-52.

[٢٥] Shokoufandeh, A., Mancoridis, S., Maycock, M., "Applying Spectral Methods to Software Clustering," Proc. of the Ninth Working Conference on Reverse Engineering (WCRE.02).

[٢٦] A. Trifu, "Using Cluster Analysis in the Architecture Recovery of Object-Oriented Systems", Master's Thesis, Institut für Programmstrukturen und Datenorganisation, Universtät Karlsruhe, 2001.

[٢٧] Hans Bergsten, JavaServer Pages, 2nd Edition, O'Reilly Publishing, August 2002.

[۲۸] HTML 4.0 Specification, W3C Recommendation,
http://www.w3.org/TR/1998/REC-html40-19980424, revised on 24-Apr-1998.
[۲۹] B. W. Perry, Java Servlet & JSP Cookbook, O'Reilly Publishing, January 2004.
[۳۰] T. Chen, "Load Balancing Strategies on Cluster-based Web Servers",Project Report,
Department of Computing Science, National University of Singapore,
http://www.comp.nus.edu.sg/~teoym/ref.pdf , 2000.
[۳۱] L. Aversa, A. Bestavros, "Load Balancing a Cluster of Web Servers using
Distributed Packet Rewriting" Technical Report of Computer Science Department,
Boston University, 1999.
[۳۲] V. Viswanathan, "Load Balancing Web Applications",
http://www.onjava.com/pub/a/onjava/2001/09/26/load.html.
[۳۳] کریمی، سلمان. "طرح و پیاده‌سازی مکانیزمی برای توزیع بارکاری سرویس‌دهنده‌های وب بر مبنای تکنولوژی ASP"، پایان نامه‌ی کارشناسی ارشد مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، اسفند ۱۳۸۱.
[۳٤] Y. M. TEO, R. AYANI, "Comparison of Load Balancing Strategies on Cluster-based Web Servers", Transactions of the Society for Modeling and Simulation, 2001.
[۳٥] A. Aho, R. Sethi, J. D. Ullman, Compilers: Principals, Techniques, and Tools, Addison Wesley, 1986.
[۳٦] J. Conallen. Modeling web applications with uml. White paper, Conallen Inc., http://www.conallen.com/whitepapers/webapps/ModelingWebApplications.htm, March 1999.