This file has been cleaned of potential threats.

If you confirm that the file is coming from a trusted source, you can send the following SHA-256 hash value to your admin for the original file.

0621ccbf752980ce47bb3615200dc380610a1a2bd063abe12101b57e30584ad3

To view the reconstructed contents, please SCROLL DOWN to next page.

رساله دکتری

گروه مهندسی کامپیوتر

روشی مبتنی بر داده‌های پیوندی برای مهندسی وب با رویکرد قابلیت استفاده مجدد

نگارنده:

صمد پایدار

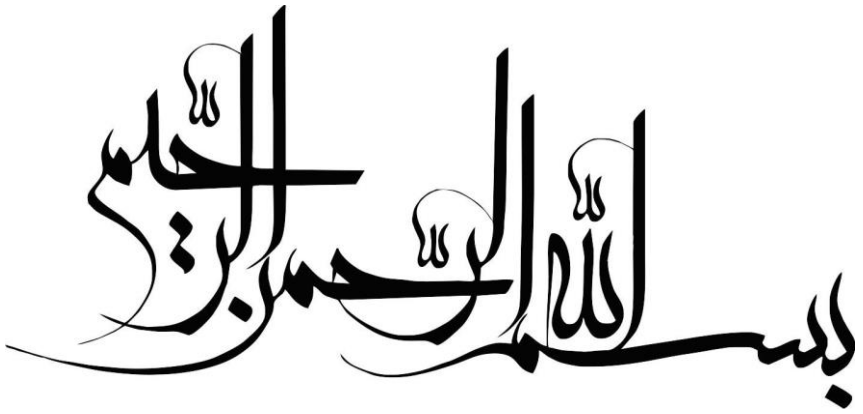
استاد راهنما:

دکتر محسن کاهانی

استاد مشاور:

دکتر سعید عربان

اسفند ۱۳۹۲



تقدیر و تشکر

بدینوسیله بر خود لازم می‌دانم که از زحمات بی‌دریغ استاد گرامی، جناب آقای دکتر محسن کاهانی که راهنمایی‌های ارزشمند ایشان در تمام مراحل انجام این پایان‌نامه راه‌گشای من بوده است تشکر نمایم.

همچنین از تلاش‌های بی‌وقفه و پشتیبانی‌های پدر و مادر عزیزم و همسر مهربانم کمال سپاس‌گزاری را دارم.

چکیده فارسی

مهندسی وب بعنوان شاخه‌ای از مهندسی نرم‌افزار بر ارائه روش‌های سامانمند برای توسعه برنامه-های کاربردی تحت وب تمرکز دارد. بدین منظور اغلب روشگان‌های مهندسی وب از رویکرد توسعه مبتنی بر مدل استفاده می‌کنند. با توجه به نقش اساسی مدل‌ها در این رویکرد، چالشی که بوجود می‌آید آن است که توسعه هر برنامه کاربردی جدید نیازمند تولید تعداد زیادی مدل می‌باشد. ارائه یک راهکار استفاده مجدد که برای مدل‌سازی یک برنامه کاربردی از مدل‌های برنامه‌های کاربردی مشابه استفاده کند می‌تواند در پاسخ به این چالش مؤثر باشد. با توجه به اهمیت مدل نیازمندی‌های عملیاتی، در این رساله، یک رویکرد جدید برای استفاده مجدد در مدل‌سازی نیازمندی‌های عملیاتی ارائه شده است که امکان استفاده مجدد را در مراحل اولیه توسعه فراهم می‌کند. در این رویکرد با دریافت توصیف کلی نیازمندی-های عملیاتی در قالب نمودار مورد کاربرد *UML*، نسخه اولیه توصیف جزئی نیازمندی‌های عملیاتی، بطور نیمه‌خودکار و در قالب نمودارهای فعالیت *UML* ایجاد می‌شود. روش پیشنهادی شامل دو مرحله اصلی است که مرحله اول به آماده‌سازی مخزن معنایی مدل‌ها و مرحله دوم به استفاده مجدد از این مخزن اختصاص دارد. بدین منظور، در مرحله اول، از الگوریتم‌های جدیدی برای حاشیه‌نویسی نمودارهای فعالیت و تشخیص مفاهیم و رفتار مورد کاربردها و همچنین از فناوری‌های وب معنایی استفاده شده است. مرحله دوم نیز از معیار جدیدی برای تشخیص شباهت دو مورد کاربرد و از الگوریتم جدیدی برای تطبیق نمودارهای فعالیت و همچنین از منابع وب معنایی و داده‌های پیوندی برای تأمین نیازهای اطلاعاتی استفاده می‌کند. ارزیابی‌های انجام شده نشان می‌دهد روش پیشنهادی از دقت و کارایی مناسبی برخوردار است و استفاده از وب معنایی نقش مؤثری در بهبود نتایج آن دارد. بدین ترتیب درستی ایده اصلی روش پیشنهادی، که استفاده از نمودار مورد کاربرد بعنوان نقطه شروع فرآیند استفاده مجدد می-باشد، مورد تأیید قرار می‌گیرد. با این حال، الگوریتم‌های ارائه شده دارای کاستی‌هایی هستند که رفع آن-ها نیازمند تحقیق و نوآوری بیشتری می‌باشد.

فهرست مطالب

۱- مقدمه ۱۱

۱-۱ تعریف مسأله ۱۱

۱-۲ راه حل پیشنهادی ۱۵

۱-۳ نوآوری‌های راه حل پیشنهادی ۱۷

۱-۴ ساختار رساله ۱۸

۲- مرور کارهای گذشته ۱۹

۲-۱ مهندسی وب ۱۹

۲-۱-۱ روشگان UWE ۲۴

۲-۱-۲ نتیجه‌گیری ۲۸

۲-۲ استفاده مجدد از نرم‌افزار ۲۹

۲-۲-۱ نتیجه‌گیری ۴۰

۲-۳ معیارهای شباهت نرم‌افزار ۴۲

۲-۳-۱ نتیجه‌گیری ۴۴

۲-۴ تولید خودکار مدل ۴۵

۲-۴-۱ نتیجه‌گیری ۴۸

۲-۵ مهندسی نرم‌افزار مبتنی بر وب معنایی ۵۱

۲-۵-۱ رویکرد اول: استفاده از فناوری‌های وب معنایی ۵۴

۲-۵-۱-۱ داده‌های پیوندی ۵۶

۲-۵-۱-۲ پروژه LD2SD ۵۹

۲-۵-۲ رویکرد دوم: استفاده از وب معنایی ۶۰

۲-۵-۳ نتیجه‌گیری ۶۳

۳- روش پیشنهادی ۶۶

۳-۱ مقدمه ۶۸

۳-۱-۱ الگوریتم تشخیص مفاهیم و رفتار مورد کاربرد ۶۹

۳-۲ فرآیند آماده‌سازی مخزن معنایی مدل‌ها ۷۰

۳-۲-۱ اعتبارسنجی مدل‌ها ۷۱

۳-۲-۲ ایجاد بازنمایی معنایی مدل‌ها ۷۳

۳-۲-۲-۱ هستان‌شناسی UML ۷۴

۳-۲-۲-۲ مترجم ۷۵

۳-۲-۲-۳ مخزن معنایی مدل‌ها ۷۹

۳-۲-۳ حاشیه‌نویسی مدل‌ها ۷۹
۳-۲-۳-۱ الگوریتم حاشیه‌نویسی مدل‌ها ۸۰
۳-۲-۳-۲ انواع حاشیه‌نویسی‌ها ۸۲
۳-۲-۳-۳ رویه حل تعارض ۸۶
۳-۳ فرآیند استفاده مجدد ۸۸
۳-۳-۱ پیشنهاد مورد کاربرد ۸۹
۳-۳-۱-۱ معیار شباهت مورد کاربرد ۹۰
۳-۳-۲ تطبیق نمودار فعالیت ۹۴
۳-۳-۲-۱ الگوریتم تطبیق نمودار فعالیت ۹۶
۳-۴ خلاصه ۱۰۳
۴- ارزیابی ۱۰۵
۴-۱ سوال‌های تحقیق ۱۰۷
۴-۲ آماده‌سازی مخزن معنایی مدل‌ها ۱۰۷
۴-۲-۱ مجموعه داده ۱۰۸
۴-۳ آمادگی وب معنایی ۱۰۹
۴-۴ حاشیه‌نویسی مدل‌ها ۱۱۳
۴-۵ الگوریتم حل تعارض حاشیه‌نویسی ۱۲۰
۴-۶ الگوریتم تشخیص مفاهیم و رفتار مورد کاربرد ۱۲۶
۴-۶-۱ معیار شباهت مورد کاربرد ۱۳۰
۴-۷ الگوریتم تطبیق ۱۳۷
۴-۸ نقش وب معنایی ۱۴۳
۴-۹ خلاصه ۱۴۵
۵- نتیجه‌گیری و کارهای آینده ۱۴۸
۶- مراجع ۱۵۲

فهرست علائم و اختصارات

11PIAP: 11-point Interpolated Average Precision

AD: Activity Diagram

CBR: Case Based Reasoning

DSL: Domain Specific Language

LOD: Linking Open Data

MDD: Model Driven Development

MDA: Model Driven Architecture

POS Tagger: Part-Of-Speech Tagger

SLOC: Source Line of Code

UC: Use Case

UCD: Use Case Diagram

UWE: UML-based Web Engineering

XMI: XML Metadata Interchange

فهرست جداول

جدول ۴-۱ آماری درباره فرآیند بازنمایی معنایی مدل‌ها ۱۰۸
جدول ۴-۲ نتایج آزمایش آمادگی وب معنایی ۱۱۱
جدول ۴-۳ برخی از نتایج جستجو بر روی موتور جستجوی SWOOGLE ۱۱۱
جدول ۴-۴ زمان اجرای دو رویه جستجو بر روی وب معنایی ۱۱۳
جدول ۴-۵ اطلاعاتی درباره بخشی از استاندارد طلایی برای ارزیابی الگوریتم حاشیه‌نویسی ۱۱۵
جدول ۴-۶ اطلاعاتی درباره استاندارد طلایی برای ارزیابی الگوریتم حاشیه‌نویسی ۱۱۵
جدول ۴-۷ جزئیات حاشیه‌نویسیهای نمودار فعالیت ‘CreateContact’ ۱۲۰
جدول ۴-۸ ارزیابی نقش الگوریتم حل تعارض در الگوریتم حاشیه‌نویسی ۱۲۶
جدول ۴-۹ نتایج اجرای الگوریتم تشخیص مفاهیم و رفتار مورد کاربرد ۱۳۰
جدول ۴-۱۰ مقدار پارامترهای بدست آمده از الگوریتم ژنتیک ۱۳۲
جدول ۴-۱۱ نخستین جواب بدست آمده برای هر مورد آزمون با استفاده از روش اول ۱۳۵
جدول ۴-۱۲ نتایج ارزیابی الگوریتم تطبیق ۱۳۹
جدول ۴-۱۳ نتایج ارزیابی الگوریتم تطبیق به تفکیک نوع حاشیه‌نویسی ۱۴۰
جدول ۴-۱۴ نتایج الگوریتم تطبیق برای مورد آزمون اول ۱۴۱
جدول ۴-۱۵ برخی از نتایج جستجو بر روی وب معنایی برای مفهوم ‘Patient’ ۱۴۳
جدول ۴-۱۶ نتایج ارزیابی الگوریتم تطبیق بدون استفاده از قابلیت جستجو در وب معنایی ۱۴۵

فهرست شکل‌ها

شکل ۲-۱ تاریخچه روش‌های مختلف مدل‌سازی در مهندسی وب [47] ۲۳

شکل ۲-۲ روش‌های مختلف مدل‌سازی در مهندسی وب [47] ۲۴

شکل ۲-۳ تبدیل مدل‌ها در فرآیند توسعه UWE [34] ۲۷

شکل ۲-۴ رویکرد رایج در روش‌های تولید خودکار مدل (بالا)، رویکرد پیشنهادی این رساله (پایین) ۵۱

شکل ۲-۵. وضعیت ابر LOD در مارچ ۲۰۰۹ [205] ۵۸

شکل ۲-۶. وضعیت ابر LOD در سپتامبر ۲۰۱۱ [205] ۵۸

شکل ۲-۷. لایه‌های تعریف شده در روشگان LD2SD [187] ۶۰

شکل ۳-۱ نمای کلی روش پیشنهادی ۶۷

شکل ۳-۲ نمای کلی فرآیند آماده‌سازی مخزن معنایی مدل‌ها ۷۱

شکل ۳-۳. قسمتی از هستان‌شناسی UML ۷۵

شکل ۳-۴. برخی از قوانین مورد استفاده مترجم ۷۷

شکل ۳-۵. مثالی از نمودار مورد کاربرد UML ۷۸

شکل ۳-۶. قسمت کوچکی از بازنمایی معنایی مربوط شکل ۳-۵ ..... ۷۹

شکل ۳-۷. شبه کد الگوریتم حاشیه‌نویسی نمودار فعالیت ۸۱

شکل ۳-۸ نمای کلی فرآیند استفاده مجدد ۸۹

شکل ۳-۹. شبه کد الگوریتم تطبیق نمودار فعالیت ۹۷

شکل ۳-۱۰. دو نمونه پرس و جوی SPARQL برای بازیابی زیرکلاس‌های کلاس Student ۱۰۲

شکل ۳-۱۱. یک نمونه پرس و جوی SPARQL برای بازیابی کلاس‌های مرتبط با کلاس Student ۱۰۲

شکل ۴-۱. نمایی از واسط گرافیکی نمونه اولیه ۱۰۵

شکل ۴-۲. نمایی از واسط گرافیکی نمونه اولیه ۱۰۶

شکل ۴-۳. یک پرس و جوی SPARQL برای بازیابی کلاس ’Student‘ بهمراه خصیصه‌های آن ۱۱۰

شکل ۴-۴. نتایج ارزیابی الگوریتم حاشیه‌نویسی نمودار فعالیت ۱۱۶
شکل ۴-۵. نمودار فعالیت 'CreateContact' به همراه حاشیه‌نویسی‌های آن..... ۱۱۹
شکل ۴-۶. نمودار فعالیت 'Add Image' ۱۲۲
شکل ۴-۷. نمودار کلاس مربوط به سیستم 'Philoponella' ۱۲۲
شکل ۴-۸ نمودار فعالیت 'Browse LinkInfos' ۱۲۴
شکل ۴-۹. نتایج ارزیابی الگوریتم حاشیه‌نویسی بدون استفاده از الگوریتم حل تعارض حاشیه‌نویسی ۱۲۵
شکل ۴-۱۰ . نتایج الگوریتم ژنتیک برای تعیین مقدار پارامترها..... ۱۳۲
شکل ۴-۱۱. ارزیابی معیار ارائه شده از نظر P@k ۱۳۴
شکل ۴-۱۲ ارزیابی معیار ارائه شده از نظر 11pIAP ۱۳۷
شکل ۴-۱۳. نتایج ارزیابی الگوریتم تطبیق ۱۴۰
شکل ۴-۱۴. نمودار فعالیت 'createPatient' (تطبیق یافته از روی نمودار فعالیت 'createContact') ۱۴۲
شکل ۴-۱۵. نتایج ارزیابی الگوریتم تطبیق بدون استفاده از قابلیت جستجو در وب معنایی..... ۱۴۴

۱- مقدمه

در این فصل، ابتدا مسأله‌ای که در تحقیق جاری مورد توجه قرار گرفته است معرفی شده و سپس، در بخش ۱-۲، روش پیشنهادی این رساله برای حل مسأله مورد نظر به اختصار شرح داده می‌شود. در ادامه نیز نوآوری‌های روش پیشنهادی و ساختار بقیه مطالب این رساله ذکر می‌شود.

۱-۱ تعریف مسأله

طی دو دهه اخیر، وب به عنوان یک سکوی نرم‌افزاری که امکان دسترسی توزیع شده به برنامه‌های کاربردی و فرایندهای کسب و کار را فراهم می‌کند، مورد توجه مهندسان نرم‌افزار قرار داشته و برنامه‌های کاربردی تحت وب به عنوان دسته مهمی از نرم‌افزارها مطرح بوده‌اند. علیرغم شباهت‌های زیاد، توسعه برنامه‌های کاربردی تحت وب با توسعه نرم‌افزارهای معمولی از جنبه‌های متعددی متفاوت است و ویژگی‌های خاص برنامه‌های کاربردی تحت وب، چالش‌های جدیدی را مطرح کرده است [1~~1~~]. مهندسی وب به عنوان شاخه‌ای از مهندسی نرم‌افزار در پاسخ به این چالش‌ها شکل گرفته و به دنبال روش‌های سامانمند[۱] مقرون به صرفه و قابل کمی‌سازی[۲] برای تحلیل نیازمندی‌ها، طراحی، پیاده‌سازی، تست، عملیاتی کردن و نگهداری برنامه‌های کاربردی تحت وب با کیفیت بالا می‌باشد [2~~2~~].

از اوایل دهه ۹۰ میلادی تا کنون چند روشگان[۳] مهندسی وب توسط محققان معرفی شده است که عموما از رویکرد توسعه مبتنی بر مدل[۴] استفاده می‌کنند [3~~3~~]. در این رویکرد، تمرکز بر ایجاد مدل‌های با کیفیت و دقیقی است که برنامه کاربردی[۵] مورد نظر را از جنبه‌های مختلف توصیف می‌کنند. این مدل‌ها به عنوان پیش‌ران[۶] فرآیند توسعه عمل می‌کنند و با بکارگیری تکنیک‌های تبدیل مدل[۷] انواع جدیدی از مدل‌ها از روی مدل‌های اولیه ایجاد می‌شود. این رویه که بطور خودکار یا نیمه‌خودکار انجام

---

۱ Systematic
۲ Quantifiable
۳ Methodology
۴ Model-Driven Development (MDD)
۵ بمنظور پرهیز از اطاله کلام، در ادامه، عبارت‌های "برنامه کاربردی" و "برنامه کاربردی تحت وب" معادل یکدیگر در نظر گرفته می‌شوند.
۶ Driver
۷ Model Transformation

می‌شود، ممکن است چند مرحله ادامه پیدا کرده و تا ایجاد بخش‌هایی از نسخه اجرایی برنامه کاربردی نیز پیش رود [4, 5].

علیرغم اینکه روشگان‌های مهندسی وب موجب بهبود فرآیند توسعه می‌شوند، چالشی که در استفاده از آن‌ها وجود دارد آن است که با توجه به اهمیت اساسی مدل‌ها در مهندسی وب، برای توسعه هر برنامه کاربردی جدید تحت وب، باید تعداد زیادی مدل ایجاد شود که هر یک بخشی از برنامه کاربردی را از دیدگاه خاصی توصیف می‌کنند. ایجاد این مدل‌ها، بخصوص برای یک برنامه کاربردی بزرگ کاری زمان‌بر و پیچیده است. با توجه به اینکه برنامه‌های کاربردی مختلف علیرغم نقاط تمایزشان دارای ویژگی‌ها، نیازمندی‌ها و قابلیت‌های مشابهی هستند طبیعی است که مدل‌های مختلف این برنامه‌های کاربردی نیز دارای شباهت‌هایی باشند. بنابراین انتظار می‌رود فرصت و پتانسیلی برای بهره‌گیری از مدل-های برنامه‌های کاربردی موجود، در ایجاد مدل‌های یک برنامه کاربردی جدید وجود داشته باشد. تحقق چنین امری به معنای استفاده مجدد از مدل‌های موجود می‌باشد که می‌تواند هزینه و پیچیدگی فرآیند مدل‌سازی یک برنامه کاربردی جدید را کاهش دهد.

استفاده مجدد، یکی از مهم‌ترین و قدیمی‌ترین رویکردها در حوزه مهندسی نرم‌افزار است که از دهه ۶۰ میلادی بطور جدی مورد توجه محققان قرار گرفته است [6]. ضرورت استفاده مجدد از مدل[۱] نیز بطور خاص از دهه ۹۰ بیشتر مورد توجه قرار گرفته است [7].

در حالت کلی، فرآیند استفاده مجدد شامل چهار مرحله است [8]:

۱. بازنمایی[۲] مصنوعات نرم‌افزاری که پتانسیل کافی برای استفاده مجدد را دارند به شکل مناسبی بازنمایی و این بازنمایی در یک مخزن ذخیره می‌شود.
۲. بازیابی[۱] مصنوعات نرم‌افزاری موجود در مخزن، با مصنوع ورودی که در حکم پرس و جو است، مقایسه شده و بر اساس یک معیار شباهت یا برازندگی، مناسب‌ترین مصنوعات از مخزن بازیابی می‌شود

[۱] Model Reuse
[۲] Representation

۳. تطبیق[۲] مصنوعات نرم‌افزاری بازیابی شده، بمنظور بکارگیری در فعالیت پیش رو، مورد تطبیق و ویرایش قرار گرفته و با استفاده از آن‌ها، مصنوعات جدید مورد نظر ایجاد می‌شود.

۴. الحاق[۳] در این مرحله، مصنوعات جدید حاصل از فرآیند استفاده مجدد، به مخزن اولیه افزوده و برای استفاده مجدد در آینده آماده می‌شوند

مطالعه کارهای موجود نشان می‌دهد عموم کارهایی که در زمینه استفاده مجدد انجام شده مربوط به دو مرحله اول می‌باشد. به بیان دیگر، بیشتر کارهای انجام شده، نظیر [9~~9~~, 10~~10~~, 11~~11~~, 12~~12~~]، با معرفی یک روش یا ابزار جدید برای ذخیره و بازیابی مصنوعات امیدبخش، سعی کرده‌اند فرآیند استفاده مجدد را محقق کنند. از آنجا که این جستجو و بازیابی به معیاری برای مقایسه مصنوعات موجود نیاز دارد، در برخی از کارها نیز معیار جدیدی برای محاسبه شباهت مصنوعات نرم‌افزاری ارائه شده است [12~~12~~, 13~~13~~]. همچنین، کارهای انجام شده، تحت عناوین مختلفی معرفی شده‌اند که می‌تواند متأثر از نوع مصنوع نرم‌افزاری مورد نظر، تکنیک بکاررفته یا دیدگاه محققان مربوطه باشد، مثلا جستجوی کد منبع[۴][14~~14~~]، پیشنهاد اجزای نرم‌افزاری[۵][15~~15~~]، انتخاب اجزای نرم‌افزاری[۶][16~~16~~]، موتور جستجوی کد منبع[۷][17~~17~~, 18~~18~~]، موتور جستجوی مدل[۸][13~~13~~] و بازیابی کد منبع[۹][19~~19~~].

در رابطه با مرحله سوم، یعنی تطبیق، متأسفانه کارهای بسیار کمتری انجام شده است. در حالی که دیدگاه رساله جاری آن است که استفاده مجدد، عملا در مرحله تطبیق است که تحقق می‌یابد. مطالعه کارهای انجام شده نشان می‌دهد که محققان، عموما هدف خود را به مرحله بازیابی محدود کرده و موضوع تطبیق را به عهده خود کاربر گذاشته‌اند. شاید یک دلیل این امر را بتوان در این نکته دانست که فعالیت انجام شده در مرحله بازیابی از جنس جستجو است و با توجه به قابلیت کامپیوترها در بحث

---

۱ Retrieval
۲ Adaptation
۳ Incorporation
۴ Source Code Search
۵ Component Recommendation
۶ Component Selection
۷ Source Code Search Engine
۸ Model Search Engine
۹ Source Code Retrieval

جستجو بهتر است این فعالیت به آن‌ها سپرده شود. اما فعالیت مرحله دوم یک فعالیت مولد[۱]است و نیازمند دانش قابل توجهی (دانش پس‌زمینه[۲] دانش دامنه[۳]و دانش فرآیند[۴] است که معمولا انتظار نمی‌رود کامپیوترها تا حد کافی از آن برخوردار باشند.

با توجه به بحث فوق و با توجه به اهمیت استفاده مجدد از مدل‌ها در مهندسی وب، هدف تحقیق جاری ارائه روشی برای استفاده مجدد از مدل در مهندسی وب می‌باشد که همه مراحل فوق و بخصوص موضوع تطبیق را پوشش دهد. بطور خاص، از آنجا که مدل نیازمندی‌های عملیاتی[۵]از مهمترین انواع مدل‌ها هستند، تحقیق جاری بر روی این گونه از مدل‌ها تمرکز دارد. همچنین راه حل پیشنهادی این رساله بمنظور فراهم کردن دانش مورد نیاز برای استفاده مجدد، استفاده از منابع وب معنایی می‌باشد.

یکی از روش‌های رایج در توصیف مدل نیازمندی‌های عملیاتی، مدل‌سازی مورد کاربرد[۶]می‌باشد که در آن، ابتدا توصیفی سطح بالا از نیازمندی‌ها در قالب نمودار مورد کاربرد *UML*[۷] ایجاد می‌شود و سپس هر یک از موارد کاربرد مربوطه، با استفاده از تکنیک دیگری مانند نمودار فعالیت *UML*[۸] بطور جزئی‌تر مورد توصیف قرار می‌گیرد. از آنجا که مدل نیازمندی‌های عملیاتی معمولا در نخستین مراحل توسعه ایجاد می‌شود و امکان استخراج برخی مدل‌های دیگر از روی مدل نیازمندی‌های عملیاتی وجود دارد، چنانچه بتوان قابلیت استفاده مجدد را در ایجاد این مدل‌ها فراهم کرد، می‌توان از مزایای آن در نخستین مراحل توسعه برخوردار شد.

با توجه به مباحث فوق، پرسش‌های اصلی در این پژوهش عبارتند از:

- آیا می‌توان روشی خودکار یا نیمه‌خودکار ارائه داد که توصیف مختصر نیازمندی‌های عملیاتی یک برنامه کاربردی تحت وب را در قالب نمودار مورد کاربرد *UML* دریافت کرده و نسخه اولیه توصیف جزئی نیازمندی‌های مربوطه را در قالب تعدادی نمودار فعالیت *UML* ایجاد کند؟

---

۱ Generative
۲ Background Knowledge
۳ Domain Knowledge
۴ Process Knowledge
۵ Functional Requirements
۶ Use Case Modeling
۷ Use Case Diagram
۸ Activity Diagram

- آیا می‌توان معیاری ارائه کرد تا بتوان برنامه‌های کاربردی تحت وب که از نظر مدل نیازمندی‌های عملیاتی شبیه برنامه کاربردی جدید هستند را مشخص نمود تا بتوان از مدل‌های آن برنامه‌ها برای ایجاد مدل‌های برنامه کاربردی جدید استفاده کرد؟
- آیا استفاده از منابع وب معنایی و بخصوص داده‌های پیوندی می‌تواند راهکار مناسبی بمنظور فراهم کردن دانش مورد نیاز برای الگوریتم‌های استفاده شده در فرآیند استفاده مجدد باشد؟
- آیا استفاده از فناوری‌های وب معنایی می‌تواند در بهبود فرآیند استفاده مجدد ارائه شده مؤثر باشد؟

۱-۲ راه حل پیشنهادی

روش پیشنهادی این رساله شامل دو مرحله اصلی است که در مجموع، امکان استفاده مجدد در تولید مدل نیازمندی‌های عملیاتی برنامه‌های کاربردی تحت وب را فراهم می‌کند. مرحله نخست به آماده-سازی یک مخزن معنایی از مدل‌های برنامه‌های کاربردی موجود اختصاص دارد. مخزن معنایی، مؤلفه‌ای نرم‌افزاری است که قابلیت ذخیره بازنمایی معنایی اطلاعات مدل‌ها بر اساس مدل داده وب معنایی را داشته و امکان جستجو و بازیابی این اطلاعات با استفاده از پرس‌وجوهای به زبان *SPARQL* را فراهم می-کند. دلیل استفاده از بازنمایی معنایی برای ذخیره اطلاعات مدل‌ها آن است که بطور تجربی نشان داده شده که ایجاد یک لایه معنایی بر روی مصنوعات نرم‌افزاری، قابلیت جستجو، بازیابی و دسترسی به اطلاعات آن‌ها را بهبود بخشیده و پیاده‌سازی تحلیل‌های مختلف بر روی این اطلاعات را ساده‌تر می‌کند [20~~20~~].

مرحله نخست روش پیشنهادی، علاوه بر ایجاد و ذخیره بازنمایی معنایی از مدل‌های موجود، شامل حاشیه‌نویسی[۱] این مدل‌ها نیز می‌باشد. جزئیات مربوط به عمل حاشیه‌نویسی در بخش ۳-۲-۳ توضیح داده خواهد شد. در اینجا تنها به ذکر این نکته بسنده می‌شود که در روش پیشنهادی، حاشیه-

[۱] Annotation

نویسی‌ها عنصری اساسی هستند و هر چقدر حاشیه‌نویسی‌های بیشتر و دقیق‌تری برای یک مدل ایجاد شود، پتانسیل بیشتری برای استفاده مجدد از آن مدل وجود خواهد داشت.

پس از آنکه مخزن معنایی مدل‌ها ایجاد و عمل حاشیه‌نویسی انجام شد، مرحله دوم روش پیشنهادی قابل اجرا می‌باشد که طی آن، کاربر با دادن ورودی مناسب می‌تواند به استفاده مجدد از مدل-های موجود بپردازد. ورودی این مرحله، نمودار مورد کاربرد مربوط به برنامه کاربردی جدیدی که قرار است توسعه داده شود. این نمودار مورد کاربرد در حکم توصیف کلی نیازمندی‌های عملیاتی برنامه کاربردی مورد نظر می‌باشد.

با دریافت ورودی، روش پیشنهادی ابتدا به جستجو در مخزن معنایی مدل‌ها پرداخته و برای هر مورد کاربرد از نمودار ورودی، شبیه‌ترین مورد کاربردها را در مخزن پیدا کرده و لیست مرتب موارد کاربرد بازیابی شده را به کاربر ارائه می‌کند. کاربر پس از انجام بررسی کوتاهی، مناسب‌ترین مورد را از بین موارد کاربرد پیشنهادی انتخاب می‌کند. پس از آن، نمودار فعالیتی که مورد کاربرد انتخابی کاربر را توصیف می‌کند از مخزن بازیابی شده و به عنوان یک الگو برای ایجاد نمودار فعالیتی که مورد کاربرد ورودی را توصیف کند استفاده می‌شود. در این رساله، برای اشاره به این عمل، از عنوان تطبیق استفاده می‌شود، یعنی نمودار فعالیت مورد نیاز از طریق تطبیق یک نمودار فعالیت موجود ایجاد می‌شود.

محاسبه شباهت معنایی مورد کاربرد ورودی با موارد کاربرد موجود، بر اساس معیار جدیدی که در این تحقیق ارائه شده است انجام می‌شود. در طراحی این معیار، جنبه‌های مختلف دو مورد کاربرد در نظر گرفته شده که در بخش ۱-۱-۳-۳ بطور کامل توضیح داده شده است.

با توجه به اینکه ورودی‌ای که از کاربر گرفته می‌شود صرفا یک نمودار مورد کاربرد است و میزان اطلاعاتی که بطور صریح در آن بیان شده اندک است، برای خودکارسازی تطبیق نمودار فعالیت، نیاز به اطلاعات بیشتری می‌باشد. بدین منظور، روش پیشنهادی از منابع وب معنایی و داده‌های پیوندی برای تأمین نیازهای اطلاعاتی روش ارائه شده استفاده می‌کند.

۳-۱ نوآوری‌های راه حل پیشنهادی

کارهای اصلی انجام شده در این رساله را می‌توان در موارد زیر خلاصه کرد:

- ارائه راهکاری برای ایجاد لایه معنایی بر روی مدل‌های *UML*، که شامل موارد زیر می‌باشد:
  - یک هستان‌شناسی[۱] برای توصیف رسمی مدل‌های *UML*
  - یک مترجم برای پردازش مدل‌های *UML* و انتشار اطلاعات آن‌ها در قالب معنایی
  - مخزنی برای ذخیره بازنمایی معنایی مدل‌ها با قابلیت اجرای پرس و جو به زبان *SPARQL*
- ارائه الگوریتمی برای حاشیه‌نویسی نمودارهای فعالیت *UML*
- ارائه معیاری برای محاسبه شباهت معنایی دو مورد کاربرد *UML*
- ارائه الگوریتمی برای تشخیص مفاهیم و رفتار یک مورد کاربرد
- یک الگوریتم پیشنهاد مورد کاربرد
- ارائه الگوریتمی برای تطبیق نمودار فعالیت *UML*

روش ارائه شده، یک فرآیند کامل برای استفاده مجدد ارائه می‌کند که بطور خاص، مرحله تطبیق مصنوعات بازیابی شده را پشتیبانی می‌کند. این امر یکی از نوآوری‌های مهم این تحقیق می‌باشد و آن را از کارهای مختلفی که تنها به بازیابی مصنوعات امیدبخش می‌پردازند متمایز می‌کند.

ایده حاشیه‌نویسی نمودار فعالیت برای ایجاد قابلیت تطبیق و همچنین ارائه الگوریتم حاشیه-نویسی به همراه جزئیاتی نظیر حل تعارض[۲] جنبه دیگری از نوآوری تحقیق جاری می‌باشد.

معیار شباهت معنایی ارائه شده نیز معیار جدیدی است و تا جایی که ما می‌دانیم تا کنون معیاری برای محاسبه شباهت دو مورد کاربرد *UML* ارائه نشده است. همچنین الگوریتم‌هایی که برای تشخیص رفتار و مفاهیم مورد کاربرد ارائه شده است و ایده استفاده از چاشنی[۳] نیز کاملا جدید هستند.

---

۱ Ontology
۲ Conflict Resolution
۳ Salt

نوآوری مهم دیگر این تحقیق آن است که برای نخستین بار یک رویکرد مبتنی بر نمودار مورد کاربرد برای استفاده مجدد ارائه می‌کند. اگرچه کارهایی وجود دارند که از مورد کاربرد *UML* برای استخراج و تولید انواع دیگر مدل‌ها استفاده می‌کنند اما در همه این کارها، آنچه تحت عنوان مورد کاربرد استفاده می‌شود یا ترکیب توصیف کلی و توصیف جزئی مورد کاربرد، یا تنها توصیف جزئی مورد کاربرد می‌باشد. به بیان دیگر، در این روش‌ها توصیف جزئیات و جریان رویدادهای مورد کاربرد به عنوان ورودی از کاربر گرفته می‌شود در حالیکه روش پیشنهادی این رساله، تنها از نمودار مورد کاربرد که شامل توصیف کلی مورد کاربرد است استفاده می‌کند. با توجه به سادگی ایجاد نمودارهای مورد کاربرد و محبوبیت آن‌ها نزد طراحان، این ویژگی روش پیشنهادی، موجب کاهش هزینه بکارگیری آن می‌شود.

الگوریتم ارائه شده برای تطبیق نمودار فعالیت، یکی دیگر از جنبه‌های نوآوری این تحقیق می‌باشد. همچنین استفاده از خود وب معنایی و نه تنها فناوری‌های وب معنایی[1]بمنظور ارائه دانش لازم جهت خودکارسازی یک فعالیت مهندسی نرم‌افزار، یکی دیگر از جنبه‌هایی است که در کارهای بسیار کمی مورد توجه قرار گرفته است.

۱-۴ ساختار رساله

در این بخش، ساختار ادامه مطالب این رساله توضیح داده می‌شود. با توجه به اینکه راهکار پیشنهادی در این رساله دارای جنبه‌های مختلفی است کارهای زیادی وجود دارند که به نوعی با آن مرتبط هستند. در فصل دوم رساله، در پنج بخش به مرور مهم‌ترین زمینه‌های مرتبط و مهمترین کارهای موجود در هر زمینه پرداخته می‌شود. این زمینه‌ها عبارتند از مهندسی وب، استفاده مجدد از نرم‌افزار، معیارهای شباهت نرم‌افزار، تولید خودکار مدل و مهندسی نرم‌افزار مبتنی بر وب معنایی.

در فصل سوم، جزئیات روش پیشنهادی به همراه الگوریتم‌های مربوطه توضیح داده می‌شود. فصل چهارم رساله نیز به ارزیابی‌های انجام شده بر روی نمونه اولیه‌ای که توسعه داده شده اختصاص دارد. در فصل پنجم به بیان نتیجه‌گیری پرداخته و جهت‌های اصلی کارهای آینده ترسیم شده است.

[1] توضیح این موضوع در بخش ۵-۲-۵ ~~۳-۵۲~~ ارائه شده است.

Formatted: C
Persian (Iran)

۲- مرور کارهای گذشته

با توجه به اینکه راهکار پیشنهادی در این رساله دارای جنبه‌های مختلفی است، کارهای زیادی وجود دارد که به نوعی با آن مرتبط هستند. در این فصل به مرور مهم‌ترین موارد از کارهای مرتبط پرداخته می‌شود.

در بخش ۲-۱ موضوع مهندسی وب بطور اجمالی مرور می‌شود. بخش ۲-۲ به مرور کارهای انجام شده در زمینه استفاده مجدد و بطور خاص استفاده مجدد در سطح مدل اختصاص دارد. با توجه به اینکه روش پیشنهادی شامل معیاری برای محاسبه شباهت معنایی مورد کاربرد *UML* است، در بخش ۲-۳ معیارهای شباهت که برای اجزای مختلف نرم‌افزار ارائه شده‌اند بطور مختصر بررسی شده‌اند. روش‌های ایجاد خودکار مدل نیز در بخش ۲-۴ مرور شده‌اند و در آخر، کاربرد فناوری‌های وب معنایی در مهندسی نرم‌افزار در بخش ۲-۵ بررسی شده است.

۲-۱ مهندسی وب

وب، علاوه بر بستری برای اشتراک و انتشار اطلاعات به عنوان یک سکوی نرم‌افزاری مهم که امکان انتشار و دسترسی توزیع شده به فرآیندهای کسب و کار را فراهم می‌کند نیز مورد توجه است. علیرغم وجود اشتراک، توسعه سیستم‌های تحت وب با سیستم‌های معمولی تفاوت‌های مهمی دارد [21, 22, 23]. در نتیجه توسعه موفق سیستم‌های تحت وب معمولا با چالش‌ها و پیچیدگی‌های فراوانی روبرو است [24, 25] که این امر ناشی از تفاوت‌های تجربه‌های موجود نشان می‌دهد بکارگیری روش‌های و ابزارهای متداول مهندسی نرم‌افزار برای سیستم‌های تحت وب کافی نمی‌باشد و به راهکارها و روش‌های خاصی نیاز است تا توسعه سیستم‌های تحت وب، با کیفیت بالا و با صرف زمان و هزینه پیش‌بینی شده بطور موفقی انجام شود [26, 27].

مهندسی وب[1] که در پاسخ به همین نیاز مطرح شده عبارت است از یک رویکرد مهندسی نرم-افزار که بطور خاص به موضوع طراحی و توسعه سامانمند سیستم‌های تحت وب می‌پردازد [28~~28~~, 29~~29~~]. بطور دقیق‌تر، مهندسی وب را می‌توان بدین شکل تعریف نمود: بکارگیری راهکارهای سامانمند و قابل کمی‌سازی برای انجام مقرون به صرفه تحلیل نیازمندی‌ها، طراحی، پیاده‌سازی، تست، عملیاتی کردن و نگهداری برنامه‌های کاربردی تحت وب با کیفیت بالا [22, 30~~30~~]

اصطلاح مهندسی وب نخستین بار در سال ۱۹۹۶ و در مقاله [31~~31~~] مطرح شد و پس از آن در بسیاری از رویدادها و منابع مربوط به سیستم‌های تحت وب (نشست‌های علمی، کنفرانس‌ها و مستندات) مورد استفاده قرار گرفته و در نتیجه امروزه به عنوان یک عنوان خاص و شناخته شده مطرح است.

در دو دهه اخیر چند روشگان برای مهندسی وب ارائه شده است که همگی بر اصول مشابهی مبتنی هستند. یکی از این اصول، استفاده از مدل‌ها به عنوان عنصر پایه در فرآیند توسعه می‌باشد. به بیان دیگر، توسعه یک برنامه کاربردی تحت وب با ایجاد مدل‌های اولیه که برنامه مورد نظر را از جنبه-های مختلفی توصیف می‌کنند شروع شده و سپس این شماهای مفهومی، طی فرآیندهای مشخصی به مصنوعات نرم‌افزاری مراحل بعد تبدیل می‌شود.

شماهای مفهومی در این روشگان‌ها با مدل‌های مستقل از سکو (*PIM*[2]) که در معماری مبتنی بر مدل (*MDA*[3]) مطرح می‌شوند معادل‌اند. در *MDA*، مدل‌های مستقل از سکو نیز با استفاده از تبدیل‌های مدل به مدل، به مدل‌های خاص سکو (*PSM*[4]) تبدیل می‌شوند، همانطور که شماهای مفهومی ایجاد شده در مهندسی وب نیز با انجام تبدیل‌هایی به مصنوعات مورد نظر تبدیل می‌شوند.

روشگان‌های مختلف مهندسی وب، از نظر اینکه چه ساز و کارهایی برای مدل‌سازی برنامه کاربردی تحت وب ارائه می‌کنند با یکدیگر متفاوت هستند. به عنوان چند نمونه از این روشگان‌ها می‌توان

[1] Web engineering
[2] Platform Independent Model
[3] Model Driven Architecture
[4] Platform Specific Model

به *OOHDM*[۱] [32]، *WebML*[۲] [33]، *UWE*[۳] [34]، *WSDM*[۴] [35] و *NDT*[۵] [36] اشاره کرد.

جنبه‌های مختلفی که برای ایجاد شمای مفهومی یک برنامه کاربردی تحت وب مورد توجه قرار می‌گیرد در روشگان‌های مختلف مهندسی وب، تا حدی با یکدیگر متفاوت است، اما بطور کلی می‌توان این جنبه‌ها را برشمرد:

- محتوا:[۶] مشخص می‌کند چه اطلاعات و مفاهیمی در برنامه مورد نظر ذخیره، پردازش و بازنمایی می‌شود.
- ناوش:[۷] مشخص می‌کند محتوای مورد نظر در قالب چه ساختاری از صفحات و پیوندها ارائه می‌شود.
- نمایش:[۸] واسط کاربری که طرح و اجزای صفحات مختلفی که کاربر می‌بیند را مشخص می‌کند.
- فرآیند:[۹] رفتار برنامه کاربردی و منطق کسب و کار آن را مشخص می‌کند.

هر روشگان مهندسی وب یک زبان صوری برای توصیف جنبه‌های مختلف برنامه کاربردی تحت وب ارائه می‌کند. از این منظر، روشگان‌های مختلف مهندسی وب را می‌توان به گروه‌های زیر تقسیم کرد:

- روشگان‌های داده گرا:[۱۰] که ریشه در سیستم‌های پایگاه داده دارند و عمدتا روش مدل‌سازی *ER*[۱۱] را با افزودن مفاهیم خاصی برای مدل‌سازی برنامه‌های کاربردی تحت وب بکار می‌گیرند. تمرکز اصلی این روش‌ها به مدل‌سازی برنامه‌های کاربردی تحت وب مبتنی بر پایگاه داده

۱ Object-Oriented Hypermedia Design Model
۲ Web Modeling Language
۳ UML-based Web Engineering
۴ Web Site Design Method
۵ Navigational Development Techniques
۶ Content
۷ Navigation
۸ Presentation
۹ Process
۱۰ Data Oriented
۱۱ Entity Relationship

معطوف می‌باشد. به عنوان چند نمونه از این روش‌ها می‌توان به *RMM*[۱] [37~~37~~]، *Hera* [38~~38~~] و *WebML* [33~~33~~] اشاره کرد.

- روشگان‌های فرامتن‌گرا[۲] که بر ویژگی فرامتنی ساختار برنامه‌های کاربردی تحت وب تمرکز دارند و عموما ریشه در حوزه سیستم‌های فرامتنی دارند. روش‌هایی نظیر *HDM*[۳] [39~~39~~]، *W2000* [40~~40~~] که شکل توسعه‌یافته *HDM* است، *HDM-lite* [41~~41~~] و *WSDM* [42~~42~~] را می‌توان به عنوان نمایندگان این گروه معرفی کرد.
- روشگان‌های شیءگرا که عموما مبتنی بر *UML* بوده و به شکل‌های مختلفی به توسعه آن برای مدل‌سازی برنامه‌های کاربردی تحت وب می‌پردازند. روش‌هایی نظیر *OOHDM*[۴] [43~~43~~]، *UWE* [44~~44~~]، *OOWS*[۵] [45~~45~~] و *OO-H*[۶] [46~~46~~] در این دسته قرار می‌گیرند.

در شکل ۲-۱~~شکل ۲-۱~~، روشگان‌های مختلف مهندسی وب از نظر تاریخی نشان داده شده‌اند. در شکل ۲-۲~~شکل ۲-۲~~ نیز جزئیات بیشتری درباره این روش‌ها ارائه شده است. به عنوان مثال، نوع نشانه-گذاری[۷] که توسط هر روش مورد استفاده قرار می‌گیرد، ابعاد مدل‌سازی مورد پشتیبانی آن روش، ابزارهای پشتیبان موجود و در نهایت، قدرت هر یک از این روش‌ها در این شکل نشان داده شده است.

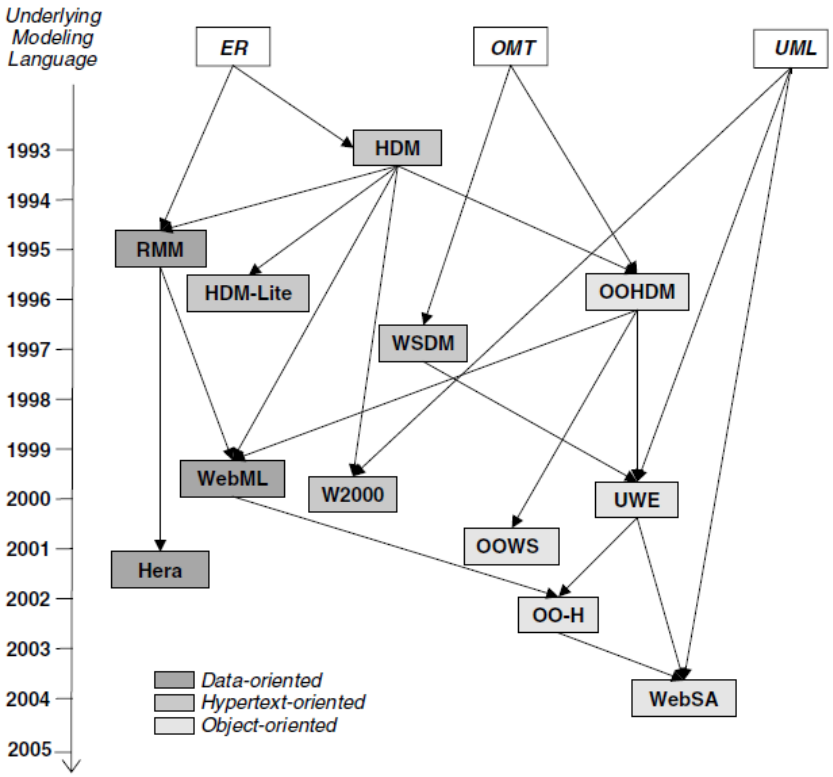
روش *HDM-lite* که توسعه‌یافته *HDM* است، با تمرکز بر خودکارسازی فرآیند توسعه و قابلیت تولید خودکار برنامه‌های کاربردی تحت وب ایجاد شده است. روش *W2000* که باز هم توسعه‌یافته *HDM* است، برنامه کاربردی تحت وب را با مرکزیت ساختار فرامتنی و از دیدگاه کاربر مدل‌سازی می‌کند. در روش *RMM* اساس مدل‌سازی در توسعه مدل‌های *ER* است و فرآیند تدریجی برای بهبود مدل‌ها تا ایجاد مدل‌های نهایی ارائه شده است. روش دیگری که مبتنی بر مدل‌های *ER* است روش *Hera* می‌باشد که از نشانه‌گذاری‌های *RMM* استفاده می‌کند. روشگان *WebML* یک روش ساده و قابل فهم و یک زبان مدل-

---

۱ Relationship Management Methodology
۲ Hypertext Oriented
۳ Hypertext Design Model
۴ Object Oriented Hypermedia Design Method
۵ Object Oriented Web Solutions
۶ Object Oriented Hypermedia
۷ Notation

سازی قوی و کامل برای مدل‌سازی برنامه‌های کاربردی تحت وب که داده‌های زیادی دارند ارائه می‌کند. ابزار *WebRatio* نیز از این رویکرد پشتیبانی کرده و امکان مدل‌سازی و ایجاد خودکار کد منبع از روی مدل‌ها را فراهم می‌کند. در رویکرد *WSDM*، تمرکز به سمت نیازمندی‌های کاربران و استفاده از آن‌ها به عنوان پیش‌ران مدل‌سازی می‌باشد. روشگان *UWE* رویکردی مبتنی بر نشانه‌گذاری *UML* و قابلیت‌هایی نظیر بررسی صحت مبتنی بر متامدل ارائه می‌کند. روش *OO-H* یکی از رویکردهای به نسبت جدیدتر است که سعی دارد مزایای *WebML*، *OOHDM* و *UWE* را ترکیب کند. در این روش از ابزاری با نام *VisualWADE* برای تولید خودکار کد به روش مبتنی بر مدل استفاده می‌شود.

در ادامه، با توجه به ارتباط تحقیق حاضر با روشگان *UWE*، توضیح مختصری از این روشگان ارائه می‌شود.



شکل ۲-۱ تاریخچه روش‌های مختلف مدل‌سازی در مهندسی وب [47~~47~~]

Modeling Method	Modeling Paradigm	Notation	Evolving	Requirements Modeling	Content Modeling	Hypertext Modeling	Presentation Modeling	Customization Modeling	Structure and Behavior	Process / Approach	Tool Support	Generation	Strengths
HDM-lite	HT	ER + own notation	×	×	×	✓	✓	×	s	own	generation tools	auto	process for model transformation, automatic generation
Hera	DB	ER + RMM+ own notation	✓	×	✓	✓	✓	✓	s + b	own	authoring & generation tool	semi	model-driven development
OO-H	OO	UML + own notation	✓	✓	✓	✓	×	pers	s + b	own	modeling- & generation tool	auto	tool for automatic generation
OOHDM	OO	UML + own notation	✓	✓	✓	✓	✓	pers	s + b	own	×	×	powerful concepts for contextual navigation, personalization
OOWS	OO	UML + own notation	✓	✓	✓	✓	✓	×	s + b	own	modeling- & generation tool	auto	advanced (commercial) tool for automatic generation
RMM	DB	ER + own notation	×	×	✓	✓	✓	×	s	own	authoring tool	semi	hypertext modeling based on ER-model, predefined process
UWE	OO	UML	✓	✓	✓	✓	✓	pers	s + b	RUP	extended UML tool & generation tools	semi	UML-based method, model-driven development, aspect-oriented customization
W2000 (HDM)	HT	UML	✓	✓	×	✓	✓	pers	s	×	extended UML-tool	×	user-centric hypertext modeling
WebML	DB	ER, UML	✓	✓	✓	✓	×	pers	s + b	own	modeling- & generation tool	auto	well-elaborated notation, database integration, generation
WS DM	HT	own notation	✓	×	✓	✓	×	×	s + b	own	×	×	user-centric approach for analysis

✓	supported
×	not supported

pers	personalization

s	structure modeling
b	behavior modeling

RUP	Rational Unified Process
own	own process model / approach

auto	automatic generation
semi	semi-automatic generation

DB	data-oriented
HT	hypertext-oriented
OO	object-oriented

شکل ۲-۲ روش‌های مختلف مدل‌سازی در مهندسی وب [4747]

۲-۱-۱ روشگان UWE

روشگان *UWE* در اواخر دهه ۹۰ معرفی شد و همانطور که از نام آن مشخص است، زبان *UML* در این روشگان نقش مهمی ایفا می‌کند و به نوعی پایه و اساس آن می‌باشد. علت نیز به پذیرش و مقبولیت زبان *UML* که مهم‌ترین و رایج‌ترین زبان مدل‌سازی در مهندسی نرم‌افزار بحساب می‌آید است.

همچنین انعطاف‌پذیری خوب *UML* که بواسطه قابلیت توسعه[۱]آن فراهم شده و امکان تعریف زبان‌های خاص حوزه[۲]را فراهم می‌کند نیز حائز اهمیت است. در نهایت، وجود ابزارهای قوی که از مدل‌سازی به زبان *UML* پشتیبانی می‌کنند قابلیت استفاده این روشگان را افزایش می‌دهد. علاوه بر انتخاب *UML* به عنوان پایه زبان مدل‌سازی، روشگان *UWE* از *XMI* نیز به عنوان قالب تبادل مدل[۳] از *MOF*[۴] برای متامدل‌سازی و از قواعد توسعه مبتنی بر مدل که در معماری مبتنی بر مدل *OMG* ارائه شده نیز استفاده می‌کند. همچنین از *QVT*[۵] به عنوان زبان تبدیل مدل‌ها و از *XML* به عنوان قالب بازنمایی استفاده شده است.

روشگان *UWE* شامل ۳ عنصر اصلی است.

۱. یک زبان مدل‌سازی که نشانه‌گذاری لازم برای توصیف مدل‌ها را فراهم می‌کند و در قالب یک پروفایل *UML* و توسعه ای بر متامدل *UML* تعریف شده است.
۲. یک فرآیند توسعه مدل‌گرا که رویه مدل‌سازی و ترتیب فعالیت‌های لازم را مشخص می‌کند.
۳. ابزارهای پشتیبان برای مهندسی برنامه‌های کاربردی تحت وب

جنبه‌های مختلفی که در فرآیند مدل‌سازی *UWE* پوشش داده می‌شود عبارتند از: محتوا، ساختار ناوش، نمایش، فرآیند و تطبیق‌پذیری[۶] در مورد آخر که به تطبیق دادن سیستم برای زمینه‌های خاص و یا شخصی‌سازی ویژگی‌های آن مرتبط است، از تکنیک‌های مدل‌سازی جنبه‌گرا[۷]استفاده می‌شود.

گام اول در فرآیند توسعه *UWE*، توصیف نیازمندی‌ها است که با استفاده از مدل نیازمندی‌ها بیان می‌شود. نیازمندی‌ها را می‌توان در سطوح مختلف توصیف کرد. روشگان *UWE* دو سطح دانه‌بندی[۸] را برای مدل‌سازی نیازمندی‌ها پشتیبانی می‌کند: ابتدا یک توصیف کلی از قابلیت‌ها ایجاد می‌شود که با

---

۱ Extendability
۲ Domain Specific Language (DSL)
۳ Model Exchange Format
۴ Meta Object Facility
۵ Query/View/Transformation
۶ Adaptivity
۷ Aspect-Oriented
۸ Granularity

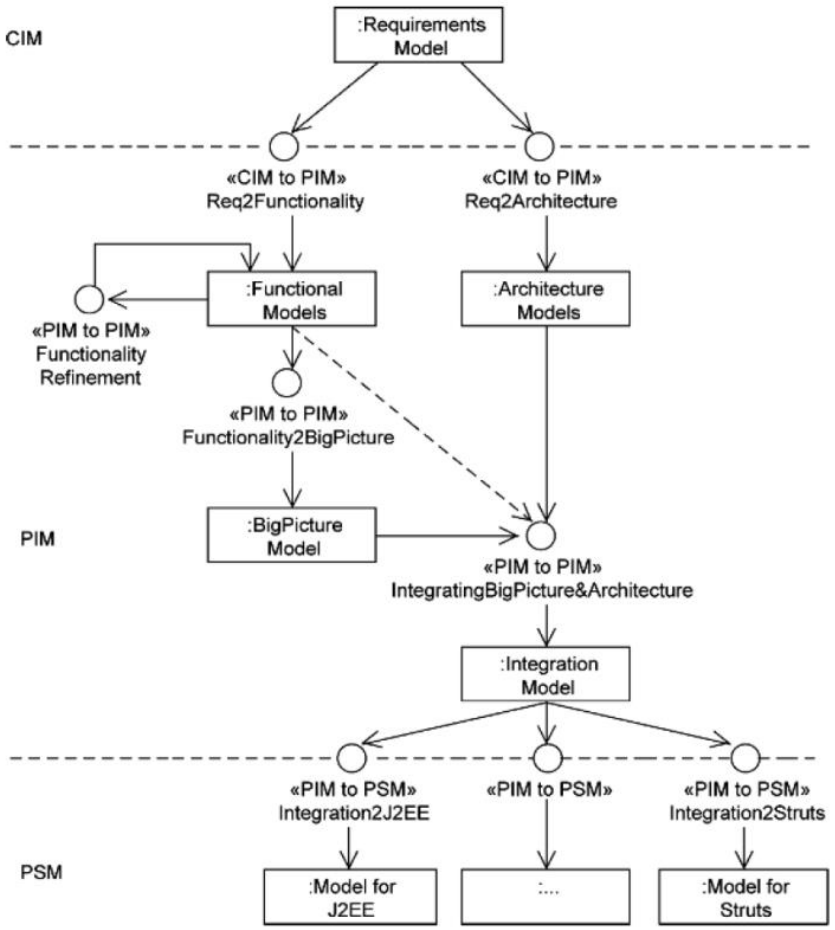
استفاده از نمودار مورد کاربرد[1] *UML* انجام می‌شود. سپس توصیف دقیق‌تر موارد کاربرد با استفاده از نمودار فعالیت *UML* ایجاد می‌شود.

پس از توصیف نیازمندی‌ها، نوبت به توصیف مدل‌های تحلیل[2] می‌رسد که مبنای مدل‌های طراحی[3] و بطور خاص مدل محتوای[4] برنامه کاربردی می‌باشد. هدف از ایجاد مدل محتوا، توصیف تصویری اطلاعات مربوط به حوزه برنامه کاربردی می‌باشد، اطلاعاتی که عمده محتوای سیستم را مشخص می‌کنند. اشیایی که در توصیف دقیق و جزئی موارد کاربرد وجود دارند بطور طبیعی نماینده‌ای برای ایجاد یک نوع موجودیت در مدل محتوا می‌باشند. مدل محتوا با استفاده از نمودار کلاس[5] *UML* نمایش داده می‌شود.

در گام بعد بر اساس تحلیل نیازمندی‌ها و مدل محتوا، ساختار ناوش برنامه کاربردی مدل‌سازی می‌شود. پس از ایجاد مدل‌های ناوش، هر یک از گره‌های موجود در این مدل‌ها که از نوع فرآیند هستند به یک مدل فرآیند جدا و مستقل تبدیل می‌شود. برای مدل‌سازی این فرآیندها نیز از نمودار فعالیت *UML* استفاده می‌شود. در آخر، مدل نمایش برنامه کاربردی ایجاد می‌شود که دید انتزاعی از واسط کاربری برنامه ارائه می‌کند. این مدل نیز بر اساس مدل ناوش ایجاد می‌شود و بجای بیان جزئیات واسط کاربری، نظیر رنگ و اندازه قلم‌ها یا موقعیت دقیق تصاویر، به توصیف ساختار پایه واسط کاربری می‌پردازد،

فرآیند توسعه *UWE* یک فرآیند مبتنی بر مدل است که از تبدیل بین مدل‌ها بخوبی پشتیبانی می‌کند و ویژگی اصلی آن امکان توسعه سامانمند و نیمه‌خودکار برنامه کاربردی تحت وب می‌باشد. از دیدگاه تبدیل‌های مدل به مدل، فرآیند توسعه *UWE* در شکل ۲-۳~~شکل ۲-۳~~ در قالب یک نمودار فعالیت دارای *stereotype* نشان داده شده است. در این شکل، مدل‌ها در قالب اشیا و تبدیل‌ها در قالب فعالیت‌های دارای *stereotype* (به شکل دایره) نشان داده شده است.

[1] Use Case
[2] Analysis models
[3] Design models
[4] Content Model
[5] Class Diagram



شکل ۲-۳ تبدیل مدل‌ها در فرآیند توسعه UWE [3434]

همانطور که در شکل ۲-۳شکل ۲-۳ دیده می‌شود، فرآیند تبدیل مدل‌ها از مدل نیازمندی‌ها شروع می‌شود که در معماری مبتنی بر مدل (*MDA*) در حکم مدل مستقل از محاسبات (*CIM*[۱]) می‌باشد. مدل‌های مستقل از سکو (*PIM*) از روی مدل نیازمندی‌ها استخراج می‌شود. مجموعه مدل‌های طراحی، جنبه‌های مختلف برنامه کاربردی را بازنمایی می‌کند که شامل محتوا، ساختار ناوش، فرآیندهای کسب و کار، نمایش و تطبیق‌پذیری می‌باشد. در گام بعد، این نماهای مختلف با هم یکپارچه‌سازی شده تا تصویر کامل از مدل برنامه کاربردی مورد نظر ایجاد شود. این نمای کامل و یکپارچه برای اعتبارسنجی و همچنین ایجاد مدل‌های وابسته به سکو (*PSM*) استفاده می‌شود.

[۱] Computational Independent Model

۲-۱-۲ نتیجه‌گیری

در این قسمت به توضیح ارتباط تحقیق انجام شده با موضوع مهندسی وب و روشگان‌های آن پرداخته می‌شود. کارهای زیادی در حوزه مهندسی وب انجام شده و روشگان‌ها و ابزارهای پشتیبان خوبی توسعه داده شده است. با این حال، بر اساس مطالعات انجام شده، تا کنون در روشگان‌های مهندسی وب، کار خاصی در رابطه با استفاده مجدد از مدل‌ها انجام نشده است و بیشتر توجه معطوف به بهبود روش-های مدل‌سازی و یا تکنیک‌های تبدیل مدل به مدل بوده است. دیدگاه این رساله آن است که روشگان-های موجود از نظر فرآیند مدل‌سازی به بلوغ خوبی رسیده‌اند و روش کاملا جدید برای مدل‌سازی نیاز نیست. در عوض، چنانچه بتوان قابلیت استفاده مجدد را در نخستین مراحل فرآیند مدل‌سازی یکی از همین روشگان‌های موجود فراهم کرد، می‌توان با کاهش هزینه و زمان ایجاد مدل‌های اولیه، فرآیند مدل-سازی را بهبود داد. در نتیجه در این تحقیق، رویکردی مبتنی بر استفاده مجدد از مدل ارائه شده است که می‌تواند با یک روشگان مهندسی وب مناسب، نظیر *UWE*، تلفیق شود.

از آنجا که در روشگان *UWE*، مدل نیازمندی‌ها در قالب نمودار مورد کاربرد و نمودارهای فعالیت توصیف می‌شود، روش ارائه شده می‌تواند بخوبی با این روشگان همراه شود. یعنی اگر مخزن معنایی مدل‌ها شامل مدل‌هایی باشد که بر اساس روشگان *UWE* ایجاد شده‌اند، برای مدل‌سازی یک برنامه کاربردی جدید با استفاده از روشگان *UWE*، پس از توصیف کلی نیازمندی‌های عملیاتی آن در قالب نمودار مورد کاربرد، می‌توان این نمودار را به عنوان ورودی به روش پیشنهادی این رساله داد و با هزینه خیلی کمی نسخه‌های اولیه نمودارهای فعالیت مورد نیاز را بدست آورد. بدین ترتیب مدل‌سازی نیازمندی‌های عملیاتی در *UWE* ساده و هزینه این کار کم می‌شود.

روشگان‌های دیگری هستند که از تکنیک‌های دیگری برای توصیف مدل نیازمندی‌ها استفاده می‌کنند و بنابراین تلفیق روش ارائه شده با آن روشگان‌ها مناسب نمی‌باشد. به عنوان نمونه روشگان *NDT* از ترکیب نمودار مورد کاربرد و قالب‌های متنی دارای ساختار، روشگان *WSDM* از توصیف زبان

طبیعی و نمودارهای خاصی به اسم نمودار وظیفه[1] استفاده می‌کند. در [33]، روشگان‌های مختلف مهندسی وب از دیدگاه تکنیک مورد استفاده برای توصیف مدل نیازمندی‌ها مورد بررسی قرار گرفته است و نتیجه بررسی‌های انجام شده نشان از این دارد که روشگان *UWE* از دیدگاه انطباق با رویکرد توسعه مبتنی بر مدل از مقبولیت خوبی برخوردار است. همچنین تکیه *UWE* به استانداردهای موجود باعث می‌شود یادگیری آن برای طراحان نرم‌افزار کاری ساده باشد و استفاده از آن در محیط‌های توسعه مختلفی که امکان مدل‌سازی *UML* را فراهم می‌کنند میسر گردد. بنابراین، انتظار می‌رود بعلت قابلیت همراهی با روشگان *UWE*، رویکرد پیشنهادی این رساله از نظر قابلیت استفاده و مفید بودن نیز امیدبخش باشد.

۲-۲ استفاده مجدد از نرم‌افزار

استفاده مجدد از نرم‌افزار به معنای استفاده از مصنوعات نرم‌افزاری موجود برای ایجاد یک نمونه نرم‌افزار جدید می‌باشد و هدف آن، افزایش کیفیت و بهره‌وری در فرآیند توسعه نرم‌افزار است. استفاده مجدد، یکی از قدیمی‌ترین مقوله‌ها در حوزه مهندسی نرم‌افزار است [48, 49]. به عنوان مثال می‌توان از [50] به عنوان یکی از پیشگامان این عرصه یاد کرد که با طرح ایده‌هایی نظیر مدارات مجتمع نرم‌افزار[2] بر اهمیت استفاده مجدد و رویکرد صنعتی در تولید نرم‌افزار تأکید کرده است. استفاده مجدد، بواسطه مزایایی که فراهم می‌کند به یکی از مهمترین زمینه‌های تحقیقاتی در حوزه مهندسی نرم‌افزار تبدیل شده است [51]. برخی از این مزایا که در کارهای متعددی به آنها اشاره شده است عبارتند از: کاهش هزینه تولید، افزایش قابلیت اطمینان و سرعت تولید، بهبود کیفیت و بهره‌وری [52, 53, 54].

تحقیقات نشان داده است که پیاده‌سازی یک فرآیند منظم برای استفاده مجدد در چرخه توسعه نرم‌افزار، امری پیچیده است که نه تنها نیازمندی‌های فنی، بلکه مسائل غیر فنی و عوامل انسانی مختلفی نیز در آن دخیل هستند [55]. عوامل اصلی که به موفقیت یا شکست یک برنامه استفاده مجدد در ابعاد سازمانی منجر می‌شود، در [56] مورد بررسی قرار گرفته است. به عنوان نمونه، یکی از دلایلی

[1] Task Diagram
[2] Software Integrated circuits

ریشه‌ای برای شکست استفاده مجدد، این سوء برداشت است که "به صرف استفاده از فناوری توسعه شیءگرا می‌توان به موفقیت در استفاده مجدد رسید"، یا این برداشت نادرست که "ایجاد یک مخزن از مصنوعات نرم‌افزاری خوب، لزوما به موفقیت در استفاده مجدد منجر می‌شود".

برخی از موانع موفقیت در استفاده مجدد عبارتند از: عدم وجود ابزارهای پشتیبان مناسب، عدم تعهد و التزام کافی در سطح مدیریت سازمان برای اجرای فرآیندهای استفاده مجدد، ناتوانی در تخمین و اندازه‌گیری دقیق هزینه، سود و بازگشت سرمایه در صورت سرمایه‌گذاری روی برنامه استفاده مجدد [57, 58, 59]. در رابطه با مورد آخر، کارهایی در زمینه ارائه مدل‌های اقتصادی برای استفاده مجدد ارائه شده است که به تحلیل و تخمین سود و هزینه روش‌های استفاده مجدد اختصاص دارند [60, 61, 62, 63].

در یک برنامه منظم برای استفاده مجدد، لازم است سطح بالایی از سازگاری بین قابلیت‌ها و ویژگی‌های اجزایی که مورد استفاده مجدد قرار می‌گیرند و ویژگی‌های سیستمی که این اجزا در آن استفاده می‌شوند برقرار باشد. با توجه به اینکه هم این اجزا و هم سیستم مورد نظر پیوسته در معرض تغییر و تکامل هستند، حفظ چنین سازگاری کار پیچیده‌ای است. در [64] سعی شده با توسعه متامدل *UML*، قابلیت پشتیبانی از قراردادهای استفاده مجدد[1] [65, 66] را به آن افزود. بدین ترتیب می‌توان تناقض و ناسازگاری بین ویژگی‌های سیستمی که عمل استفاده مجدد را انجام می‌دهد و قابلیت‌های اجزایی که مورد استفاده مجدد قرار می‌گیرند را بطور خودکار تشخیص داد.

در [67]، قواعد طراحی استفاده مجدد[2] که معمولا برای پشتیبانی از طراحی برای استفاده مجدد[3] استفاده می‌شوند، مورد بحث و بررسی قرار گرفته‌اند. منظور از طراحی برای استفاده مجدد، طراحی اجزای نرم‌افزاری به نحوی که سطح بالایی از قابلیت استفاده مجدد را داشته باشند می‌باشد. همچنین عوامل مختلفی که بر طراحی با استفاده مجدد[4] اثرگذار است بطور تجربی مورد بررسی قرار

[1] Reuse Contract
[2] Reuse Design Principles
[3] Design for Reuse
[4] Design with Reuse

گرفته است. به عنوان مثال، رابطه بین قابلیت استفاده مجدد یک جزء نرم‌افزاری و اندازه آن بر مبنای تعداد سطر کد منبع (*SLOC*[۱]) ارزیابی شده است. بطور مشابه، در [6868] نیز نویسندگان به بررسی رابطه بین پیچیدگی یک جزء نرم‌افزاری، قابلیت استفاده مجدد آن و قواعد طراحی استفاده مجدد که در ساخت آن جزء مورد استفاده قرار گرفته است، پرداخته‌اند.

استفاده مجدد در سطح کد منبع یکی از ابتدایی‌ترین و رایج‌ترین شکل‌های استفاده مجدد است [6969]. کارهای زیادی در این زمینه انجام شده که عموما به ایجاد یک مخزن شامل اطلاعات کدهای منبع پروژه‌های مختلف پرداخته و ساز و کاری برای جستجو و بازیابی نمونه کدها از مخزن فراهم می‌کنند [1717, 1919, 1414, 1515]. چنانچه راهکار ارائه شده امکان جمع‌آوری خودکار نمونه کدهای موجود از بستر وب را نیز داشته باشد، معمولا از اصطلاح موتور جستجوی کد منبع برای این کارها استفاده می‌شود. به عنوان چند نمونه می‌توان به [7070, 7171, 7272] و همچنین سیستم‌های Koders[۲]، Krugle[۳] و Google Code Search[۴] اشاره کرد.

در برخی کارها نظیر [7373, 7474] نیز که از اصطلاح سیستم پیشنهاد دهنده کد منبع[۵] استفاده می‌کنند راهکاری ارائه شده است که وقتی برنامه‌نویس در حال نوشتن کد جدیدی می‌باشد، با توجه به شباهت کدهای نوشته شده او با کدهای موجود در مخزن، نمونه کدهای جدیدی که برای برنامه‌نویس مفید خواهند بود، مثلا دستورات بعدی که لازم است در برنامه نوشته شود، تشخیص داده و به او پیشنهاد شود [1818, 7575].

با این حال استفاده مجدد محدود به کد منبع نیست و در مراحل مختلف چرخه توسعه و نگهداری نرم‌افزار و درباره مصنوعات مختلف نرم‌افزاری قابل طرح است [7676]. به عنوان مثال، در [7777] به ۱۰ جنبه مختلف پروژه‌های نرم‌افزاری که قابلیت استفاده مجدد دارند اشاره شده است. برخی

---

۱ Source Line Of Code
۲ http://code.ohloh.net/
۳ http://krugle.org/
۴ http://www.google.com/codesearch
۵ Source Code Recommender

از جنبه‌ها عبارتند از: مؤلفه‌های نرم‌افزاری[1] مستندات سیستم، مدل‌های طراحی و داده‌های آزمون قابل طرح است.

محققان زیادی این مسأله را تأیید کرده‌اند که هرچه استفاده مجدد در سطوح بالاتر و بر روی مصنوعات سطح بالاتر نظیر مدل‌های طراحی انجام شود، بهتر است [78, 79]. همچنین، هرچه استفاده مجدد، در مراحل زودتر در چرخه توسعه نرم‌افزار تحقق یابد، مزایای بیشتری فراهم می‌کند [80, 81, 82]. به عنوان مثال، استفاده مجدد در سطح مدل‌های طراحی در مقایسه با استفاده مجدد در سطح کد منبع، مزایای بیشتری بهمراه دارد [83, 84]. از سوی دیگر، استفاده مجدد در سطح طراحی در مقایسه با سطح کد منبع با چالش‌های بیشتری روبرو است، چرا که پیچیدگی‌هایی نظیر تنوع و تعداد زیادی مدل‌هایی که در این سطح مطرح هستند و همچنین سطح انتزاع بالاتر که در نتیجه برخی جزئیات بطور دقیق بیان نشده‌اند، عمل استفاده مجدد را با مشکل مواجه می‌کنند [85, 86].

یکی دیگر از رویکردهای متداول به استفاده مجدد، توسعه نرم‌افزار مبتنی بر مؤلفه[2]است که به استفاده مجدد از مؤلفه‌های نرم‌افزاری، شامل کد منبع، ماژول‌ها، کتابخانه‌های برنامه‌نویسی و ابزارهای تجاری آماده[3]می‌پردازد [87, 88]. در این رویکرد بر خلاف روش‌های کلاسیک توسعه نرم‌افزار نظیر روش آبشاری، فرآیند توسعه بر مبنای یکپارچه‌سازی مؤلفه‌های نرم‌افزاری مناسب استوار است [89]. یک مؤلفه مناسب، قابلیتی مشخص را بر اساس یک مدل خاص پیاده‌سازی کرده و واسط مشخصی برای تعامل با آن پیاده‌سازی ارائه می‌کند [91]. در [90]، انگیزه‌ها، مشکلات و مزایای توسعه نرم‌افزار مبتنی بر مؤلفه مورد بحث قرار گرفته است. همچنین در [92]، نتایج یک تحقیق تجربی درباره وضعیت و میزان بکارگیری عملی تکنیک‌های استفاده مجدد مبتنی بر مؤلفه گزارش شده است. میزان و شکل استفاده مجدد از مؤلفه‌های نرم‌افزاری در پروژه‌های متن‌باز جاوا در [93] مورد تحقیق و بررسی قرار گرفته است که نتایج قابل توجهی دارد. به عنوان مثال، در ۹۰٪ پروژه‌های مورد بررسی، مؤلفه‌های

[1] Software Components
[2] Component-based Software Development
[3] Commercials off the shelf

نرم‌افزاری خارجی مورد استفاده مجدد قرار گرفته است. همچنین در بین پروژه‌های مورد بررسی هیچ موردی با کمتر از ۴۰٪ استفاده مجدد به سبک جعبه سیاه، با احتساب کتابخانه استاندارد جاوا، دیده نمی‌شود. حتی بدون احتساب کتابخانه استاندارد جاوا، تنها ۲۵٪ از پروژه‌ها کمتر از ۱۰٪ استفاده مجدد به سبک جعبه سیاه داشته‌اند و مقدار میانه نیز در حدود ۴۰٪ می‌باشد. در نهایت، با افزایش حجم پروژه-ها، عموما نرخ استفاده مجدد کاهش می‌یابد.

موفقیت استفاده مجدد از مؤلفه‌های نرم‌افزاری، تا حد بسیار زیادی به امکان دستیابی و بازیابی مؤثر مؤلفه‌هایی با مشخصات مورد نظر بستگی دارد [94]. در نتیجه کارهای زیادی در زمینه جستجو، بازیابی و انتخاب مؤلفه‌های نرم‌افزاری برای استفاده مجدد انجام شده که از ایده‌ها و تکنیک‌های مختلفی نظیر الگوریتم‌های تقریبی[1] [95]، روش‌های محاسبات تکاملی [96]، فناوری‌های وب معنایی [16, 99, 111]، تکنیک‌های خوشه‌بندی و طبقه‌بندی [97] استفاده می‌کنند.

با توجه به اهمیت زیاد استفاده مجدد از مدل‌های طراحی [98, 7]، کارهای مختلف و متعددی نیز در این حوزه انجام شده است. یک گروه از کارها به موضوع الگوهای طراحی [99] پرداخته و زمینه تشخیص الگوهای طراحی موجود در نرم‌افزار [100, 101]، کمک به طراحان در انتخاب الگوی طراحی مناسب [102] و همچنین توصیف و به اشتراک‌گذاری نمونه‌های الگوهای طراحی [103, 104] را فراهم می‌کنند.

در دسته دیگری از کارها، راهکارهایی تحت عنوان موتور جستجوی مدل ارائه شده است [105, 106] که به ایجاد یک موتور جستجو برای بازیابی مدل‌ها اختصاص دارند. به عنوان مثال *Moogle* [13] یک موتور جستجوی مدل است که می‌تواند مدل‌های *UML* یا هر زبان مدل‌سازی دیگری که دارای یک متامدل خوش‌تعریف است را نمایه‌سازی کرده و امکان جستجو بر روی آن‌ها را

[1] Approximation Algorithms

فراهم کند. عمل جستجو همانند نمایه‌سازی، بر اساس متامدل مورد نظر انجام می‌شود. در *Moogle* امکان جستجو بر حسب کلمه کلیدی و همچنین جستجوی مبتنی بر ویژگی[۱] وجود دارد.

در [107~~107~~]، زبان *MoScript* به عنوان یک زبان خاص دامنه (*DSL*[۲]) ارائه شده است که با استفاده از آن کاربران می‌توانند به روشی مبتنی بر پرس و جو، مدل‌ها را در یک مخزن مدل ذخیره کرده و یا از آن بازیابی کنند. در [12~~12~~]، مدل‌های موجود در قالب گراف‌های برچسب‌دار بازنمایی و در یک مخزن ذخیره می‌شوند. سپس با دریافت یک گراف برچسب‌دار ورودی که نماینده یک مدل است، مشابهت گراف ورودی با گراف‌های موجود محاسبه شده و شبیه‌ترین موارد به عنوان خروجی‌های روال جستجو به کاربر ارائه می‌شود. نویسندگان ادعا کرده‌اند که راهکار پیشنهادی‌شان امکان پشتیبانی از هر زبان مدل‌سازی که دارای متامدل است را دارا می‌باشد. به عنوان دو نمونه دیگر از کارهای انجام شده برای استفاده مجدد از مدل‌ها می‌توان به [108~~108~~, 109~~109~~, 110~~110~~] اشاره کرد.

در بین کارهایی که به استفاده مجدد از مدل اختصاص دارند، انواع مختلفی از مدل‌ها، مانند مدل فرآیندهای جریان کاری [109~~109~~]، نمودار توالی[۳] [111~~111~~]، نمودار کلاس [10~~10~~] و مدل فرآیندهای کسب و کار[۴] [112~~112~~, 113~~113~~] مورد توجه قرار گرفته است.

با توجه به اهمیت مدل نیازمندی‌ها[۵] در فرآیند توسعه نرم‌افزار، استفاده مجدد از این نوع مدل‌ها به نسبت بقیه انواع مدل‌ها از مزایای بیشتری برخوردار است چرا که مدل نیازمندی‌ها جزء اولین مدل-هایی است که در توسعه یک نرم‌افزار ایجاد می‌شود [114~~114~~, 115~~115~~]. همچنین، نرم‌افزارهای مختلف دارای قابلیت‌های مشابه و در نتیجه نیازمندی‌های مشابه هستند. استفاده مجدد، امکان بهره‌گیری از این شباهت‌ها در اولین مراحل توسعه را فراهم کرده و موجب کاهش هزینه و انرژی طراحان و توسعه-دهندگان می‌شود. با توجه به اینکه مورد کاربرد *UML*[۶]، یکی از رایج‌ترین ابزارها برای توصیف نیازمندی-

---

۱ Faceted Search
۲ DomainSpecific Language
۳ Sequence Diagram
۴ Business Process Model
۵ Requirements models
۶ Use Case

های عملیاتی نرم‌افزار می‌باشد، محققان زیادی بر اهمیت استفاده مجدد مبتنی بر مورد کاربرد، یعنی راهکاری که از مورد کاربرد به عنوان پیشران اصلی در فرآیند استفاده مجدد استفاده می‌کند، تأکید کرده‌اند [116~~116~~, 117~~117~~, 118~~118~~].

بدلیل اهمیت UML، که در عمل به زبان استاندارد برای مدل‌سازی در حوزه نرم‌افزار تبدیل شده است، کارهای زیادی در زمینه استفاده مجدد از انواع مدل‌های *UML* انجام شده است. به عنوان مثال، [86~~86~~] یک روش مبتنی بر هستان‌شناسی برای بازیابی نمودارهای کلاس *UML* ارائه کرده است. در این روش، از هستان‌شناسی برای محاسبه شباهت معنایی اجزای مدل‌ها و همچنین برای توسعه پرس و جو[۱] استفاده شده است. در [119~~119~~] نیز روش دیگری برای استفاده مجدد از نمودارهای کلاس *UML* ارائه شده است.

کارهای موجود در زمینه استفاده مجدد از مدل‌های *UML* در [120~~120~~] مورد بررسی قرار گرفته‌اند. این بررسی، تکنیک‌های مختلفی که برای مقایسه یک مدل ورودی با مدل‌های موجود در مخزن مدل‌ها مورد استفاده قرار می‌گیرد را به چهار گروه تقسیم کرده‌اند: ۱) تکنیک‌های مبتنی بر انطباق گراف[۲] [121~~121~~, 122~~122~~, 123~~123~~]، ۲) تکنیک‌هایی که از هستان‌شناسی برای محاسبه شباهت معنایی استفاده می‌کنند [86~~86~~, 124~~124~~]، ۳) تکنیک‌های مبتنی بر استنتاج موردی[۳] [125~~125~~, 126~~126~~] و ۴) تکنیک‌های مبتنی بر بازیابی اطلاعات[۴](نظیر روش‌هایی که از خوشه‌بندی و طبقه‌بندی استفاده می‌کنند) [85~~85~~, 128~~128~~, 127~~127~~].

موضوع استفاده مجدد هنوز برای تقریبا نیمی از انواع نمودارهای *UML* مورد توجه قرار نگرفته و کار خاصی در این زمینه انجام نشده است، اما نمودارهای کلاس، نمودار توالی و نمودارهای مورد کاربرد سه نوع نموداری هستند که بیش از بقیه نمودارهای *UML* مورد توجه بوده‌اند [120~~120~~].

۱ Query Expansion
۲ Graph Matching
۳ Case Based Reasoning (CBR)
۴ Information Retrieval

نمودار مورد کاربرد، در بین مدل‌های *UML* از اهمیت خاصی برخوردار است و استفاده از آن، بخصوص برای توصیف نیازمندی‌های نرم‌افزار، خیلی رایج است [117, 129]. نمودار مورد کاربرد و مورد کاربردها جزء مدل‌هایی هستند که در اولین مراحل چرخه توسعه نرم‌افزار ایجاد می‌شوند. در نتیجه، ارائه یک فرآیند استفاده مجدد بر اساس مورد کاربرد، بسیار حائز اهمیت است [130]. کارهایی که در این زمینه انجام شده است را می‌توان به دو گروه تقسیم کرد. در گروه اول، روشی برای استفاده مجدد از توصیف متنی مورد کاربرد ارائه می‌شود. توصیف متنی، جریان رویدادهای اصلی مورد کاربرد را بهمراه اطلاعات دیگری نظیر پیش‌شرط‌ها و جریان جایگزین را توصیف می‌کند. در گروه دوم، روشی برای استفاده مجدد از خود نمودار مورد کاربرد و نه توصیف متنی مورد کاربردها، ارائه می‌شود. در حالیکه بیشتر کارها در زمینه استفاده مجدد مبتنی بر مورد کاربرد به گروه اول تعلق دارد [120]، اما اهمیت ارائه روشی برای استفاده مجدد از مورد کاربرد ها به شکلی که در نمودار مورد کاربرد ظاهر می‌شوند، نیز مورد تأکید قرار گرفته است [131].

به عنوان یک نمونه از کارهای مربوط به گروه اول می‌توان به [132, 133] اشاره کرد. در [133] روشی برای بازیابی توصیف متنی مورد کاربرد بر اساس ساختار مورد کاربرد ارائه شده است. در این روش ۱۲ ویژگی برای توصیف متنی مورد کاربرد در نظر گرفته شده است، مثلا نام مورد کاربرد، هدف آن، و نام مورد کاربردهایی که از آن مشتق می‌شوند. برای هر مورد کاربرد موجود در مخزن، کلماتی که در مقادیر این ۱۲ ویژگی ظاهر شده‌اند استخراج شده و با استفاده از تکنیک‌های متداول بازیابی اطلاعات نمایه شده‌اند. سپس در مرحله بازیابی، کاربر پرس و جوی مورد نظر خود را با وارد کردن کلماتی که می‌خوهد در هر یک از این ۱۲ ویژگی وجود داشته باشند مشخص می‌کند. همچنین کاربر باید مقدار وزن هر یک از این ۱۲ ویژگی را مشخص کند (عددی صحیح از ۱ تا ۵). در پایان، یک معیاز شباهت متنی وزن‌دار استفاده می‌شود تا پرس و جوی کاربر را با مورد کاربردهای موجود در مخزن مقایسه کرده و شبیه‌ترین موارد را بازیابی کند. نکته مهم آن است که این مقایسه، صرفا بر اساس شباهت متنی است و از تکنیک‌های شباهت معنایی استفاده نشده است.

به عنوان نمونه‌ای دیگر، در [127~~127~~] روشی برای بازیابی توصیف متنی مورد کاربردهای *UML* ارائه شده است. در این روش، شباهت دو مورد کاربرد با محاسبه شباهت دنباله رویدادهای مربوط به هر یک از دو مورد کاربرد بدست می‌آید. برای این منظور از شباهت بردار کلمات که از نام‌های جریان رویدادهای هر یک از دو مورد کاربرد بدست می‌آید استفاده می‌شود. همچنین از یک روش خوشه‌بندی استفاده می‌شود تا مورد کاربردهای موجود در خوشه‌هایی سازماندهی شوند تا در زمان مقایسه و بازیابی، پیچیدگی زمانی کاهش یابد. عمل خوشه‌بندی بطور دستی و توسط فرد خبره انجام می‌شود که این امر باعث می‌شود نتایج خوشه‌بندی دقیق و قابل اطمینان باشند. اما همانطور که نویسندگان [127~~127~~] نیز اذعان کرده‌اند این کار هزینه قابل توجهی دارد. همچنین این احتمال وجود دارد که نتایج خوشه‌بندی بر اساس دیدگاه فرد خبره، به سمت خاصی متمایل[1] شود.

به عنوان یک نمونه از کارهای گروه دوم، در [131~~131~~] از استنتاج موردی برای بازیابی نمودار مورد کاربرد استفاده شده است. مهمترین جزء این روش یک معیار برای محاسبه شباهت دو نمودار مورد کاربرد است. این معیار شامل دو بعد است. در بعد اول، شباهت متنی دو نمودار مورد کاربرد محاسبه می‌شود که برای این کار از انطباق قطعی نام‌های مورد کاربردها، عامل‌ها و نام سیستم‌های مورد نظر استفاده می‌شود. در بعد دوم، روابط مختلفی که در نمودار مورد کاربرد وجود دارند مورد توجه قرار گرفته و شباهت دو نمودار بر اساس نوع و جهت این روابط محاسبه می‌شود.

به عنوان نمونه‌ای دیگر، روشی در [124~~124~~] برای استفاده مجدد از نمودار مورد کاربرد ارائه شده است. در این روش، نمودار مورد کاربرد ورودی پردازش شده و اطلاعات عامل‌ها و مورد کاربردهای آن استخراج شده و به بازنمایی مبتنی بر هستان‌شناسی تبدیل می‌شود. این بازنمایی معنایی در مخزنی ذخیره شده و در نهایت، کاربر می‌تواند از طریق فرم جستجویی که واسط کاربری فراهم می‌کند، به جستجوی مبتنی بر کلمه کلیدی بپردازد و نمودارهایی که شامل کلمه کلیدی خاصی هستند را بازیابی کند.

[1] Bias

با توجه به اینکه یک روش استفاده مجدد کامل شامل چهار مرحله است: بازنمایی، بازیابی، تطبیق و الحاق [88]، مرور کارهای موجود در زمینه استفاده مجدد و بخصوص استفاده مجدد از مدل‌ها، نشان می‌دهد بیشتر کارها فقط دو مرحله اول را پوشش می‌دهند. به بیان دیگر، بیشتر کارها صرفا به بازیابی مصنوعاتی که می‌توانند مورد استفاده مجدد قرار گیرند می‌پردازند، اما امکانات خاصی برای آنکه مصنوعات بازیابی شده، برای ایجاد مصنوعات جدید استفاده شوند ارائه نمی‌کنند و این موضوع را به خود کاربر واگذار می‌کنند [16, 95, 96, 11, 106, 134]. در نتیجه می‌توان گفت عمده کارهای انجام شده در زمینه استفاده مجدد، مرحله تطبیق را مورد پوشش قرار نمی‌دهند در حالیکه دیدگاه ما این است که اساسا در این مرحله است که استفاده مجدد محقق می‌شود.

یک استثنا در این باره، رویکرد مهندسی خط تولید نرم‌افزار [135] است که با ارائه راهکاری سامانمند تحت عنوان مهندسی برنامه کاربردی[1] فعالیت دوم را نیز مورد توجه و پشتیبانی قرار می‌دهد. با این حال، یکی از انتقادات به مهندسی خط تولید وابستگی به دامنه است، چرا که اساسا به توسعه سریع برنامه‌هایی از یک خانواده و یک دامنه اختصاص دارد و از استفاده مجدد مابین دامنه[2]پشتیبانی نمی‌کند. در این رویکرد، تمرکز بر توصیف و بازنمایی صریح و یکپارچه وجوه اشتراک و وجوه تنوع یک دسته از برنامه‌های کاربردی مخصوص یک دامنه خاص می‌باشد [136]. فرآیند توسعه در مهندسی خط تولید، شامل دو مرحله اصلی است: مهندسی دامنه[3]و مهندسی برنامه کاربردی. در مرحله مهندسی دامنه، دامنه مربوطه مورد بررسی قرار گرفته و برنامه‌های کاربردی آن دامنه بطور دقیق تحلیل و بطور رسمی مدل-سازی می‌شوند. یکی از خروجی‌های مهم این مرحله، مدل ویژگی‌ها[4]است که ویژگی‌ها و جنبه‌های مختلف (نقاط اشتراک و تنوع) برنامه‌های کاربردی آن دامنه را به شکل رسمی و صریح، توصیف و مستندسازی می‌کند. در مرحله بعد، یعنی مهندسی برنامه کاربردی، یک نمونه خاص از برنامه‌های کاربردی دامنه مورد نظر ایجاد می‌شود. این کار با استفاده از مدل ویژگی‌ها و با مشخص کردن اینکه

[1] Application Engineering
[2] Cross-Domain
[3] Domain Engineering
[4] Feature model

کدامیک از ویژگی‌ها و جنبه‌های توصیف شده در مدل ویژگی‌ها برای برنامه کاربردی مورد نظر لازم هستند، انجام می‌شود. به بیان دیگر، خروجی مرحله مهندسی دامنه، قابلیت استفاده مجدد را فراهم می-کند و به کمک آن می‌توان تنها با تغییر پیکربندی و انتخاب ویژگی‌ها و جنبه‌های مورد نیاز، نمونه برنامه‌های کاربردی متنوع و جدیدی ایجاد کرد [137~~137~~].

در [138~~138~~]، نتایج یک تحقیق درباره مشکلات، مزایا و معایب استفاده مجدد در خط تولید نرم‌افزار به بحث گذاشته شده است. این تحقیق، مبتنی بر مصاحبه و پرسش و پاسخ با افراد خبره این حوزه می‌باشد و بطور کلی نتیجه‌اش این است که استفاده مجدد، جزء جدایی‌ناپذیر مهندسی خط تولید است، اما موفقیت آن به خبرگی زیاد در دانش دامنه، مهارت‌های فنی در تکنیک‌های مهندسی خط تولید، وجود ابزارهای پشتیبان مناسب، کارکنان باانگیزه و تعهد جدی مدیریت مربوطه بستگی دارد. همچنین استفاده مجدد در مهندسی خط تولید نرم‌افزار معایبی نیز دارد که یکی از مهم‌ترین آن‌ها عبارت است از پیچیدگی. به بیان دیگر، استفاده مجدد موجب ایجاد وابستگی بین واحدهای سازمانی مختلف که پیش از آن بصورت خودمختار عمل می‌کرده‌اند می‌شود و این وابستگی، هزینه هماهنگی و یکپارچه‌سازی را افزایش می‌دهد. از دیگر معایب آن است که استفاده مجدد می‌تواند به کاهش نوآوری و خلاقیت منجر شود.

رساله جاری روشی برای استفاده مجدد از مدل‌ها ارائه می‌کند. این روش در مقایسه با روش‌های موجود دارای چند تفاوت است، که مهمترین آن‌ها این است که روش ارائه شده یک فرآیند کامل استفاده مجدد است که بطور خاص، از مرحله تطبیق نیز پشتیبانی می‌کند.

تفاوت دیگر آن است که روش ارائه شده، امکان استفاده مجدد را بر اساس مورد کاربرد *UML*، در شکل گرافیکی آن و در قالب نمودار مورد کاربرد و نه توصیف متنی، فراهم می‌کند. کارهای کمی در این زمینه انجام شده است. با توجه به اینکه ورودی این روش شامل توصیف جزئی مورد کاربرد نمی‌باشد، در مقایسه با روش‌های دیگری که توصیف جزئی مورد کاربرد را هم به عنوان ورودی از کاربر می‌گیرند هزینه استفاده از روش ارائه شده خیلی کمتر است.

اگرچه کارهای موجود، نظیر [131~~131~~]، معیارهایی برای محاسبه شباهت دو نمودار مورد کاربرد یا توصیف جزئی دو مورد کاربرد ارائه می‌کنند، اما مطالعات ما نشان داد که هیچ کاری در زمینه محاسبه شباهت دو مورد کاربرد در شکل گرافیکی آن ارائه نشده است. بهمین دلیل، معیاری که روش پیشنهادی برای محاسبه شباهت دو مورد کاربرد ارائه می‌کند یکی دیگر از جنبه‌های نوآوری این روش است. لازم به ذکر است که این معیار، برخلاف برخی کارهای موجود نظیر [124~~124~~] که صرفا بر اساس مقایسه متنی و جستجوی مبتنی بر کلمه عمل می‌کنند، هم جنبه‌های متنی و هم روابط یک مورد کاربرد را در نظر می‌گیرد و البته از معیار شباهت معنایی، بجای شباهت متنی ساده، استفاده می‌کند.

روش ارائه شده، از الگوریتم‌های جدیدی برای حاشیه‌نویسی مدل‌ها، تشخیص مفاهیم و رفتار مورد کاربرد و الگوریتم تطبیق نمودار فعالیت استفاده می‌کند که هر یک از جنبه‌های نوآوری روش پیشنهادی هستند. این دو مورد در واقع، برای تأمین نیازمندی‌های هفتم و هشتم که در [139~~139~~] برای یک مخزن مدل معرفی شده است در نظر گرفته شده است.

در [140~~140~~] شش دسته از معیارها و مدل‌ها برای استفاده مجدد از نرم‌افزار مورد بررسی قرار گرفته و یک طبقه‌بندی از انواع استفاده مجدد از نرم‌افزار ارائه شده است. بر اساس این طبقه‌بندی، روش پیشنهادی این رساله دارای ویژگی‌های زیر است:

- حوزه توسعه[۱] خارجی (زیرا مدل‌های مورد استفاده مجدد به پروژه‌ها و سیستم‌های خارجی تعلق دارند)
- مدل تغییرات: جعبه سفید (الگوریتم ارائه شده برای تطبیق، تغییرات داخلی انجام می‌دهد)
- حوزه دامنه[۲] افقی (می‌توان مدل‌هایی که به حوزه‌های متفاوت تعلق دارند را مورد استفاده مجدد قرار داد)
- مدیریت: موردی[۳] (روش استفاده مجدد ارائه شده را نمی‌توان یک روش منظم به حساب آورد.

[۱] Development Scope
[۲] Domain Scope
[۳] Ad-hoc

۱-۲-۲ نتیجه‌گیری

مرور کارهای انجام شده در حوزه استفاده مجدد از نرم‌افزار نشان می‌دهد کارهای زیادی در رابطه با جستجو و بازیابی مصنوعاتی که پتانسیل استفاده مجدد را دارند انجام شده است، اما اینکه مصنوعات بازیابی شده چگونه مورد استفاده مجدد قرار می‌گیرند خیلی کمتر مورد توجه بوده است و نحوه اجرای این کار به کاربر نهایی واگذار شده است. به عنوان مثال، روش ارائه شده در [10~~10~~] به بازیابی نمودارهای کلاس *UML* اختصاص دارد، اما این که نمودارهای بازیابی شده چگونه می‌توانند در تولید نمودارهای جدید برای سیستم‌های مشابه استفاده شوند مورد بحث قرار نگرفته است.

دیدگاه این رساله آن است که یک فرآیند کامل استفاده مجدد باید هم بازیابی مصنوعات امیدبخش و هم استفاده از آن‌ها در تولید مصنوعات جدید را شامل شود. بدین ترتیب تحقیق جاری سعی در ارائه یک فرآیند کامل استفاده مجدد برای مدل‌های نیازمندی‌های عملیاتی دارد.

اهمیت استفاده مجدد از مدل‌ها و بطور خاص استفاده مجدد بر اساس مورد کاربرد، امری است که ضرورت آن از دیرباز مورد تأیید قرار گرفته است [7~~7~~, 117~~117~~]. با این حال نکته مهم آن است که در کارهایی که در زمینه استفاده مجدد مبتنی بر مورد کاربرد انجام شده است، توصیف دقیق موارد کاربرد مورد نظر است. به بیان دیگر، توصیف جزئیات مورد کاربرد، چه در قالب نمودار فعالیت و یا بوسیله قالب-های متنی ساخت‌یافته، به عنوان ورودی دریافت می‌شود. این موضوع از دیدگاه تحقیق جاری امری نامطلوب است، چرا که هزینه بکارگیری روش را افزایش می‌دهد. به بیان دیگر دیدگاه این تحقیق آن است که اگر کاربر ملزم باشد هم توصیف کلی و هم توصیف دقیق برنامه کاربردی را به عنوان ورودی ارائه کند، با توجه به اینکه تولید نمودارهای فعالیت یا قالب‌های متنی ساخت‌یافته کاری زمان‌بر است، انرژی و زمان زیادی از کاربر گرفته می‌شود. حال آن که بهتر است بجای آن که کاربر یک به یک نمودارهای فعالیت را از ابتدا ایجاد کند، تنها توصیف کلی را در قالب نمودار کاربرد، که ایجاد آن سخت و زمان‌بر نیست، ارائه کرده و سپس نسخه اولیه نمودارهای فعالیت لازم را دریافت کند و با صرف وقت اندکی، این نسخه‌های اولیه را به آنچه مدنظرش است تغییر دهد.

ایده دریافت نمودار مورد کاربرد و تولید خودکار نمودارهای فعالیت مربوط به آن از طریق تطبیق نمودارهای فعالیت مربوط به موارد کاربرد مشابه، ایده جدیدی است که در این تحقیق ارائه و دنبال شده است.

با توجه به اهمیت مدل نیازمندی‌های عملیاتی و اینکه می‌توان از این مدل برای استخراج انواع دیگر مدل‌ها استفاده کرد [141~~141~~]، چون خروجی روش ارائه شده در این رساله عبارت است از مدل نیازمندی‌های عملیاتی، در نتیجه می‌توان این روش را به همراه برخی روش‌های موجود استفاده کرد. به بیان دیگر چنانچه روش پیشنهادی بتواند با استفاده مجدد از مدل‌های موجود، مدل نیازمندی‌های عملیاتی را با کارایی خوبی ایجاد کند، خروجی حاصل از آن می‌تواند به عنوان ورودی برای تکنیک‌ها و روش‌هایی که مدل‌های دیگر را از روی مدل نیازمندی‌ها استخراج می‌کنند مورد استفاده قرار گیرد و کل فرآیند مدل‌سازی تا حد زیادی خودکارسازی شود.

۳-۲ معیارهای شباهت نرم‌افزار

روش پیشنهادی در این رساله شامل یک معیار برای محاسبه شباهت دو مورد کاربرد *UML* می‌باشد. بهمین دلیل در این بخش، مرور مختصری بر روی معیارهای شباهت نرم‌افزار انجام می‌شود. در کارهای موجود، معیارهای مختلفی برای محاسبه شباهت انواع مختلفی از مصنوعات نرم‌افزاری ارائه شده است. این معیارها در حوزه‌ها و با اهداف گوناگونی ارائه شده‌اند، به عنوان مثال:

- تشخیص کپی‌های نرم‌افزاری[۱] منظور از کپی‌های نرم‌افزاری، قطعاتی از نرم‌افزار است که در چند نقطه از نرم‌افزار بطور صریح تکرار شده‌اند. کپی‌های نرم‌افزار، بخصوص در سطح کد منبع، موجب کاهش سودمندی و افزایش هزینه‌های نگهداری و توسعه می‌شوند [142~~142~~]. در [143~~143~~] روشی برای استفاده در حوزه توسعه مبتنی بر مدل ارائه شده که با استفاده از یک معیار شباهت مبتنی بر گراف، به تشخیص خودکار کپی‌ها در مدل‌های بزرگ می‌پردازد.

[۱] Software clones

- محاسبه تفاضل[۱] و مدیریت نسخه[۲] بدلیل آنکه بسیاری از فعالیت‌های مهندسی نرم‌افزار نیاز به همکاری و تعامل افراد مختلف دارند و نیز توسعه نرم‌افزار فرآیندی تدریجی است، مکانیزم‌هایی برای محاسبه تفاوت و تفاضل دو نسخه متوالی از یک مصنوع نرم‌افزاری مورد نیاز است. روش-های مختلفی برای محاسبه تفاضل دو مدل *UML* ارائه شده است [144~~144~~] که تعدادی از این کارها در [145~~145~~] مرور شده است. همانطور که در [145~~145~~] عنوان شده است، بیشتر کارهای موجود بر محاسبه تفاضل نمودارهای کلاس *UML* تمرکز دارند. به عنوان مثال، [146~~146~~] الگوریتمی برای محاسبه تفاضل دو نمودار کلاس *UML*، بر اساس ویژگی‌های متفاوتی نظیر نام، *sterotype*، نوع و امضای اجزای مختلف نمودار کلاس ارائه کرده است.
- استفاده مجدد از نرم‌افزار. یک مؤلفه مهم در هر روش استفاده مجدد، معیاری است که از آن برای محاسبه شباهت یک مصنوع نرم‌افزاری با مصنوعات نرم‌افزاری موجود در مخزن استفاده می‌شود. در [147~~147~~] روش مورد استفاده برای محاسبه شباهت توصیف نیازمندی‌ها در پروژه *RedSeeDS* توضیح داده شده است. در این پروژه، نیازمندی‌ها در قالب جملاتی به زبان طبیعی و همچنین به شکل سناریوهای مورد کاربرد به یک زبان کنترل شده بیان می‌شوند. شباهت توصیف نیازمندی‌های دو سیستم بر اساس ترکیبی از الگوریتم‌های شباهت جمله و کلمه، مانند [148~~148~~]، و همچنین معیارهای شباهت ساختاری محاسبه می‌شود. در مورد آخر، از روی سناریوهای مورد کاربرد یک بازنمایی گرافی تولید شده و سپس معیارهای شباهت مبتنی بر گراف، با استفاده از ابزار *SiDiff* [144~~144~~] مورد استفاده قرار می‌گیرد.

اندازه‌گیری شباهت نرم‌افزاری، در سطوح مختلفی و برای انواع مختلفی از مصنوعات نرم‌افزاری مورد توجه قرار گرفته است. به عنوان مثال [149~~149~~, 150~~150~~] بر محاسبه شباهت دو سیستم نرم‌افزاری کامل تمرکز دارند، در حالی که [151~~151~~] به محاسبه شباهت دو برنامه و [152~~152~~] به محاسبه شباهت دو مدل می‌پردازد.

---

۱ Difference computation
۲ Version Management

در برخی کارها، برای محاسبه شباهت مدل‌های *UML*، ابتدا آن‌ها را به تعدادی گراف تبدیل کرده و سپس از تکنیک‌های انطباق گراف، نظیر [153~~153~~] استفاده می‌شود. با این حال، همانطور که در [120~~120~~] بحث شده است، بهتر است بجای استفاده از معیارهای عمومی که برای هر نوع مدلی می‌توانند استفاده شوند، برای بازیابی هر نوع مدل *UML*، تکنیک مخصوصی استفاده شود که معنای اجزای آن نوع مدل را مورد توجه قرار دهد.

در [154~~154~~] یک معیار ترکیبی برای محاسبه شباهت معنایی دو مفهوم ارائه شده است. این معیار بر اساس ترکیبی از معیارهای مختلف مبتنی بر گراف (نظیر طول کوتاه‌ترین مسیر بین دو مفهوم در درخت مفاهیم که از *WordNet* [155~~155~~] استخراج شده است، یا چگالی محلی[۱] یا قدرت اتصال[۲] یک لبه) و همچنین معیارهای محتوای اطلاعات[۳] عمل می‌کند.

در [156~~156~~] یک روش استفاده مجدد برای خط تولید نرم‌افزار ارائه شده است. در این روش، مدل‌های نیازمندی‌ها و مدل‌های تحلیل در یک مخزن ذخیره می‌شوند. این مدل‌ها شامل مورد کاربرد، مدل‌های ویژگی‌ها، مدل‌های پویا و ایستا می‌باشند. برای هر مدل در مخزن، همه کلمات آن بازیابی شده و بر اساس معیار رایج TF/IDF نمایه می‌شوند. بعلاوه، یک سازوکار جستجوی مبتنی بر کلمه کلیدی فراهم شده که به کاربر اجازه می‌دهد مدل‌های مورد نظر خود را بر اساس کلمه کلیدی بازیابی کند. این بازیابی بر اساس محاسبه شباهت کسینوسی در فضای برداری، بین کلمات کلیدی و کلمات نمایه انجام می‌شود.

۱-۳-۲ نتیجه‌گیری

در مقایسه با کارهای موجود، تحقیق جاری معیاری ارائه می‌کند که بطور خاص برای محاسبه شباهت دو مورد کاربرد با در نظر گرفتن نمودار مورد کاربرد مربوطه طراحی شده است. این طراحی خاص امکان توجه به جنبه‌ها و اجزای خاص مورد کاربرد را فراهم می‌کند. کار مشابهی در [131~~131~~] ارائه شده

---

۱ Local density
۲ Connect power
۳ Information content

است اما روش ارائه شده در این تحقیق از چند نظر با کارهای موجود متفاوت است. اول آنکه معیار ارائه شده در روش پیشنهادی، بر خلاف معیار [131~~131~~] که از انطباق متنی قطعی استفاده می‌کند، از شباهت معنایی مفاهیم استفاده می‌کند و در نتیجه مشکلات شباهت متنی را ندارد. همچنین معیار ارائه شده، شباهت دو مورد کاربرد را محاسبه می‌کند، برخلاف [131~~131~~] که شباهت دو نمودار مورد کاربرد را محاسبه می‌کند. بعلاوه، یک نوآوری مهم معیار ارائه شده، که نمونه‌ای از توجه به معنای اجزای مورد کاربرد *UML* است، آن است که این معیار بین اجزای مختلف مورد کاربرد، نظیر رفتار و مفاهیم مورد کاربرد، تمایز قائل شده و الگوریتم‌های ساده‌ای برای تشخیص رفتار و مفاهیم مورد کاربرد ارائه می‌کند.

لازم به ذکر است که معیارهای ارائه شده در کارهای موجود برای محاسبه تفاضل مدل‌ها، برای تحقیق جاری قابل استفاده نمی‌باشند. زیرا خروجی این معیارها، لیستی از تفاضل دو مدل ورودی می‌باشد و نه یک مقدار عددی. مشکل چنین معیارهایی آن است که اگر یک مدل جدید M_1 با دو مدل M_2 و M_3 موجود در مخزن مقایسه شود، نمی‌توان نتیجه مقایسه M_1 و M_2 را با نتیجه مقایسه M_1 و M_3 مقایسه نمود. در نتیجه نمی‌توان مشخص کرد که M_2 به M_1 شبیه‌تر است یا M_3 به M_1 شبیه‌تر است. همچنین، هیچیک از معیارهای موجود بطور خاص به محاسبه شباهت دو مورد کاربرد با در نظر گرفتن معنای آن‌ها نمی‌پردازند.

روش پیشنهادی این تحقیق معیاری برای محاسبه شباهت دو مورد کاربرد ارائه می‌کند. این معیار با کارهای دیگر مثل [156~~156~~] متفاوت است، چرا که نه تنها جنبه‌های متنی، بلکه جنبه‌های رابطه‌ای مورد کاربردها را نیز در نظر می‌گیرد. همچنین، معیار ارائه شده به شباهت متنی محدود نیست و از تکنیک‌های شباهت معنایی استفاده می‌کند.

۲-۴ تولید خودکار مدل

در فرآیند توسعه نرم‌افزار، مدل‌ها نقش مهمی ایفا می‌کنند، بخصوص در رویکردهای توسعه مبتنی بر مدل که نقش مدل‌ها در آن‌ها کلیدی است. حال، مشکلی که پیش می‌آید آن است که تولید

مدل‌های دقیق، درست و کامل برای یک برنامه کاربردی، بخصوص برنامه‌های کاربردی تحت وب که از پیچیدگی‌های بیشتری برخوردارند، عملی زمان‌بر، پیچیده و طاقت‌فرسا است. در نتیجه به راهکاری نیاز است که تولید مدل‌ها را بطور خودکار یا نیمه‌خودکار انجام دهد. به همین دلیل، کارهای متعددی در زمینه تولید خودکار مدل از روی مصنوعات نرم‌افزاری مختلف انجام شده است.

چالش پایه‌ای که در تولید خودکار مدل‌ها وجود دارد آن است که باید به شکلی، دانش مورد نیاز برای تولید مدل، به کامپیوتر منتقل ارائه شود. رویکردهای متفاوتی برای حل این چالش وجود دارد. به عنوان نمونه، در [157~~157~~]، روشی ارائه شده که مستندات نیازمندی‌های سیستم را در قالب توصیف متنی دریافت کرده و با استفاده از پردازش‌های زبان طبیعی، آن را به شکل خاصی از گراف تبدیل می‌کند. سپس با استفاده از این گراف میانی و براساس قواعد از پیش تعریف شده، نمودارهای کلاس و نمودار مورد کاربرد سیستم بطور خودکار ایجاد می‌شود. در اینجا، دانش مورد نیاز برای خودکارسازی تولید مدل، در قالب تعدادی قانون که نگاشت بین گراف میانی و مدل‌های نهایی را تعریف می‌کنند توصیف می‌شود.

روش مشابهی در [158~~158~~] ارائه شده که نمودار کلاس *UML* را از توصیف زبان طبیعی سیستم استخراج می‌کند. در این کار، علاوه بر استفاده از قوانین خوش‌تعریف برای نگاشت داده‌های میانی به مدل‌های مقصد، از راهکار دیگری نیز استفاده شده است. با توجه به اینکه پردازش توصیف زبان طبیعی دارای مشکلاتی نظیر ابهام در عبارات، لغات دارای چند معنی و حذف‌های لفظی یا معنوی می‌باشد، برای اینکه روال تولید خودکار مدل بتواند بر این مشکلات غلبه کند به دانش زمینه یا دانش دامنه نیاز دارد. ایده [158~~158~~]، پاسخگویی به این نیاز از طریق استفاده از هستان‌شناسی می‌باشد. به بیان دقیق‌تر، از هستان‌شناسی *RCyc*[1] که حجم زیادی از دانش پس‌زمینه و عمومی را بصورت رسمی و قابل پردازش توسط کامپیوتر بازنمایی کرده است، برای رفع ابهام مستندات استفاده شده است.

رویکرد دیگری که در [159~~159~~] استفاده شده است این است که توصیف سیستم، بجای زبان طبیعی، در قالب یک زبان کنترل شده با گرامر دقیق دریافت شود. بدین ترتیب، با محدود و مقید کردن

[1] CyCorp. ResearchCyc. http://research.cyc.com.

توصیف ورودی، از پیچیدگی‌های ناشی از مشکلات پردازش زبان طبیعی کاسته شده، رویه تولید مدل بهبود یافته و کیفیت مدل‌های خروجی نیز بهتر می‌شود.

در [160~~160~~]، یک مجموعه قوانین خوش‌تعریف برای محدود کردن کاربران در توصیف متنی موارد کاربرد ارائه شده است. ایده این است که با مقید کردن کاربران به رعایت این قوانین، توصیف متنی حاصل، ابهامات و مشکلات کمتری خواهد داشت و در نتیجه پردازش‌های زبان طبیعی با موفقیت بیشتری می‌توانند آن را تحلیل و برای استخراج مدل‌های مقصد از آن استفاده کنند. نتیجه آزمایش‌ها نشان داده است که قوانین ارائه شده، ساده و قابل درک بوده، یادگیری و استفاده از آن‌ها برای کاربران کار سختی نیست و رعایت آن‌ها کیفیت مدل‌های استخراج شده را بهبود می‌دهد.

به دلیل اهمیت مدل‌سازی مورد کاربرد، که معمولا در نخستین مراحل توسعه نرم‌افزار انجام می‌شود [117~~117~~, 161~~161~~, 162~~162~~, 163~~163~~, 164~~164~~]، تحقیقات زیادی بر استخراج خودکار انواع دیگر مدل‌ها از روی مدل‌های مورد کاربرد تمرکز دارند [165~~165~~, 166~~166~~, 167~~167~~, 168~~168~~, 169~~169~~]. به عنوان نمونه، در [170~~170~~] روشی ارائه شده که توصیف نیازمندی‌های سیستم را به شکل قالب‌های متنی ساخت‌یافته هر مورد کاربرد دریافت کرده و نمودارهای ماشین حالت قابل اجرا[1]را از روی آن استخراج می‌کند. در [171~~171~~] نیز کارهای مربوط به این حوزه مورد مرور و بررسی قرار گرفته است.

در کنار روش‌های موردی برای تولید خودکار مدل، یک رویکرد قاعده‌مند در حوزه توسعه مبتنی بر مدل دیده می‌شود که در آن، مدل‌ها به عنوان پیشران فرآیند توسعه از اهمیت اساسی برخوردار می‌شوند [172~~172~~]. در این رویکرد، برخی مدل‌های اولیه به صورت دستی ایجاد شده و سپس از روی آن‌ها و با استفاده از تکنیک‌های تبدیل مدل به مدل، انواع دیگر مدل‌ها و یا نسخه اجرایی، که باز هم یک نوع مدل است، بطور خودکار یا نیمه خودکار تولید می‌شوند [173~~173~~, 174~~174~~]. در حوزه توسعه مبتنی بر مدل، توصیف متنی نیازمندی‌ها از مقبولیت کمی برخوردار است که دلیل آن نیز ابهام، دقت کم و عدم

[1] Executable State Machines

رسمیت[1] کافی زبان طبیعی می‌باشد [175~~175~~]. به همین دلیل، در این حوزه تمرکز بر مدل‌سازی تصویری و نموداری می‌باشد [44].

به عنوان نمونه در [176~~176~~]، ساز و کاری ارائه شده است که توصیف مبتنی بر سناریوی سیستم را در قالب نمودارهای توالی *UML* دریافت کرده و طی یک فرآیند تبدیل مدل به مدل، از روی آن‌ها نمودارهای فعالیت مربوطه را ایجاد می‌کند. این تبدیل، بر اساس مجموعه‌ای از قوانین انجام می‌شود که نحوه نگاشت را برای الگوهای خاص از پیش تعریف شده‌ای تعریف می‌کنند.

۱-۴-۲ نتیجه‌گیری

همانطور که مرور کارهای گذشته نشان می‌دهد، روش‌های مختلفی برای تولید خودکار مدل ارائه شده است. در حالت کلی این روش‌ها یک یا چند نمونه مدل به عنوان ورودی گرفته و بر اساس یکسری قواعد تبدیل، یک یا چند نمونه مدل دیگر تولید می‌کنند. روش‌های مختلف، در نوع مدل‌های ورودی و خروجی و همچنین قواعد مورد استفاده‌شان با یکدیگر متفاوت هستند.

بر اساس مطالعات انجام شده، نقدی که بر کارهای موجود در زمینه تولید خودکار مدل وارد است آن است که این کارها، موضوع استفاده مجدد از مدل‌ها را مورد توجه قرار نمی‌دهند. به عنوان مثال، فرض کنید مدل M_1 به عنوان ورودی به روش A داده شده و مدل M_2 به عنوان خروجی این روش تولید شده است. حال، چنانچه مدل دیگری نظیر M_3 به عنوان ورودی داده شود، مدل خروجی مربوط به آن، بطور کاملا مستقل از مدل‌های M_1 و M_2 که پیش از این تناظرشان برقرار شده است تولید می‌شود. حتی اگر مدل جدید، یعنی M_3 بسیار شبیه مدل M_1 باشد، باز هم در تولید خروجی مربوط به آن، هیچ استفاده‌ای از خروجی مربوط به M_1 نمی‌شود. این امر بدان معناست که روال تولید مدل، صرفا وابسته به قواعد از پیش تعریف شده است و هیچ استفاده‌ای از دانشی که در قالب مدل‌های موجود، که ممکن است تعدادشان خیلی هم زیاد باشد، ذخیره شده است صورت نمی‌گیرد.

[1] Formalism

نتیجه غیرمستقیم این امر آن است که موفقیت روال تولید مدل، به کیفیت، درستی و جامعیت قواعد تبدیل بستگی دارد. تعریف قواعد مناسب، به نوع مدل‌های ورودی و خروجی وابسته است و هرچه مدل‌های ورودی مدل‌های ساده‌تر و جزئیات موجود در آن‌ها کم‌تر باشد، کیفیت و میزان جزئیات مدل-های خروجی نیز محدودتر می‌باشد. به همین دلیل مشاهده می‌شود که در کارهای موجود، برای آنکه بتوان مدل‌های خروجی بهتری تولید کرد، بر الزامات مدل‌های ورودی نیز افزوده می‌شود. به عنوان نمونه، در هیچیک از کارهای موجود، از مدل‌های مورد کاربرد *UML* (به شکل خالص آن) به عنوان ورودی استفاده نشده است، چرا که این مدل‌ها جزئیات خیلی کمی را بیان می‌کنند که نمی‌توان بر اساس آن‌ها قواعد تبدیل مناسبی تعریف نمود. در مواردی هم که از عنوان مورد کاربرد به عنوان مدل‌های ورودی استفاده شده است، نظیر [165~~165~~, 166~~166~~, 159~~159~~, 170~~170~~]، آنچه در واقع مد نظر بوده است توصیف جزئیات مورد کاربرد در قالب نمودار فعالیت یا قالب‌های متنی ساخت‌یافته بوده است.

دیدگاه این رساله آن است که بجای آن که دانش مورد نیاز برای تولید مدل‌ها صرفا در قواعد تبدیل خلاصه شود، بهتر است بخشی از این دانش را در قواعد از پیش تعریف شده بیان نمود و بخشی را نیز از نمونه مدل‌های موجود استخراج نمود. بدین ترتیب هم می‌توان با چالش تعریف قوانین مناسب مقابله کرد و هم از حجم قابل توجه مدل‌های موجود، که می‌توان به عنوان مخزنی از تجربه و دانش مدل‌سازی بدان نگریست، بهره‌برداری کرد.

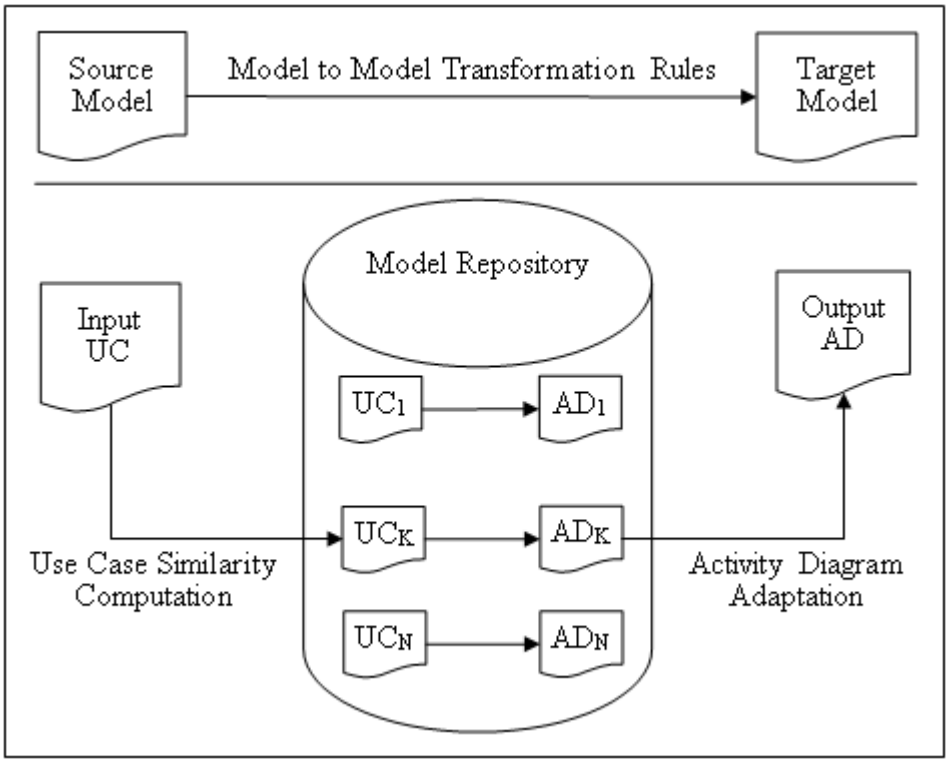
بدین ترتیب، روش پیشنهادی این رساله، به نوعی به تلفیق دو رویکرد استفاده مجدد و تولید خودکار مدل می‌پردازد. همانطور که در بخش ۲-۲ بدان اشاره شد، در اکثر کارهایی که در حوزه استفاده مجدد از مدل انجام شده است، صرفا به جستجو و بازیابی مدل‌هایی که برای استفاده مجدد مناسب هستند پرداخته شده است و رویه خاصی برای اینکه مدل‌های بازیابی شده به مصنوعات جدید تبدیل شوند ارائه نشده است. از آن سو، در حوزه تولید خودکار مدل، به اینکه در تعریف مدل ورودی و تبدیل آن به مدل خروجی می‌توان از مدل‌های موجود استفاده مجدد کرد توجهی نشده است. روش پیشنهادی این رساله، از مزایای کارهای هر دو حوزه بدین ترتیب استفاده می‌کند که اولا با بهره‌گیری از مخزن

مدل‌های موجود، امکان استفاده مجدد از مدل‌های موجود را فراهم می‌کند و دوم آن که با ارائه روشی برای تطبیق مدل‌ها امکان تولید خودکار مدل را نیز فراهم می‌کند.

رویکرد رایج در روش‌های تولید خودکار مدل در شکل ۲-۴~~شکل ۲-۴~~ (قسمت بالای شکل) نشان داده شده است. در این روش‌ها با استفاده از قوانین تبدیل، نگاشتی از فضای مدل مبدأ به فضای مدل مقصد ایجاد می‌شود. با توجه به اینکه نوع مدل مبدأ با نوع مدل مقصد متفاوت است، ایجاد نگاشت مناسب بین دو فضای متفاوت کاری سخت است. مثلا ایجاد یک مجموعه قوانین برای نگاشت نمودار مورد کاربرد *UML* به نمودار فعالیت *UML*، با چالش‌های زیادی همراه است چرا که اساسا این دو نوع مدل، اهداف و سطح انتزاع متفاوتی دارند.

رویکرد مربوط به روش پیشنهادی این رساله در شکل ۲-۴~~شکل ۲-۴~~ (قسمت پایین شکل) نشان داده شده است. طبق این رویکرد، چون مخزن مدل‌ها شامل نمونه‌های فراوانی از زوج‌های [مورد کاربرد، نمودار فعالیت] می‌باشد می‌توان در تولید مدل‌های جدید، از دانش نهفته در تناظر بین این زوج‌ها استفاده کرد و نیازی نیست قواعد از پیش تعریف شده‌ای برای تبدیل مورد کاربرد به نمودار فعالیت ارائه نمود. همانطور که در فصل سوم توضیح داده خواهد شد، روش ارائه شده در این رساله، نمودار مورد کاربرد را به عنوان ورودی گرفته و نمودارهای فعالیت مربوطه را به عنوان خروجی تولید می‌کند. با این حال در هیچ جای روش پیشنهادی، قوانینی کلی برای نگاشت مورد کاربرد به نمودار فعالیت گنجانده نشده است. در عوض، روشی برای جستجو و بازیابی موارد کاربرد مشابه موارد کاربرد ورودی ارائه شده و سپس نمودارهای فعالیت مربوط به موارد کاربرد ورودی، با تطبیق نمودارهای فعالیت مربوط به موارد کاربرد مشابه تولید شده است. در نتیجه اساسا قوانینی برای نگاشت بین دو فضای متفاوت نمودارهای مورد کاربرد و نمودار فعالیت ارائه نشده است بلکه قوانینی برای تطبیق نمودارهای فعالیت به نمودارهای فعالیت جدید ارائه شده است که طبیعتا چون این قوانین، محدود به یک فضا هستند انتظار می‌رود از دقت و توانایی بیشتری برخوردار باشند.

بدین ترتیب، بر اساس توضیحات فوق این امکان فراهم شده که ورودی روش پیشنهادی تنها نمودار مورد کاربرد UML باشد که طبیعتا ایجاد آن برای کاربر هزینه خیلی کمی دارد.



شکل ۲-۴ رویکرد رایج در روش‌های تولید خودکار مدل (بالا)، رویکرد پیشنهادی این رساله (پایین)

۲-۵ مهندسی نرم‌افزار مبتنی بر وب معنایی

فناوری‌های وب معنایی کمک می‌کنند تا معنای داده‌ها به همراه خود داده‌ها به شکلی صوری بازنمایی شود و پردازش، مدیریت و تحلیل داده‌ها و ارتباطات بین داده‌های مختلف توسط ماشین با انعطاف‌پذیری و کارایی بیشتری انجام می‌شود. چند نمونه از مزایای مدل داده وب معنایی در مقایسه با مدل داده‌های متداول‌تر نظیر مدل داده رابطه‌ای ذکر می‌شود عبارتند از: بازنمایی یکسان داده‌ها و شمای داده‌ها، قابلیت پرس و جو به شکل مشابه بر روی داده‌ها و شمای داده‌ها، کمک به درک ساده‌تر شمای داده‌ها بخصوص برای نرم‌افزارهای بزرگ، قابلیت استنتاج خودکار، استقلال داده‌ها از برنامه‌های کاربردی، توصیف دقیق‌تر شمای داده‌ها با استفاده از ساختارهایی که زبان‌های هستان‌شناسی فراهم می‌کنند، انعطاف‌پذیری مدل داده، و سازگاری با وب [177~~177~~].

با توجه به مزایایی که مدل داده وب معنایی فراهم می‌کند، کارهای متعددی در زمینه استفاده از فناوری‌های وب معنایی در مهندسی نرم‌افزار انجام شده است. با توجه به حجم و تنوع داده‌هایی که در فرآیند توسعه نرم‌افزار تولید می‌شوند، استفاده از فناوری‌های وب معنایی کمک می‌کند ماشین‌ها بتوانند این داده‌ها را بهتر پردازش کنند. هر چه دخالت و توانایی ماشین در پردازش این داده‌ها بیشتر شود، امکان مهار پیچیدگی‌های مهندسی نرم‌افزار بیشتر فراهم می‌شود. این امر می‌تواند در خودکارسازی فعالیت‌های مهندسی نرم‌افزار، به عنوان مثال استخراج داده‌های آزمون از روی مدل‌های سیستم، نمایان شود.

به عنوان نمونه‌ای از کارهای موجود در این حوزه می‌توان به مجموعه مقالات کارگاه‌های آموزشی *SWESE*[۱] که از سال ۲۰۰۵ به بعد هر سال برگزار می‌شود یا به مقالات کنفرانس *SEKE*[۲] اشاره نمود.

با توجه به اینکه هستان‌شناسی از مهمترین فناوری‌های وب معنایی است، در [178~~178~~] بطور مفصل، درباره کاربردهای هستان‌شناسی در مهندسی نرم‌افزار و همچنین هستان‌شناسی‌های مختلفی که تا کنون در این حوزه ایجاد شده بحث شده است. در [179~~179~~] دیدگاه مهندسی نرم‌افزار مبتنی بر وب معنایی معرفی و برخی از کاستی‌های تکنیک‌ها و فرآیندهای فعلی مهندسی نرم‌افزار ذکر شده است. نویسندگان [179~~179~~]، همچنین به این موضوع که بکارگیری فناوری‌های وب معنایی چطور می‌تواند در حل این مشکلات کمک کند بحث کرده و معماری یک محیط توسعه یکپارچه برای تحقق این سبک از مهندسی نرم‌افزار را ارائه کرده‌اند.

در مرور کارهای موجود باید به تفاوت فناوری‌های وب معنایی و خود وب معنایی توجه کرد. وب معنایی به لحاظ فنی بر پشته‌ای از فناوری‌ها و پروتکل‌ها استوار است که مهمترین آن‌ها عبارتند از: *URI*، *XML*، *RDF*، زبان‌های هستان‌شناسی نظیر *OWL* و *RDFS*، زبان‌های پرس و جو بر روی داده‌های

[۱] Semantic Web Enabled Software Engineering
[۲] Software Engineering and Knowledge Engineering

معنایی نظیر *SPARQL*، *RDQL* و *SerQL*، مخازن سه‌گانه[1] و موتورهای استنتاج نظیر *Pellet*، *Fact++* و *HermiT*.

نکته مهم آن است که استفاده از این فناوری‌ها برای بازنمایی، ذخیره و بازیابی داده‌ها لزوما به معنای انتشار داده‌های مورد نظر بر روی وب نمی‌باشد. به بیان دیگر ممکن است در یک سیستم، برای بازنمایی داده‌ها از مدل داده وب معنایی استفاده شود اما در نهایت، داده‌ها بر روی یک سیستم بسته قرار داشته باشند و از طریق وب قابل دسترس نباشند. در بسیاری از کارهایی که در این بخش توضیح داده خواهد شد چنین رویکردی به وب معنایی قابل مشاهده است.

وب معنایی که بصورت اسم خاص مورد اشاره قرار می‌گیرد عبارت است از یک فضای داده سراسری بر روی وب که داده‌های آن بر اساس مدل داده وب معنایی بازنمایی شده‌اند و از طریق وب قابل دسترس، بازیابی و استفاده مجدد هستند. در حال حاضر وب معنایی شامل حجم قابل توجهی داده از حوزه‌های مختلف است که توسط نهادها، سازمان‌ها و افراد مختلف منتشر شده است. به عنوان نمونه‌های عینی از منابع تشکیل دهنده وب معنایی می‌توان به این موارد اشاره کرد:

- داده‌های فراوانی که بر روی ابر *LOD*[2] منتشر شده‌اند.
- صفحات وبی که قطعه کدهای *RDFa* یا *Microformat* در آن‌ها تعبیه شده است.
- موتورهای جستجوی معنایی که داده‌های معنایی را از وب جمع‌آوری و نمایه کرده و امکان جستجو بر روی آن‌ها را فراهم می‌کنند.
- هستان‌شناسی‌های منتشر شده بر روی وب که هر یک حوزه خاصی را مدل‌سازی می‌کنند.
- فایل‌های *RDF* که بصورت پراکنده در وب منتشر شده‌اند، نظیر فایل‌های مربوط به پروفایل *FOAF* افراد.

با توجه به بحث فوق، در بین کارهایی که در زمینه مهندسی نرم‌افزار مبتنی بر وب معنایی انجام شده است می‌توان دو رویکرد پایه را مشاهده کرد:

[1] Triple Stores
[2] Linking Open Data

۱. در رویکرد نخست، از فناوری‌های وب معنایی برای بازنمایی داده‌های مربوط به مصنوعات نرم-افزاری استفاده شده است تا از مزایایی نظیر سادگی و افزایش انعطاف‌پذیری در تحلیل و بازیابی این داده‌ها استفاده شود. همچنین قابلیت استنتاج خودکار بر روی داده‌ها نیز مورد توجه می-باشد.

۲. در رویکرد دوم، از خود وب معنایی، یعنی از داده‌های معنایی یا هستان‌شناسی‌هایی که هم اکنون بر بستر وب منتشر شده استفاده شده است. استفاده از این منابع، عمدتا با اهداف زیر انجام می‌شود:

- فراهم کردن دانش لازم برای ماشین تا بتواند در یک مساله خاص تصمیم گیری کرده و عملی را بطور خودکار یا نیمه‌خودکار انجام دهد.
- انجام حاشیه‌نویسی خودکار برای غنی‌سازی توصیف معنایی داده‌ها

در ادامه، کارهای انجام شده بر اساس هر یک از دو رویکرد فوق مرور می‌شود.

۱-۵-۲ رویکرد اول: استفاده از فناوری‌های وب معنایی

در این رویکرد، از فناوری‌های وب معنایی و بطور خاص از هستان‌شناسی و مدل داده *RDF* برای بازنمایی داده‌های مربوط به یک یا چند نوع از مصنوعات نرم‌افزاری استفاده می‌شود تا داده‌ها برای ماشین قابل فهم شده و امکان دسترسی به داده‌ها، تحلیل، بازیابی و استنتاج بر روی آن‌ها بهبود یابد.

در کارهایی که بر اساس این رویکرد انجام شده‌اند، اهداف مختلفی برای استفاده از فناوری‌های وب معنایی وجود دارد. برخی از این اهداف، که در واقع نقاط قوت و مزایای فناوری‌های وب معنایی را نیز مشخص می‌کنند عبارتند از:

- یکپارچه‌سازی داده‌های مربوط به مصنوعات مختلف نرم‌افزاری. به عنوان مثال، در [180~~180~~] و [181~~181~~] از هستان‌شناسی برای یکپارچه‌سازی معنایی برنامه‌های کاربردی سازمانی استفاده شده است.

- خودکارسازی فرآیندها. در کارهای متعددی نظیر [182, 183, 184, 185] از فناوری‌های وب معنایی برای خودکارسازی فعالیت‌های مهندسی نرم‌افزار، نظیر آزمون نرم‌افزار، استفاده شده است.
- افزایش قابلیت استفاده مجدد مصنوعات نرم‌افزاری. یکی از مراحل اصلی استفاده مجدد، بازیابی مصنوعاتی است که قابلیت استفاده مجدد را دارند. با توجه به اینکه نقش هستان‌شناسی‌ها در بهبود جستجو و بازیابی اطلاعات بطور تجربی ثابت شده است [186] می‌توان از آن‌ها برای بهبود فرآیند استفاده مجدد استفاده کرد. به عنوان نمونه، در [10]، روشی مبتنی بر هستان-شناسی برای استفاده مجدد از مدل‌ها و بطور خاص نمودار کلاس *UML* ارائه شده است. در [11]، راهکاری مبتنی بر هستان‌شناسی برای بازیابی مؤلفه‌های نرم‌افزاری مطرح شده است.
- بهبود بازنمایی و قابلیت تحلیل و استنتاج بر روی داده‌ها. در کارهای مختلفی از هستان‌شناسی برای بازنمایی صوری داده‌ها و استنتاج خودکار بر روی داده‌ها استفاده شده است. به عنوان مثال [16] از این الگو در بحث انتخاب مؤلفه‌های نرم‌افزاری استفاده کرده است. در [188] و [187] نیز فناوری‌های وب معنایی با هدف بهبود قابلیت تحلیل و رهگیری گزارش‌های خطا استفاده شده‌اند. در [189, 20] نیز موضوع تحلیل نرم‌افزار با استفاده از فناوری‌های وب معنایی مورد بررسی قرار گرفته که نتایج قابل توجهی نیز ارائه شده است. همچنین در مدل‌سازی صوری نیز از فناوری‌های وب معنایی استفاده شده است [190].
- انتشار و اشتراک داده‌ها. به عنوان نمونه، در [191] از داده‌های پیوندی برای به اشتراک گذاری اطلاعات کد منبع و فراهم کردن امکان جستجو و بازیابی بر روی آن‌ها استفاده شده است. بطور مشابه، [192] نیز از داده‌های پیوندی بمنظور انتشار اطلاعات بسته‌های نرم-افزاری[1] پروژه متن‌باز *Debian* استفاده می‌کند.

[1] Software Packages

- بهبود قابلیت مدیریت، درک و اعتبارسنجی داده‌ها. به عنوان نمونه، استفاده از هستان‌شناسی برای مدیریت پیکربندی نرم‌افزارهای بزرگ در [193] مطرح شده است. در [194] از فناوری‌های وب معنایی برای کاهش مشکلات مربوط به شناخت و نگهداری یک سیستم نرم-افزاری بزرگ استفاده شده است. فناوری‌های وب معنایی همچنین بمنظور توصیف و مدیریت الگوهای طراحی [102, 195]، اعتبارسنجی پالایش فرآیندها[1] [196]، مدیریت و مستندسازی نیازمندی‌های نرم‌افزار [198, 197]، بررسی خودکار صحت و یکپارچگی توصیف نیازمندی‌ها [202, 201, 200]، مدیریت پروژه [199] و احراز اصالت سرویس‌های نرم‌افزاری [203] استفاده شده‌اند.

با معرفی داده‌های پیوندی در سال ۲۰۰۶، وب معنایی بیش از پیش مورد توجه قرار گرفت. می-توان گفت داده‌های پیوندی، مسیر تحقق ایده وب معنایی را هموارتر کرد. به جهت اهمیت این موضوع، داده‌های پیوندی در بخش بعد بطور مختصر معرفی شده و سپس یکی از پروژه‌هایی که به استفاده از داده‌های پیوندی در عرصه مهندسی نرم‌افزار پرداخته است توضیح داده می‌شود.

۲-۵-۱-۱ داده‌های پیوندی

داده‌های پیوندی، در واقع مجموعه‌ای از روش‌های خوب[2] برای انتشار داده‌ها بر روی وب و همچنین ایجاد پیوندهای معنادار بین این داده‌ها می‌باشد. با پیروی از قواعد داده‌های پیوندی می‌توان به تحقق وب داده[3] نزدیک شد. وب داده، بمنزله یک پایگاه داده جهانی عمل کرده که داده‌های مربوط به حوزه‌های مختلف بصورت معنادار و قابل فهم برای کامپیوتر در آن منتشر شده است. قواعد داده‌های پیوندی در سال ۲۰۰۶ و توسط آقای *Tim Berners-Lee* که مبدع وب می‌باشد، معرفی شد [204] و از آن پس منابع و سازمان‌های مختلفی، داده‌های خود را بر اساس این قواعد بر روی وب انتشار داده‌اند.

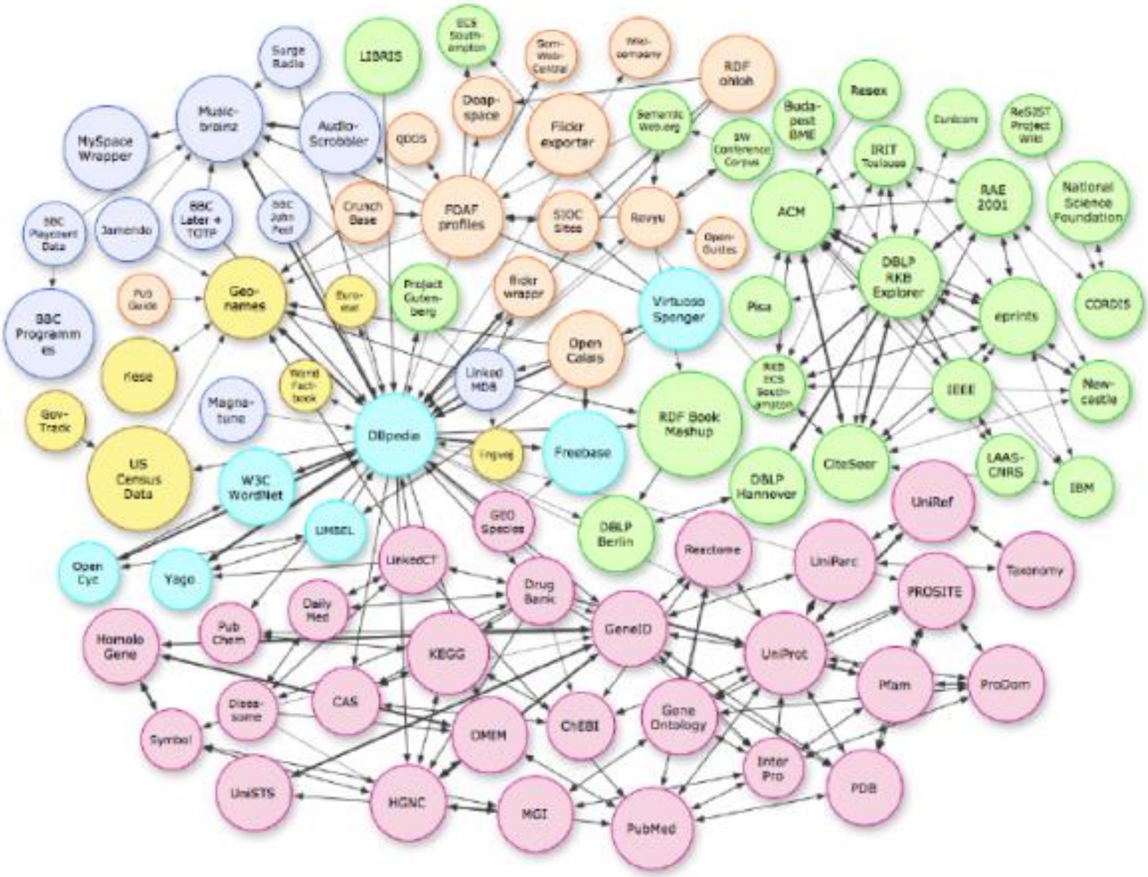
---

[1] Process Refinement
[2] Best Practices
[3] Web of Data

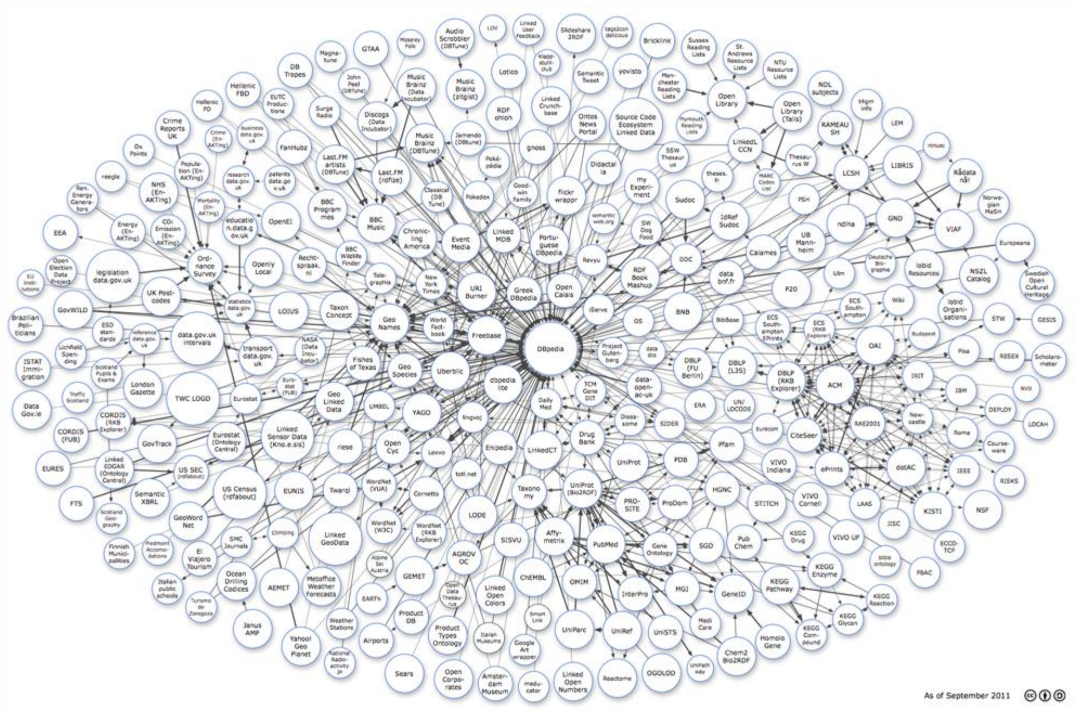
مهم‌ترین نمونه تطبیق و بکارگیری قواعد داده‌های پیوندی را می‌توان در پروژه *LOD*[1] مشاهده کرد که در ژانویه ۲۰۰۷ و تحت حمایت کنسرسیوم *W3C* آغاز شد و با گذشت زمان، با اقبال چشمگیری مواجه گشته و سازمان‌های بزرگی نظیر *BBC*، کتابخانه مجلس آمریکا و دولت بریتانیا نیز به جمع اعضای این پروژه پیوسته‌اند.

برای نمایش حجم و ابعاد موضوعی داده‌های این پروژه می توان ابر *LOD* را مورد توجه قرار داد که به شکل یک گراف جهت‌دار در نظر گرفته می‌شود. هر گره که با یک دایره نشان داده می‌شود نماینده یک مجموعه داده مستقل که بر اساس قواعد داده‌های پیوندی منتشر شده، می باشد. یال موجود بین دو گره به این معناست که بین داده‌های موجود در دو مجموعه داده متناظر با آن گره‌ها، پیوند و ارتباط وجود دارد. ضخامت یال‌ها، تخمینی از تعداد این پیوندها ارائه می‌کند. هر چه تعداد پیوندها بیشتر باشد ضخامت یال نیز بیشتر است. یال‌های دوسویه نیز به معنای وجود پیوند از هر یک از دو منبع به منبع دیگر می‌باشد. شکل ۲-۵شکل ۲-۵، وضعیت ابر *LOD* را در مارچ ۲۰۰۹ و شکل ۲-۶شکل ۲-۶ نیز وضعیت این ابر را در سپتامبر ۲۰۱۱ نشان می‌دهد.

[1] Linking Open Data



شکل ۲-۵. وضعیت ابر LOD در مارچ ۲۰۰۹ [205205]



شکل ۲-۶. وضعیت ابر LOD در سپتامبر ۲۰۱۱ [205205]

محاسبه حجم دقیق داده‌های موجود در وب داده عملی چالش برانگیز است، با این حال، بر اساس آمارهای جمع آوری شده توسط انجمن *LOD*[1]، تخمین زده می‌شود که ابر *LOD* در ماه می ۲۰۰۹ شامل ۴,۷ میلیارد سه‌گانه *RDF* بوده است که از طریق حدود ۱۴۲ میلیون پیوند *RDF* به یکدیگر متصل شده‌اند. در سپتامبر ۲۰۱۱، تعداد سه‌گانه‌ها به بیش از ۳۱ میلیارد و تعداد پیوندها نیز به بیش از ۵۰۰ میلیون افزایش یافته است.

۲-۵-۱-۲ پروژه LD2SD

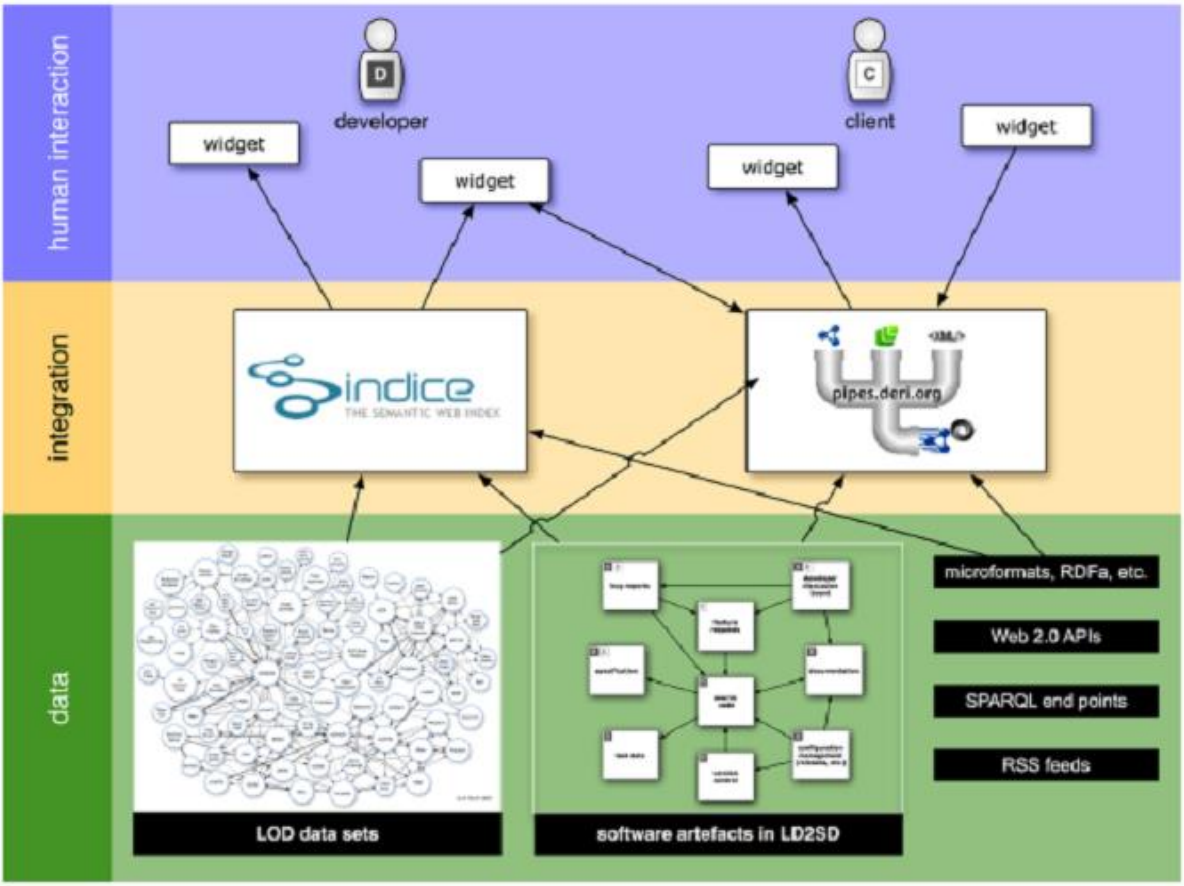
پروژه *LD2SD* یکی از پروژه‌های مرکز تحقیقات *DERI* در کشور ایرلند می‌باشد که یکی از معتبرترین مراکز تحقیقاتی در موضوع داده‌های پیوندی و وب داده می‌باشد.

پروژه *LD2SD* را می‌توان نمونه خوبی از بکارگیری فناوری داده‌های پیوندی در مهندسی نرم‌افزار دانست. بطور ساده، این پروژه با انتشار داده‌های مربوط به مصنوعات مختلف نرم‌افزاری و همچنین روابط این داده‌ها، در قالب داده‌های پیوندی، یک روشگان سبک مبتنی بر وب معنایی برای توسعه نرم افزار ایجاد می‌نماید [187~~187~~].

از نظر پیاده سازی، روشگان *LD2SD* شامل ۳ لایه می‌باشد که در شکل ۲-۷~~شکل ۲-۷~~ نشان داده شده است. در لایه اول، مصنوعات مختلف نرم‌افزاری انتخاب شده و با استفاده از ابزارهای مربوطه، داده‌های مربوط به آن‌ها به صورت داده‌های پیوندی منتشر می‌گردد. در لایه دوم که لایه یکپارچه‌سازی[2] است، با استفاده از فناوری‌هایی نظیر لوله‌های معنایی[3] و نمایه‌سازهای معنایی[4] امکان ذخیره، نمایه-سازی، پرس و جو و استفاده از داده‌های معنایی تولید شده در لایه قبل برای کاربران حرفه‌ای فراهم می-شود. در لایه سوم که لایه تعامل[5] است، ابزارهای معنایی[6] مناسب و همچنین افزونه‌هایی برای محیط

[1] http://lod-cloud.net/state/
[2] Integration layer
[3] Semantic Pipes
[4] Semantic Indexer
[5] Interaction layer
[6] Semantic Widget

توسعه *Eclipse* ایجاد شده که استفاده از داده‌های فوق را برای کاربرانی که متخصص وب معنایی نیستند ساده‌تر می‌کند.



شکل ۲-۷. لایه‌های تعریف شده در روشگان LD2SD [187~~187~~]

۲-۵-۲رویکرد دوم: استفاده از وب معنایی

کارهای محدودی در زمینه استفاده از وب معنایی، به معنای خاص آن، در مهندسی نرم‌افزار انجام شده است. با توجه به حجم قابل توجه داده‌های منتشر شده در وب معنایی، دو هدف کلی برای استفاده از این فضای داده در مهندسی نرم‌افزار می‌توان در نظر گرفت:

- استفاده از وب معنایی به عنوان منبع دانش پس‌زمینه یا دانش یک حوزه خاص. بدین ترتیب می‌توان دانش مورد نظر را بجای آنکه توسط کاربر فراهم شود، بطور خودکار از وب معنایی بدست آورد. این امر می‌تواند به خودکارسازی فعالیت‌های مهندسی نرم‌افزار کمک کند.
- استفاده از وب معنایی برای حاشیه‌نویسی یا برچسب‌زنی منابع. با توجه به اینکه منابع مختلفی در وب معنایی توصیف شده‌اند می‌توان از آن برای حاشیه‌نویسی و برچسب‌زنی منابع در یک سیستم استفاده کرد. بدین ترتیب با ایجاد پیوندهای معنایی بین منابع موجود در سیستم و منابع موجود در وب معنایی می‌توان توصیف منابع موجود در سیستم را بهبود داده و دانش مربوطه را غنی نمود.

در [206~~206~~]، [103~~103~~] و [207~~207~~] پروژه *WOP*[1] معرفی و چارچوبی ارائه شده که از هستان‌شناسی و همچنین از فناوری‌های وب ۲ نظیر نشانک‌گذاری اجتماعی[2]و بلاگ‌ها برای به اشتراک-گذاری دانش مهندسی نرم‌افزار بین توسعه‌دهندگان نرم‌افزار استفاده می‌کند. ابتدا یک هستان‌شناسی برای توصیف الگوهای طراحی ارائه شده است [195~~195~~] و سپس ابزارهایی در قالب افزونه‌هایی برای نرم-افزار *Eclipse* ایجاد شده است. از طریق این ابزارها کاربر، که یک توسعه‌دهنده نرم‌افزار است، الگوی طراحی مورد نظرش را مشخص می‌کند. سپس بطور خودکار از طریق سرویس‌هایی نظیر *SWOOGLE*، منابع وب معنایی جستجو می‌شود تا منابعی که الگوی طراحی مورد نظر کاربر را براساس هستان‌شناسی پایه توصیف می‌کند بازیابی شود. پس از آن، به کمک منابع وب ۲، نظیر پایگاه *del.icio.us* رتبه و میزان مقبولیت هر یک از این منابع استخراج می‌شود. بعد از ارائه نتایج، کاربر منبعی را که مناسب‌تر می‌داند انتخاب کرده و به کمک یک افزونه دیگر، توصیف مربوطه را بارگذاری می‌کند. در نهایت یک افزونه دیگر با استفاده از توصیف رسمی بارگذاری شده، پروژه کاربر را جستجو کرده و به اکتشاف نمونه‌های آن الگوی طراحی در پروژه کاربر می‌پردازد. افزونه‌های دیگری نیز ارائه شده که با استفاده از آن‌ها کاربر می‌تواند نظرش را درباره توصیف بارگذاری شده اعلام نموده و یا به آن امتیاز بدهد.

[1] Web of Patterns
[2] Social bookmarking

به عنوان نمونه‌ای دیگر از کارهایی که از منابع موجود در وب معنایی برای فراهم کردن دانش لازم استفاده کرده است، می‌توان به [158158] اشاره کرد که توصیف زبان طبیعی سیستم را به عنوان ورودی گرفته و نمودار کلاس *UML* مربوط به آن را به عنوان خروجی تولید می‌کند. روش کار بدین ترتیب است که با اجرای پردازش‌های زبان طبیعی بر روی توصیف متنی ورودی، یک سری نقش‌های معنایی استخراج می‌شود. سپس به کمک هستان‌شناسی *RCyc* یا همان *Research Cyc*، که در حکم منبع دانش پس‌زمینه یا دانش عمومی استفاده می‌شود این نقش‌ها بررسی شده و مشخص می‌شود که به ازای هر کدام از این نقش‌ها چه موجودیتی باید ایجاد شود. مثلا به ازای برخی نقش‌ها، یک کلاس و به ازای برخی دیگر یک خصیصه یا متد ایجاد می‌شود. استفاده از هستان‌شناسی، به رفع ابهام و در نتیجه افزایش دقت تصمیم‌گیری و بهبود کیفیت مدل‌های خروجی منجر می‌شود.

در [208208] نیز راهکاری برای تشخیص ابهام یا خطا در مستندات نیازمندی‌های نرم‌افزار ارائه شده است که از هستان‌شناسی برای رفع ابهام‌های ناشی از بکارگیری زبان طبیعی استفاده می‌کند. در [209209] نیز تشخیص ناسازگاری در توصیف نیازمندی‌های سیستم با بکارگیری هستان‌شناسی‌های موجود در وب معنایی انجام شده است.

کارهای محدودی نیز در زمینه استفاده از وب معنایی در مهندسی نرم‌افزار با هدف ایجاد حاشیه-نویسی و برچسب‌زنی انجام شده که به عنوان مثال می‌توان به [191191] اشاره کرده که در آن توصیف معنایی کد منبع با منابع مناسب در پایگاه‌هایی نظیر *DBpedia*، *Freebase* و *OpenCyc* از ابر *LOD* حاشیه‌نویسی شده است. در [187187] نیز چنین دیدگاهی درباره ایجاد پیوند بین منابع سیستم با منابع موجود در *DBpedia* مطرح شده است. همچنین در [210210] ابزار *SmartLink* معرفی شده که یک ویرایشگر و محیط جستجوی تحت وب است که امکان ایجاد توصیف‌های سبک برای وب سرویس‌ها به کمک منابع داده پیوندی را فراهم می‌کند. بدین ترتیب می‌توان خصیصه‌های عملیاتی و غیرعملیاتی سرویس‌ها را در قالب داده‌های پیوندی توصیف نمود. این ابزار، برای حاشیه‌نویسی سرویس‌ها از منابع وب داده نظیر *DBpedia* و موتور جستجوی *Watson* استفاده می‌کند. این حاشیه‌نویسی‌ها یا برچسب‌ها،

کاربران انسانی یا ماشینی را در کشف سرویس‌های مورد نیازشان کمک می‌کند. به عنوان یک مثال، اگر یک سرویس کارش ارائه متاداده‌های منابع است، می‌تواند با منبع http://dbpedia.org/resource/Metadata که مفهوم متاداده را توصیف می‌کند، برچسب زده شود.

۳-۵-۲ نتیجه‌گیری

مرور کارهای انجام شده در زمینه مهندسی نرم‌افزار مبتنی بر وب معنایی نشان می‌دهد که عمده این کارها مبتنی بر رویکرد اول هستند و با بکارگیری فناوری‌های وب معنایی درصدد بهبود فرآیند توسعه نرم‌افزار می‌باشند. در کارهای مختلف، مصنوعات مختلف نرم‌افزاری مد نظر بوده و اطلاعات آن‌ها با استفاده از هستان‌شناسی‌های مختلف به فضای داده‌های معنایی منتقل شده است تا از مزایایی نظیر بهبود قابلیت جستجو، استنتاج خودکار، قابلیت یکپارچه‌سازی داده‌های مختلف و افزایش انعطاف‌پذیری استفاده شود. بنابراین با توجه به رویکرد اول، نقد خاصی به کارهای موجود وارد نیست چرا که مزایای این رویکرد بارها بطور تجربی به اثبات رسیده است.

اما نقدی که به کارها و محققان حوزه مهندسی نرم‌افزار مبتنی بر وب معنایی وارد است آن است که در حال حاضر کارهای خیلی کمی بر اساس رویکرد دوم انجام شده است. اگر استفاده از فناوری‌های وب معنایی برای ایجاد یک مخزن محلی از مصنوعات نرم‌افزاری می‌تواند در بهبود فعالیت‌های مهندسی نرم‌افزار مفید باشد، آیا استفاده از خود وب معنایی که یک مخزن سراسری با مقیاس بزرگ است نباید به طریق اولی مفید باشد؟ آیا حجم قابل توجه داده‌های معنایی منتشر شده بر روی وب معنایی، مثلا داده-های ابر *LOD*، قابلیت استفاده در فرآیندهای مهندسی نرم‌افزار را ندارند؟ آیا این داده‌ها نمی‌توانند قسمتی از دانشی که در فعالیت‌های مهندسی نرم‌افزار مورد نیاز است را فراهم کنند. اگر پاسخ مثبت است، کدام فعالیت‌ها پتانسیل بیشتری برای استفاده از داده‌های وب معنایی دارند؟ همچنین اگر پاسخ منفی است، پرسش‌های جدیدی مطرح می‌شود: آیا چالش ذاتی خاصی در این زمینه وجود دارد یا

مشکل، صرفا به فقدان داده‌های مورد نیاز برمی‌گردد؟ چه نوع داده‌هایی مورد نیاز است؟ آیا مشکل، نبود هستان‌شناسی‌های لازم است یا کمبود داده‌های نمونه؟

بررسی و تشخیص علت این که چرا از رویکرد دوم خیلی کم استفاده شده است تحقیق جداگانه-ای را طلب می‌کند تا بتوان به پرسش‌هایی نظیر موارد فوق پاسخ داد. به نظر می‌رسد جای چنین تحقیقی در عرصه مهندسی نرم‌افزار مبتنی بر وب معنایی خالی است و در صورت انجام این مهم، نتایج آن می‌تواند برای مهندسان نرم‌افزار و همچنین محققین حوزه وب معنایی بسیار ثمربخش باشد.

تحقیق جاری در صدد ارائه پاسخ دقیق برای پرسش‌های فوق نیست اما با توجه به اینکه روش ارائه شده این رساله، از هر دو رویکرد استفاده می‌کند، چنانچه نتایج ارزیابی نشان دهد که روش ارائه شده از کیفیت خوبی برخوردار است، آنگاه می‌توان از نتایج آن در پاسخ به پرسش‌های فوق بهره جست.

ارتباط تحقیق جاری با وب معنایی بدین ترتیب است که در روش ارائه شده، از هر دو رویکرد مورد اشاره استفاده شده است. یعنی هم از فناوری‌های وب معنایی برای ایجاد یک لایه معنایی روی مدل‌ها استفاده شده است و هم از خود وب معنایی به عنوان یک منبع دانش که می‌تواند در خودکارسازی تولید مدل‌ها استفاده شود بهره‌گیری شده است.

در رابطه با رویکرد دوم، فرضیه تحقیق جاری آن است که داده‌های موجود در وب معنایی (نظیر منابع داده ابر LOD، موتورهای جستجوی معنایی) فضای مفهومی مناسبی را ایجاد می‌کنند که استفاده از آن می‌تواند به تولید خودکار برخی مدل‌ها کمک کند. بطور خاص، فرضیه این تحقیق آن است که اطلاعاتی از نوع آنچه در نمودار کلاس توصیف می‌شود را می‌توان از منابع وب معنایی نظیر هستان-شناسی‌ها و موتورهای جستجوی معنایی بازیابی کرد. از آنجا که نمودارهای کلاس، بخش مهمی از مدل-های طراحی یک برنامه کاربردی تحت وب هستند، چنانچه بتوان اطلاعات آن‌ها را از وب معنایی جمع-آوری کرد، این بدان معناست که فضای مفهومی ارائه شده توسط وب معنایی به عنوان منبعی جدید، جایگزین قسمتی از مدل‌های طراحی شده است.

[۱] Instance Data

درستی فرضیه فوق به نتایج ارزیابی روش ارائه شده بستگی دارد و چنانچه روش ارائه شده در تولید خودکار مدل‌های هدف موفق باشد، یعنی وب معنایی توانسته است نیازمندی‌های لازم برای درستی فرضیه فوق را برآورده کند.

استفاده از رویکرد دوم در این تحقیق با این هدف انجام شده است که بخشی از دانش مورد نیاز برای تولید خودکار نمودارهای فعالیت، بجای آن که توسط کاربر فراهم شود، از منابع وب معنایی کسب شود. در نتیجه، ورودی گرفته شده از کاربر پیچیدگی خیلی کمی دارد. در واقع، کاربر تنها یک نمودار مورد کاربرد به عنوان ورودی ارائه می‌کند و بقیه دانش مورد نیاز، مثلا اینکه مفاهیم موجود در این نمودار مورد کاربرد با یکدیگر چه روابطی دارند و هر یک چه خصیصه‌هایی دارند، از منابع وب معنایی استخراج می‌شود. اگر از رویکرد دوم استفاده نمی‌شد لازم بود همه دانش لازم از کاربر گرفته شود و در نتیجه لازم بود کاربر علاوه بر نمودار مورد کاربرد، نمودارهای دیگری نظیر نمودار کلاس را نیز بطور دستی ایجاد نماید. چنین استفاده‌ای از داده‌های وب معنایی به عنوان یک منبع پشتیبان دانش در فرآیند استفاده مجدد از مدل‌ها، یکی از ایده‌های اصلی این تحقیق و همچنین یکی از جنبه‌های نوآوری آن می‌باشد.

۳- روش پیشنهادی

در این فصل روش پیشنهادی تحقیق جاری توضیح داده می‌شود. هدف این تحقیق، ارائه یک راهکار استفاده مجدد از مدل‌های موجود برای توصیف مدل نیازمندی‌های برنامه‌های کاربردی تحت وب می‌باشد. در این راهکار، از فناوری‌های وب معنایی به عنوان زیرساختی برای بازنمایی اطلاعات مدل‌ها و از وب معنایی به عنوان منبعی که می‌تواند نیازمندی‌های اطلاعاتی را پاسخ دهد، استفاده شده است.

راهکار ارائه شده بر مدل نیازمندی‌ها و بطور خاص مدل نیازمندی‌های عملیاتی تمرکز دارد. یکی از روش‌های رایج برای توصیف این نیازمندی‌ها استفاده از مدل‌سازی مورد کاربرد است که طی آن، نیازمندی‌ها در دو مرحله توصیف می‌شوند. در مرحله نخست، یک توصیف کلی و مستقل از محاسبات[۱] در قالب نمودار مورد کاربرد *UML* ایجاد می‌شود و در مرحله بعد هر یک از مورد کاربردها با استفاده از تکنیک دیگری، مثلا نمودار فعالیت *UML* یا قالب‌های متنی ساخت‌یافته، بطور جزئی‌تر توصیف می‌شوند. در تحقیق جاری، بمنظور انطباق با استانداردها و پیروی از رویه‌های مرسوم، توصیف جزئی نیازمندی‌های عملیاتی در قالب نمودار فعالیت *UML* در نظر گرفته شده است.

نمای کلی روش پیشنهادی در شکل ۳-۱شکل ۳-۱ نشان داده شده است. همانطور که در این شکل دیده می‌شود، این روش شامل دو فرآیند اصلی است:

۱. فرآیند آماده‌سازی مخزن معنایی مدل‌ها
۲. فرآیند استفاده مجدد

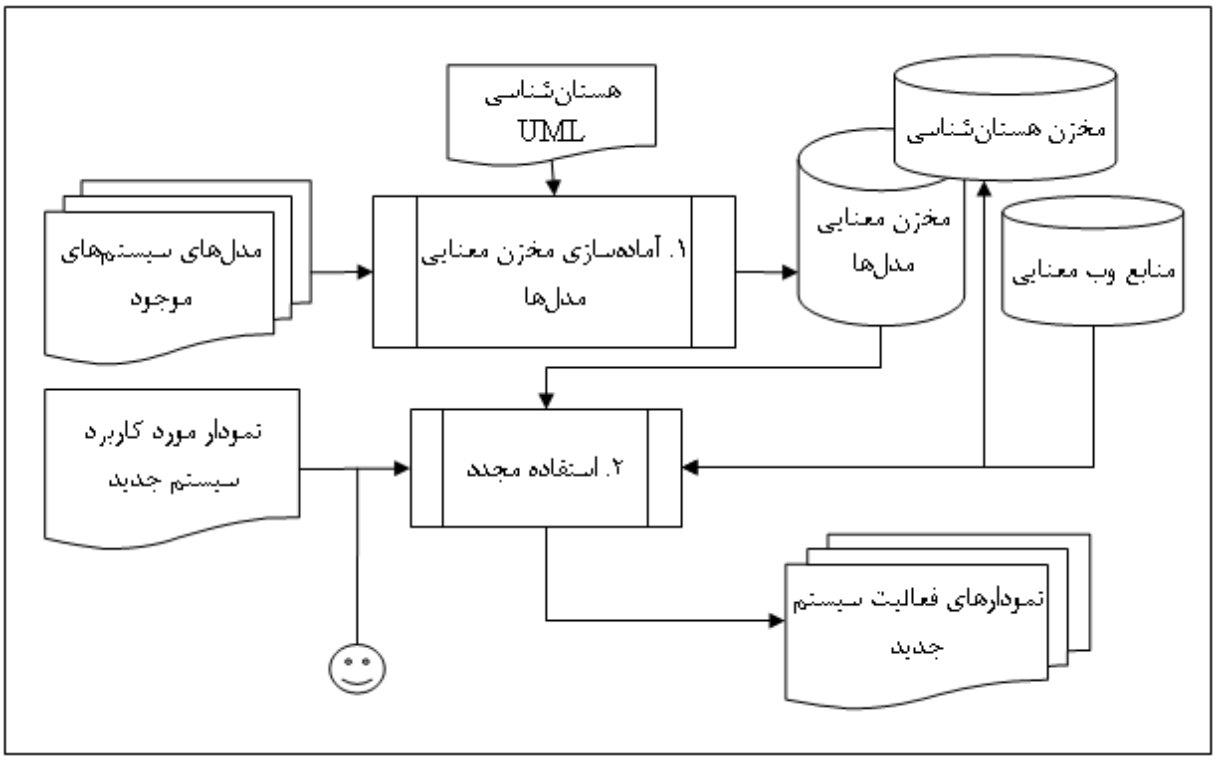
بطور خلاصه، در فرآیند آماده‌سازی مخزن معنایی مدل‌ها، مدل‌های مربوط به برنامه‌های کاربردی موجود مورد پردازش قرار گرفته، اطلاعات آن‌ها استخراج و با استفاده از یک هستان‌شناسی در قالب داده‌های وب معنایی بازنمایی می‌شود. این انتقال به فضای داده‌های معنایی، مزایایی دارد که مهم‌ترین آن‌ها سهولت دستیابی به جزئیات اطلاعات، تحلیل و استنتاج بر روی آن‌ها می‌باشد. علاوه بر ایجاد بازنمایی معنایی، فرآیند آماده‌سازی مخزن معنایی مدل‌ها شامل یک فعالیت دیگر تحت عنوان حاشیه-

[۱] Computationally Independent

نویسی مدل‌ها می‌باشد. هدف این فعالیت آن است که با انجام پردازش‌هایی بر روی مدل‌های ذخیره شده در مخزن و با ایجاد حاشیه‌نویسی‌هایی بر روی آن‌ها، زمینه و آمادگی استفاده مجدد از آن‌ها را فراهم کند.

در فرآیند استفاده مجدد، کاربر توصیف کلی نیازمندی‌های عملیاتی مربوط به برنامه کاربردی جدید را در قالب نمودار مورد کاربرد به عنوان ورودی به روش پیشنهادی می‌دهد. سپس، فرآیند استفاده مجدد با بهره‌گیری از مخزن معنایی مدل‌ها، مخزن هستان‌شناسی‌ها و همچنین منابع وب معنایی نظیر داده‌های پیوندی، بطور نیمه‌خودکار، توصیف جزئی نیازمندی‌های عملیاتی را در قالب نمودارهای فعالیت تولید می‌کند.

در ادامه، ابتدا برخی مفاهیم پایه که در روش پیشنهادی مورد استفاده قرار گرفته است مطرح شده و سپس جزئیات مربوط به هر یک از این دو فرآیند توضیح داده می‌شود.



شکل ۳-۱ نمای کلی روش پیشنهادی

۳-۱ مقدمه

نیازمندی‌های عملیاتی سیستم، در قالب نمودار مورد کاربرد و نمودارهای فعالیت، به ترتیب برای توصیف کلی و توصیف جزئی نیازمندی‌ها، بیان می‌شود. فرض بر آن است که به ازای هر مورد کاربرد تعریف شده در نمودار مورد کاربرد، یک نمودار فعالیت برای توصیف دقیق آن مورد کاربرد وجود دارد.

هر مورد کاربرد، رفتار مشخصی از سیستم را معرفی می‌کند، که این رفتار می‌تواند شامل ایجاد و استفاده از اشیایی از انواع مختلف باشد. چنانچه نمودار مورد کاربرد با دقت ایجاد و مورد کاربردها بخوبی نام‌گذاری شود، نام هر مورد کاربرد باید به رفتار و همچنین مهمترین انواع اشیایی که در آن رفتار نقش دارند اشاره داشته باشد. به عنوان مثال، یک مورد کاربرد با نام *'Edit Account'* بخوبی نشان می‌دهد که این مورد کاربرد، رفتار ویرایش (*Edit*) را ارائه می‌کند و ضمناً این رفتار مستلزم ایجاد یا دستکاری اشیایی از نوع حساب کاربری (*Account*) می‌باشد. برای مورد کاربرد UC، مجموعه مفاهیم برابر مجموعه‌ای شامل مفاهیم یا انواع موجودیت‌هایی که در نام مورد کاربرد UC ذکر شده‌اند تعریف می‌شود.

در این رساله، هر مورد کاربرد UC_i در قالب یک چندگانه به شکل زیر در نظر گرفته می‌شود:

$UC_i = \{name_i, S_i, B_i, A_i, E_i, I_i, G_i, C_i\}$

که $name_i$ نام مورد کاربرد UC_i، S_i نام سیستم مربوطه که UC_i به آن تعلق دارد و B_i مجموعه رفتارهای اعلان شده UC_i می‌باشد. همچنین بقیه عناصر این چندگانه به شکل زیر تعریف می‌شوند.

$A_i = \{A \mid$ there is a direct/indirect Association relationship between actor A and $UC_i\}$
$E_i = \{UC_j \mid$ there is a direct/indirect Extend relationship from UC_i to use case $UC_j\}$
$I_i = \{UC_j \mid$ there is a direct/indirect Include relationship from UC_i to use case $UC_j\}$
$G_i = \{UC_j \mid$ there is a direct/indirect Generalization relationship from UC_i to use case $UC_j\}$
$C_i = \{C \mid$ behavior of UC_i involves manipulating concept $C\}$

با داشتن مورد کاربرد UC_i و همچنین نمودار مورد کاربرد مربوط به آن، یعنی UCD_i، تعیین $name_i$، S_i، A_i، E_i، I_i و G_i ساده است چرا که این عناصر بطور صریح در نمودار مورد کاربرد بیان شده‌اند، اما تشخیص B_i و C_i امری بدیهی نیست. ایده این رساله آن است که اگر مورد کاربرد UC_i به شکل مناسبی نام‌گذاری شده باشد، باید در نام آن به رفتاری که این مورد کاربرد ارائه می‌کند و همچنین به مهمترین مفاهیمی که در این رفتار استفاده می‌شوند اشاره‌ای شده باشد. به بیان دیگر باید بتوان B_i و C_i

را از $name_i$، یعنی نام مورد کاربرد استخراج کرد. الگوریتمی که در ادامه توضیح داده می‌شود به همین منظور طراحی شده است.

۱-۱-۳ الگوریتم تشخیص مفاهیم و رفتار مورد کاربرد

الگوریتم پیشنهادی برای تشخیص C_i و B_i، یعنی مجموعه مفاهیم و رفتار مورد کاربرد UC_i، از روی $name_i$، یعنی نام مورد کاربرد UC_i، به شرح زیر است:

ابتدا عمل جداسازی کلمات[۱]را روی نام مورد کاربرد انجام داده و سپس با درج یک نویسه فضای خالی بین هر دو کلمه متوالی، عبارت *exp* را تولید می‌شود. به عنوان مثال اگر نام موردکاربرد برابر *'searchMovieInIMDB'* باشد، عبارت حاصل برابر *exp='search Movie In IMDB'* خواهد بود. در ادامه، عبارت *exp*، به حروف کوچک تبدیل شده و یک رشته خاص تحت عنوان چاشنی به ابتدای آن افزوده می‌شود. عبارت حاصل، به یک ابزار برچسب‌زننده نقش کلمات[۲]داده می‌شود و کلماتی که برچسب اسمی به آن‌ها تعلق گرفته است به عنوان عناصر مجموعه مفاهیم مورد کاربرد، یعنی C_i، و کلماتی که برچسب فعلی به آن‌ها انتساب داده شده به عنوان عناصر مجموعه رفتار مورد کاربرد، یعنی B_i، انتخاب می‌شوند.

از آنجا که نام مورد کاربرد معمولا یک رشته کوتاه است و نه یک جمله کامل، عمل برچسب زنی نقش کلمات بر روی آن با اشکال مواجه شده و دقت خوبی ندارد. به عنوان نمونه برای مورد کاربردی با نام *'search contacts'*، اگر ابزار برچسب زننده نقش کلمات استنفورد[۳]بر روی نام این مورد کاربرد اجرا شود، به هر دو کلمه *'search'* و *'contacts'* برچسب اسمی انتساب داده می‌شود و این با آنچه مورد انتظار است سازگاری ندارد، چرا که برای این مورد کاربرد، کلمه *'contacts'* تنها عنصر سازنده مجموعه مفاهیم مورد کاربرد است و کلمه *'search'* در واقع بیانگر رفتار این مورد کاربرد است.

---

[۱] Tokenization
[۲] Part-Of-Speech (POS) Tagger
[۳] http://nlp.stanford.edu/software/tagger.shtml

هدف از بکارگیری رشته چاشنی آن است که عبارت حاصل از نام مورد کاربرد، تا حد ممکن به یک جمله کامل شبیه شود تا ابزار برچسب زننده با دقت بیشتری کار خود را انجام دهد. به عنوان مثال درباره مورد کاربرد *'search contacts'* چنانچه از رشته *'I want to'* به عنوان چاشنی استفاده شود عبارت حاصل برابر *'I want to search contacts'* خواهد بود که ابزار برچسب زننده نقش کلمات استنفورد، این بار بدرستی، تنها به کلمه *'contacts'* برچسب اسمی انتساب می‌دهد.

الگوریتم تشخیص مفاهیم و رفتار مورد کاربرد، در چند نقطه از روش پیشنهادی مورد استفاده قرار می‌گیرد، بهمین دلیل، پیش از معرفی جزئیات روش پیشنهادی، توضیح داده شد. برای استفاده از این الگوریتم، لازم است یک مقدار مناسب به عنوان رشته چاشنی انتخاب شود. خوشبختانه چنان که در بخش ۴-۶ نشان داده خواهد شد، نتایج ارزیابی این الگوریتم نشان می‌دهد که انتخاب مقدار مناسب برای چاشنی، ساده و دقت و کارایی این الگوریتم بسیار خوب می‌باشد.

۳-۲ فرآیند آماده‌سازی مخزن معنایی مدل‌ها

مخزن معنایی مدل‌ها[۱]مخزنی است که اطلاعات مدل‌های مربوط به برنامه‌های کاربردی موجود را در قالب داده‌های معنایی ذخیره می‌کند. منظور از داده‌های معنایی، داده‌هایی است که بر اساس مدل داده وب معنایی بازنمایی شده است. در این مدل داده، داده‌ها در قالب سه‌گانه‌های *RDF* بیان می‌شوند. همچنین شمای داده‌ها با استفاده از یک یا تعدادی هستان‌شناسی توصیف می‌شود. مدل داده وب معنایی در مقایسه با مدل داده‌های متداول‌تر نظیر مدل داده رابطه‌ای مزایایی دارد که برخی از آن‌ها در بخش ۲-۵ مورد اشاره قرار گرفت. با توجه به همین مزایا، در طراحی روش پیشنهادی نیز از این مدل داده استفاده شده است.

نمای کلی فرآیند آماده‌سازی مخزن معنایی مدل‌ها در شکل ۳-۲~~شکل ۳-۲~~ نشان داده شده است. همانطور که در این شکل دیده می‌شود این فرآیند شامل سه مرحله اصلی است:

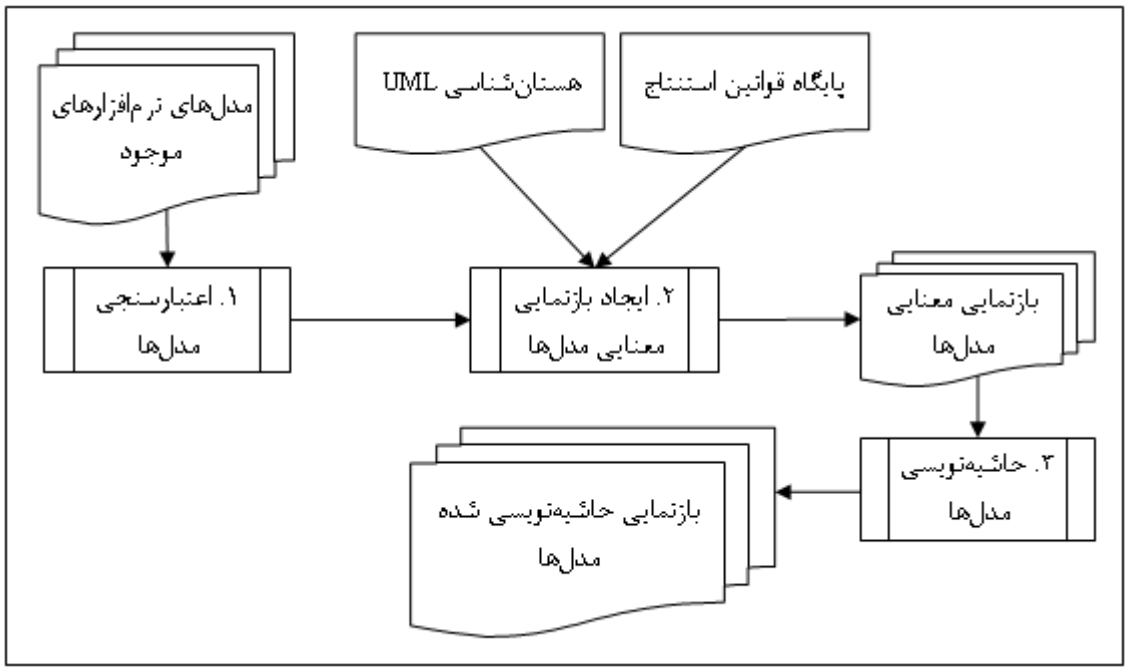
۱. اعتبارسنجی مدل‌ها

[۱] بمنظور سادگی بحث، در ادامه از "مخزن مدل‌ها" یا "مخزن" برای ارجاع به "مخزن معنایی مدل‌ها" استفاده می‌شود.

۲. ایجاد بازنمایی معنایی مدل‌ها

۳. حاشیه‌نویسی مدل‌ها

جزئیات این سه مرحله در بخش‌های بعد توضیح داده می‌شود.



شکل ۳-۲ نمای کلی فرآیند آماده‌سازی مخزن معنایی مدل‌ها

۳-۲-۱ اعتبارسنجی مدل‌ها

مدل‌های ذخیره شده در مخزن با هدف استفاده مجدد در آینده ذخیره می‌شوند. بنابراین انتظار می‌رود این مدل‌ها از کیفیت خوبی برخوردار باشند و بتوان آن‌ها را الگوهای مناسبی برای ایجاد مدل‌های جدید به حساب آورد. از این رو لازم است پیش از ذخیره یک مدل در مخزن، بررسی شود که آیا آن مدل شرایط لازم برای آنکه فرآیند ارائه شده بتواند بخوبی از آن استفاده مجدد کند را دارد یا خیر. به بیان دیگر، فرآیندی که تحقیق جاری برای استفاده مجدد ارائه کرده است، دارای الگوریتم‌هایی است و این الگوریتم‌ها نقاط ضعف و قوت و در نتیجه پیش‌نیازهایی دارند. هرچه مدل‌های موجود در مخزن، این پیش‌نیازها را برآورده کنند، فرآیند استفاده مجدد ارائه شده موفق‌تر خواهد بود.

در مرحله اعتبارسنجی مدل‌ها، مدل‌های ورودی که قرار است در مخزن ذخیره شوند، مورد بررسی قرار گرفته و اشکال‌های احتمالی آن‌ها به مدیر مخزن، یعنی کاربری که مدیریت مخزن معنایی مدل‌ها را بر عهده دارد، گزارش می‌شود. مدیر مخزن می‌تواند نسبت به رفع یا نادیده گرفتن هر مورد گزارش شده تصمیم گرفته و سپس اعتبار مدل مورد نظر را تأیید نماید تا آن مدل به مخزن افزوده شود.

بررسی‌هایی که برای اعتبارسنجی مدل‌ها انجام می‌شود شامل دو مرحله است.

در مرحله نخست، هر یک از مورد کاربردهای موجود در مدل مورد نظر، بررسی می‌شود تا مشخص شود آیا رفتار مشخصی را صریحا اعلان کرده است یا خیر. این کار با اجرای الگوریتمی که در بخش ۳-۱-۱ برای تشخیص رفتار و مفاهیم یک مورد کاربرد معرفی شده است انجام می‌شود. رفتار یک مورد کاربرد، ویژگی مهمی است که در محاسبه شباهت آن مورد کاربرد با دیگر موارد کاربرد نقش تأثیرگذاری دارد. اگر یک مورد کاربرد، فاقد رفتار مشخصی باشد، بدان معناست که شباهت آن با دیگر موارد کاربرد با دقت خیلی کمی محاسبه می‌شود و درنتیجه فرآیند استفاده مجدد نمی‌تواند بخوبی از پتانسیل آن مدل استفاده کند.. بهمین دلیل در اعتبارسنجی مدل‌ها، هر یک از مورد کاربردهایی که فاقد یک رفتار صریح باشند به مدیر مخزن گزارش می‌شود تا در صورت تمایل، با ویرایش نام آن‌ها مشکل را حل کند.

در مرحله دوم اعتبارسنجی مدل‌ها، نام هر یک از مورد کاربردها و همچنین کلیه برچسب‌های موجود در نمودارهای فعالیتی که در مدل مورد نظر وجود دارند، استخراج شده و عمل جداسازی کلمات[۱] بر روی آن انجام می‌شود. سپس

- هر کلمه از نظر املایی بررسی می‌شود و چنانچه دارای غلط املایی باشد، بهمراه لیست کلمات پیشنهادی (برای جایگزین کردن) به مدیر مخزن گزارش می‌شود.
- هر کلمه مورد ریشه‌یابی قرار می‌گیرد و چنانچه ریشه‌یابی آن ناموفق باشد به مدیر مخزن گزارش می‌شود.

[۱] Tokenization

علت بررسی‌های فوق آن است که اگر برچسب‌ها و نام‌های موجود در مدل شامل غلط‌های املایی و نام‌های اختصاری نامأنوس یا کلماتی که ریشه‌یابی آن‌ها با مشکل مواجه می‌شود باشد، علاوه بر اینکه دقت معیار ارائه شده برای محاسبه شباهت دو مورد کاربرد کاهش می‌یابد، کارایی الگوریتم ارائه شده برای حاشیه‌نویسی نمودارهای فعالیت و به تبع آن کیفیت الگوریتم تطبیق نیز افت می‌کند.

یکی از ویژگی‌هایی که در طراحی رویه اعتبارسنجی مدل‌ها در نظر گرفته شده است آن است که این رویه، سخت‌گیر و زمان‌بر نباشد و تعامل با آن برای مدیر مخزن مشکل نباشد. در واقع، مدیر این امکان را دارد که هر یک از موارد گزارش شده را اصلاح کرده یا از آن چشم‌پوشی کند.

۲-۲-۱۳ایجاد بازنمایی معنایی مدل‌ها

پس از آنکه اعتبارسنجی مدل ورودی، با تأیید مدیر مخزن به اتمام رسید، نوبت به ایجاد بازنمایی معنایی آن مدل می‌رسد. روش پیشنهادی، برای ایجاد و استفاده از بازنمایی معنایی مدل‌ها از سه جزء اصلی استفاده می‌کند:

۱. یک هستان‌شناسی که مفاهیم و روابط موجود در مدل‌های *UML* را به صورت رسمی توصیف می‌کند.
۲. یک مترجم[۱] که فایل‌های حاوی مدل‌های *UML* را پردازش، اطلاعات آنها را استخراج و این اطلاعات را بر اساس دامنه لغاتی که توسط هستان‌شناسی فوق فراهم شده است، در قالب داده-های معنایی منتشر می‌کند.
۳. یک مخزن معنایی که قابلیت ذخیره داده‌های معنایی و اجرای پرس و جو بر روی آنها را فراهم می‌کند. پس از آنکه مترجم، بازنمایی معنایی مدل‌ها را ایجاد کرد، آنها را در مخزن معنایی ذخیره می‌کند.

در ادامه، هر یک از این سه عنصر با جزئیات بیشتری معرفی می‌شوند.

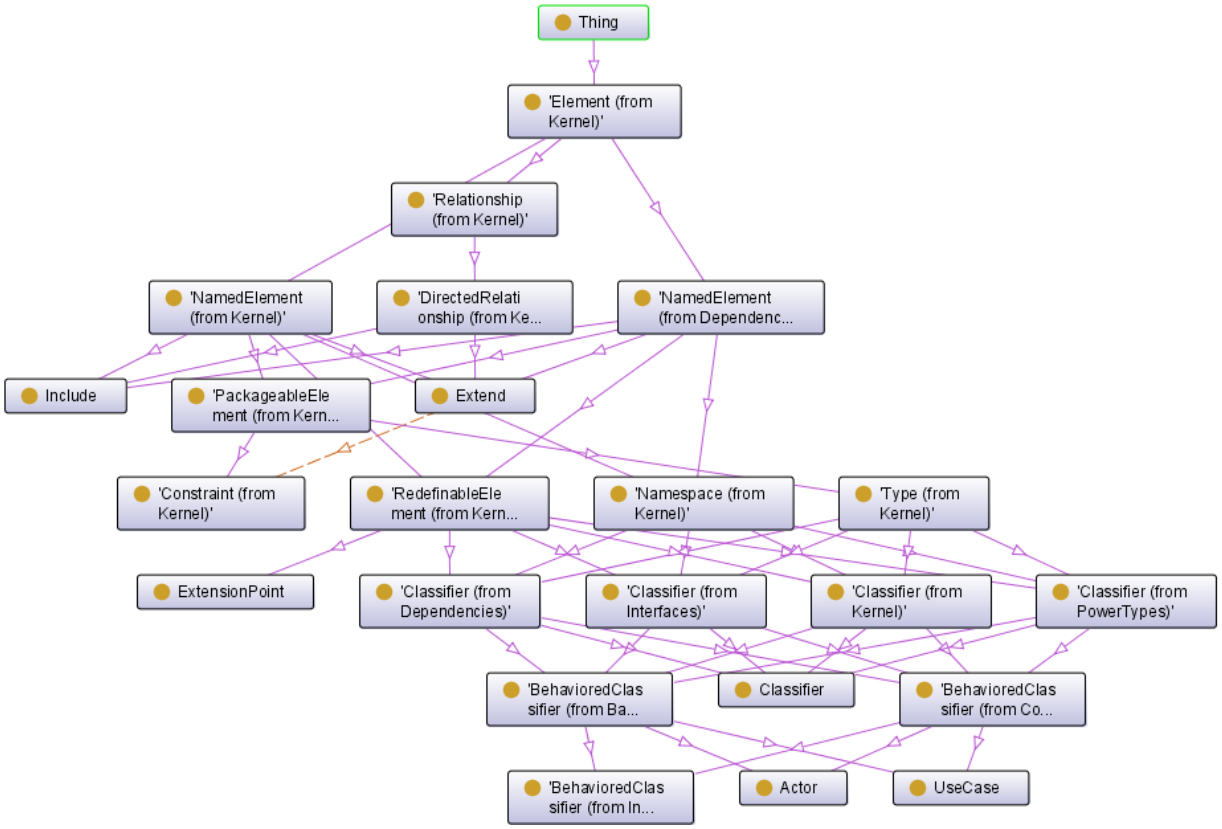
[۱] Parser

۱-۲-۲-۳ هستان‌شناسی UML

زبان *UML* زبانی غنی است که ساختارها و امکانات متعددی دارد. ایجاد یک هستان‌شناسی که فرامدل[۱] زبان *UML* را بطور کامل پوشش دهد کاری پیچیده و زمان‌بر است. به همین علت هستان-شناسی‌ای که برای پیاده‌سازی روش پیشنهادی ایجاد شده است، تنها بخشی از مفاهیم و امکانات زبان *UML* را پوشش می‌دهد که در روش پیشنهادی از آنها استفاده شده است. به عنوان مثال، مفاهیم مورد کاربرد، نمودار مورد کاربرد و نمودار فعالیت در این هستان‌شناسی توصیف شده‌اند.

برای ایجاد هستان‌شناسی مورد نظر، ابتدا بسته‌های فرامدل *UML* [211~~211~~] مورد بررسی قرار گرفته تا مشخص شود کدام عناصر از کدام بسته‌ها بطور مستقیم یا غیرمستقیم در روش پیشنهادی مورد استفاده قرار می‌گیرند. سپس برای هر یک از این بسته‌ها یک هستان‌شناسی به زبان OWL ایجاد شده است که عناصر مورد نظر را توصیف می‌کند. به عنوان نمونه، شکل ۳-۳~~شکل ۳-۳~~ هستان‌شناسی ایجاد شده برای بسته *UseCases* (از سطح ۱ فرامدل) را نشان می‌دهد. این هستان‌شناسی، مفاهیمی نظیر *Actor* و *UseCase* و همچنین روابطی نظیر *ownedUseCase* و *includingCase* را تعریف می‌کند. همچنین این هستان‌شناسی، هستان‌شناسی‌های مربوط به بسته‌های *Kernel* و *Communications* را نیز درون‌بُرد[۲] می‌کند.

[۱] Metamodel
[۲] Import



شکل ۳-۳. قسمتی از هستان‌شناسی UML

از آنجا که فرآیند آماده‌سازی مخزن معنایی مدل‌ها شامل الگوریتمی برای حاشیه‌نویسی مدل‌ها می‌باشد و حاشیه‌نویسی‌های تولید شده توسط این الگوریتم نیز باید در قالب معنایی بازنمایی شده و در مخزن ذخیره شود، یک هستان‌شناسی دیگر نیز ایجاد گردید که مفاهیم مربوط به حاشیه‌نویسی، که در توصیف الگوریتم مربوطه معرفی خواهند شد را در بر می‌گیرد.

این هستان‌شناسی‌ها در نرم‌افزار *Protégé* ایجاد شده‌اند که ابزاری شناخته شده در عرصه مهندسی دانش مبتنی بر هستان‌شناسی است. در نهایت، همه این هستان‌شناسی‌ها در قالب یک هستان-شناسی یکپارچه درون‌بُرد شده‌اند.

۳-۲-۲-۲ مترجم

مترجم، مسئول خواندن فایل‌های حاوی مدل‌ها و تولید بازنمایی معنایی این مدل‌ها می‌باشد. بسته به اینکه از چه نرم‌افزاری برای ایجاد مدل‌ها استفاده شده باشد، ساختار فایل این مدل‌ها متفاوت

است. با این حال، نرم‌افزارهای رایج عمدتاً از استاندارد *XMI*[1] پشتیبانی کرده و امکان تبدیل مدل‌ها به قالب *XMI* را فراهم می‌کنند. این استاندارد، نحوه بازنمایی اطلاعات مدل‌های *UML* با استفاده از قواعد نحوی *XML* را بیان می‌کند. در پیاده‌سازی مترجم، از کتابخانه‌های مربوط به پردازش *XML* برای پردازش فایل‌های مدل‌ها در قالب *XMI* استفاده شده است.

مترجم، اطلاعات مختلفی را درباره مدل‌ها استخراج و این اطلاعات را بصورت داده‌های معنایی بازنمایی می‌کند. علاوه بر اطلاعات مربوط به ساختار و اجزای موجود در مدل‌ها، مترجم با استفاده از الگوریتم ارائه شده در بخش ۱-۱-۳، مفاهیم و رفتارهای هر یک از مورد کاربردهای موجود در مدل را تشخیص و اطلاعات مربوطه را در قالب سه‌گانه‌هایی به بازنمایی معنایی مدل می‌افزاید.

پس از آنکه مترجم، بازنمایی معنایی مدل‌ها را ایجاد کرد، با بارگذاری[2]قوانین موجود در پایگاه قوانین، به استنتاج بر روی بازنمایی معنایی مدل‌ها می‌پردازد. در نتیجه ممکن است بواسطه استنتاج، سه‌گانه‌های *RDF* جدیدی نیز تولید شود که این سه‌گانه‌ها نیز به بازنمایی معنایی مدل‌ها افزوده شده و به نوعی آن را غنی می‌کند.

بدین ترتیب، با توجه به اینکه برخی از سه‌گانه‌های تولید شده توسط مترجم، از عمل استنتاج حاصل شده‌اند می‌توان گفت بخشی از قابلیت‌های مترجم، از طریق تعریف قوانین در پایگاه قوانین پیاده-سازی شده است و نه از طریق پیاده‌سازی در داخل کد منبع مترجم. پیاده‌سازی قابلیت‌های مترجم از طریق اعلان قوانین، مزیت مهمی است که بواسطه استفاده از مدل داده معنایی حاصل شده است و از پیچیدگی پیاده‌سازی مترجم کاسته است. این امر، همچنین موجب افزایش انعطاف‌پذیری مترجم شده است، زیرا پس از آنکه مترجم پیاده‌سازی شد، و حتی در زمان اجرای آن، می‌توان با افزودن قوانین جدید به پایگاه قوانین، با هزینه کمی، قابلیت‌های مترجم را افزایش داد.

لازم به ذکر است که پایگاه قوانین می‌تواند در قالب یک فایل متنی ساده که حاوی اعلان قوانین مورد نظر است پیاده‌سازی شود. در ارزیابی‌هایی که در فصل ۴ توضیح داده شده است، بهمین شکل عمل

[1] XML Metadata Interchange
[2] Load

شده است. دو نمونه از قوانین مورد استفاده مترجم در شکل ۳-۴~~شکل ۳-۴~~ نشان داده شده است. بطور خلاصه، قانون اول بیان می‌کند که خصیصه‌های یک ابرکلاس، به زیرکلاس‌های آن به ارث می‌رسد. همچنین قانون دوم بیان می‌کند که اگر یک عامل، از مورد کاربرد خاصی استفاده می‌کند، از همه مورد کاربردهایی که از آن عامل مشتق می‌شوند نیز می‌تواند استفاده کند.

اگر در توسعه مترجم از پایگاه قوانین استفاده نشده بود، آنگاه رویه‌های لازم برای تشخیص چنین مواردی باید در کد مترجم پیاده‌سازی می‌شد. معمولا چنین رویه‌هایی نیازمند الگوریتم‌های بازگشتی هستند که پیاده‌سازی آن‌ها نیاز به دقت زیادی دارد و اشکال‌زدایی آن‌ها نیز مشکل است. در حالی که با استفاده از پایگاه قوانین، افزودن قابلیت‌های فوق، صرفا مستلزم تعریف منطق قوانین مورد نظر است و عمل استنتاج بطور خودکار و توسط موتورهای استنتاج داده‌های معنایی قابل انجام است، در نتیجه پیچیدگی و هزینه پیاده‌سازی قابلیت‌های مورد نظر کاهش چشمگیری می‌یابد.

```
[rule1: (?subclass rdf:type uml2rdf:Class),
  (?generalization rdf:type uml2rdf:Generalization),
  (?generalization uml2rdf:hasSpecific ?subclass),
  (?generalization uml2rdf:hasGeneral ?superclass),
  (?superclass rdf:type uml2rdf:Class),
  (?superclass uml2rdf:containsAttribute ?attribute)
      -> (?subclass uml2rdf:containsAttribute ?attribute)]

[rule2: (?extend rdf:type uml2rdf:Extend),
  (?extend uml2rdf:extendedCase ?uc1),
  (?extend uml2rdf:extension ?uc2),
  (?actor uml2rdf:usesUseCase ?uc1)
      -> (?actor uml2rdf:usesUseCase ?uc2)]
```

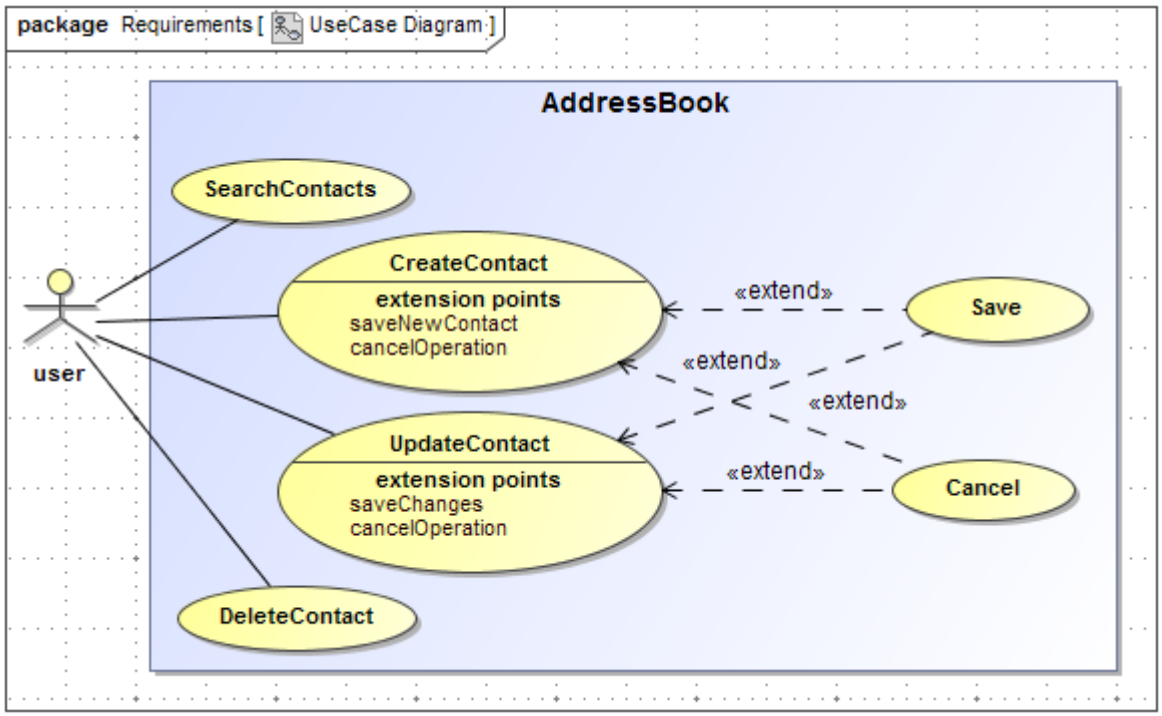
شکل ۳-۴. برخی از قوانین مورد استفاده مترجم

با توجه به اینکه مترجم می‌خواهد اطلاعات مدل‌ها را در قالب داده‌های معنایی منتشر کند، لازم است الگویی برای اختصاص شناسه‌های یکتا از جنس *URI* به موجودیت‌های مورد توصیف در نظر گرفته شود. مترجم توسعه داده شده، از الگوی ساده زیر برای این کار استفاده می‌کند:

{base_URI}{wa_name}_obj_{value}

که *base_URI* یک عبارت تحت اللفظی دلخواه، *wa_name* نام برنامه کاربردی که مدل‌های آن مورد پردازش هستند و *value* مقدار یک شمارنده است. زمانی که پردازش مدل‌های یک برنامه کاربردی شروع می‌شود، مقدار شمارنده برابر صفر است و هر بار که مترجم به موجودیت جدیدی می‌رسد و نیازمند تخصیص شناسه می‌باشد یک واحد به شمارنده اضافه کرده و سپس مقدار آن را به عنوان *value* در الگوی فوق استفاده می‌کند.

به عنوان یک نمونه، قسمت کوچکی از بازنمایی معنایی تولید شده مربوط به نمودار مورد کاربرد شکل ۳-۵شکل ۳-۵، در شکل ۳-۶شکل ۳-۶ نشان داده شده است.



شکل ۳-۵. مثالی از نمودار مورد کاربرد UML

```
uml2rdf:AddressBook_obj_257
        a       uml2rdf:UseCase ;
        uml2rdf:hasSoftwareCase     uml2rdf:AddressBook_obj_1 ;
        uml2rdf:name                "CreateContact" .

uml2rdf:AddressBook_obj_252
        a       uml2rdf:UseCase ;
        uml2rdf:hasSoftwareCase     uml2rdf:AddressBook_obj_1 ;
        uml2rdf:name                "Save" .

uml2rdf:AddressBook_obj_250
```

```
a       uml2rdf:Extend ;
uml2rdf:extendedCase        uml2rdf:AddressBook_obj_257 ;
uml2rdf:extension           uml2rdf:AddressBook_obj_249 ;
```

شکل ۳-۶. قسمت کوچکی از بازنمایی معنایی مربوط شکل ۳-۵~~شکل ۳-۵~~

۳-۲-۲-۳ مخزن معنایی مدل‌ها

مخزن معنایی مدل‌ها، مخزنی است که بازنمایی معنایی مدل‌ها را ذخیره می‌کند. سازوکار پایه برای دسترسی به اطلاعات ذخیره شده در مخزن، از طریق ارسال پرس و جوهای *SPARQL* به مخزن و دریافت پاسخ می‌باشد.

هرچه هستان‌شناسی مورد استفاده برای بازنمایی معنایی مدل‌ها غنی‌تر و کامل‌تر باشد، داده‌های ذخیره شده در مخزن، اطلاعات بیشتری را در بر می‌گیرد و پرس و جوهای متنوع‌تری قابل ایجاد و استفاده خواهد بود. در نتیجه پتانسیل بیشتری برای بازیابی اطلاعات مدل‌ها از طریق پرس و جو بر روی مخزن وجود خواهد داشت.

۳-۲-۳ حاشیه‌نویسی مدل‌ها

پس از آنکه بازنمایی معنایی مدل‌ها ایجاد و در مخزن ذخیره شد، عمل حاشیه‌نویسی بر روی این مدل‌های انجام می‌شود. در ادامه، ابتدا ایده حاشیه‌نویسی مدل‌ها توضیح داده خواهد شد و سپس الگوریتم حاشیه‌نویسی با جزئیات توصیف می‌شود.

حاشیه‌نویسی مدل‌ها مبتنی بر این ایده است که بطور طبیعی، نمودار فعالیتی که مسئول توصیف جزئی یک مورد کاربرد است، باید جزئیات رفتار آن مورد کاربرد را توصیف کند. بنابراین در این نمودار فعالیت باید ارجاع‌هایی به اشیایی که توسط آن رفتار مورد پردازش قرار می‌گیرند وجود داشته باشد. دیدگاه این رساله آن است که اگر این ارجاع‌ها بطور صریح مشخص شوند، امکان استفاده مجدد از نمودارهای فعالیت را فراهم می‌کنند. هدف مرحله حاشیه‌نویسی مدل‌ها، کشف این ارجاع‌ها و سپس بازنمایی صریح آن‌ها می‌باشد، اما اینکه این امر چگونه به استفاده مجدد مدل‌ها می‌انجامد موضوعی است

که در توضیح فرآیند استفاده مجدد و بطور خاص در معرفی الگوریتم تطبیق ارائه شده، درباره آن بحث خواهد شد.

دیدگاه این رساله آن است که اگر مدل‌سازی سیستم مورد نظر بخوبی انجام شده باشد، آنگاه هر یک از مفاهیم مورد کاربرد باید به یکی از موارد زیر ارجاع داشته باشد: ۱) کلاسی که در نمودار کلاس سیستم مورد نظر تعریف شده است، ۲) خصیصه‌ای از یکی از کلاس‌های موجود در نمودار کلاس و ۳) یک کنش‌گر که از آن مورد کاربرد استفاده می‌نماید. در غیر اینصورت، یعنی اگر یکی از مفاهیم مورد کاربرد به مقصدی ارجاع داشته باشد که در بین این سه دسته عناصر نباشد، این امر بمنزله وجود ابهام در تعریف مورد کاربرد می‌باشد و این ابهام می‌تواند در زمان ایجاد توصیف دقیق مورد کاربرد منجر به بروز مشکلاتی شود.

با توجه به اینکه الگوریتم ارائه شده برای حاشیه‌نویسی، نیازمند تشخیص مجموعه مفاهیم یک مورد کاربرد می‌باشد، ابتدا به این موضوع پرداخته و سپس الگوریتم حاشیه‌نویسی توضیح داده می‌شود.

۱-۳-۲-۳ الگوریتم حاشیه‌نویسی مدل‌ها

الگوریتم حاشیه‌نویسی مدل‌ها در قالب شبه کد در شکل ۳-۷ نشان داده شده است. در مرحله حاشیه‌نویسی مدل‌ها، این الگوریتم بر روی کلیه نمودارهای فعالیت ذخیره شده در مخزن اجرا می‌شود.

ورودی این الگوریتم یک نمودار فعالیت *AD* که به برنامه کاربردی *WA* تعلق دارد بهمراه نمودار کلاس مربوطه (*CD*) و نمودار مورد کاربرد مربوطه (*UCD*) می‌باشد. خروجی این الگوریتم نیز لیست حاشیه‌نویسی‌های نمودار فعالیت *AD* است.

Input:
    AD: an activity diagram from web application *WA*
    CD: class diagram associated with *WA*
    UCD: use case diagram associated with *WA*
Output:
    annotations: a list of annotations created for *AD*
Algorithm
1. *classes* = list of the classes declared in *CD*
2. *attributes* = list of the attributes of the classes declared in *CD*

```
3. actors = list of the actors declared in UCD
4. entities = {classes, attributes, actors}
5. labels = list of the labels declared in AD
6. annotations = an empty list
7. foreach label L in labels {
8.       annotations_L = an empty list
9.       words = list of the words of L
10.      for N=1 to MAX_NGRAM_LENGTH {
11.              foreach N-gram text from words list {
12.                      foreach entity ent in entities {
13.                        if(ent.name == text) {
14.                          annot_L = new Annotation
15.                          annot_L.source = identifier of the element to which label L
belongs
16.                          annot_L.target = ent.identifier
17.                          annot_L.text = text
18.                          annot_L.type = determine annotation type
19.                          annotations_L.add(annot_L)
20.                        }
21.                      }
22.              }
23.      }
24.      if(annotations_L.size > 1) {
25.              annotations_L = annotation_resolution(annotations_L)
26.      }
27.      annotations.add(annotations_L)
28. }
29. return annotations
```

شکل ۳-۷. شبه کد الگوریتم حاشیه‌نویسی نمودار فعالیت

همانطور که در شکل ۳-۷ نشان داده شده است، این الگوریتم ابتدا یک لیست با نام *entities* ایجاد می‌کند که شامل کلاس‌های اعلان شده در نمودار کلاس *CD*، خصیصه‌های تعریف شده برای این کلاس‌ها و عامل‌های موجود در نمودار مورد کاربرد *UCD* می‌باشد (خط‌های ۱ تا ۴). سپس لیستی با نام *labels* ایجاد می‌شود که شامل همه برچسب‌های موجود در نمودار فعالیت *AD* می‌باشد (خط ۵). همچنین لیست *annotations* که در ابتدا تهی است و در پایان اجرای الگوریتم، شامل لیست حاشیه-نویسی‌های ایجاد شده برای نمودار فعالیت *AD* است ایجاد می‌شود (خط ۶). بعد از ایجاد این لیست‌ها، برای هر برچسب L از لیست *labels*، یک لیست از حاشیه‌نویسی‌ها با نام $annotations_L$ ایجاد می‌شود که شامل حاشیه‌نویسی‌های ایجاد شده برای برچسب L است (خط‌های ۷ تا ۲۳). برای این منظور، ابتدا

برچسب L به لیست کلمات سازنده‌اش تجزیه شده و سپس *N-gram* های مختلف حاصل از این لیست کلمات، در لیست *entities* جستجو می‌شوند. به ازای هر *N-gram* که با یکی از عناصر این لیست مطابقت داشته باشد یک حاشیه‌نویسی ایجاد و به لیست $annotations_L$ افزوده می‌شود (خط‌های ۱۴ تا ۱۹). هر حاشیه‌نویسی شامل چهار مشخصه است که عبارتند از:

۴. منبع[۱]: برابر شناسه عنصری از نمودار فعالیت *AD* است که برچسب L به آن تعلق دارد.

۵. مقصد[۲]: برابر شناسه عنصری که نامش با N-gram مطابقت داشته است.

۶. متن[۳]: برابر رشته N-gram مورد مطابقت

۷. نوع: یک مقدار عددی که نوع حاشیه‌نویسی را مشخص می‌کند.

پس از آنکه همه حاشیه‌نویسی‌های ممکن برای برچسب L ایجاد شد، چنانچه بیش از یک حاشیه‌نویسی در لیست $annotaions_L$ وجود داشته باشد یک روال حل تعارض اجرا می‌شود تا مشخص شود کدام حاشیه‌نویسی نسبت به بقیه از اولویت بیشتری برخوردار است و کدام حاشیه‌نویسی‌ها قابل حذف هستند (خط‌های ۲۴ تا ۲۶). در نهایت، همه عناصر باقیمانده در $annotations_L$ به لیست *annotations* افزوده می‌شود و در پایان الگوریتم، این لیست به عنوان خروجی الگوریتم بازگردانده می‌شود.

۲-۳-۲-۳ انواع حاشیه‌نویسی‌ها

الگوریتم ارائه شده برای حاشیه‌نویسی مدل‌ها، قادر به ایجاد انواع مختلفی از حاشیه‌نویسی‌ها می‌باشد که این انواع را به دو گروه حاشیه‌نویسی‌های متصل و حاشیه‌نویسی‌های غیرمتصل می‌نامیم. یک حاشیه‌نویسی متصل را می‌توان به عنوان یک پیوند[۴] معنایی که مسیری صریح از نمودار فعالیت *AD* به

[۱] Source
[۲] Target
[۳] Text
[۴] Link

مورد کاربرد UC برقرار می‌کند به حساب آورد. یک حاشیه‌نویسی غیرمتصل، چنین مسیر صریحی ایجاد نمی‌کند.

در حال حاضر پنج نوع حاشیه‌نویسی متصل توسط الگوریتم حاشیه‌نویسی قابل تولید است که در ادامه توضیح داده شده‌اند. بمنظور درک بهتر حاشیه‌نویسی‌ها، برای هر نوع، مثال ساده‌ای ارائه شده است. در همه این مثال‌ها از یک مورد کاربرد فرضی با نام *'Create Account'* استفاده شده است. در نتیجه مجموعه مفاهیم مورد کاربرد برابر $\{$*'Account'*$\}$ می‌باشد. همچنین فرض شده است که نمودار کلاس برنامه کاربردی مربوطه شامل کلاسی به نام *'Account'* است.

۱. حاشیه‌نویسی نوع ۱: نمودار فعالیت AD شامل برچسب L است که این برچسب، شامل نام کلاس CL (از نمودار کلاس مربوطه) می‌باشد و این کلاس نیز با مفهوم C که در مجموعه مفاهیم UC قرار دارد انطباق دارد. این حاشیه‌نویسی را می‌توان به عنوان مسیری به شکل $[AD \rightarrow L \rightarrow CL \rightarrow C \rightarrow UC]$ به حساب آورد. به عنوان مثال، فرض کنید AD شامل برچسب L=*'new Account'* باشد. از آنجا که L شامل نام کلاس *'Account'* است که با یکی از مفاهیم مورد کاربرد، یعنی C=*'Account'*، انطباق دارد، یک حاشیه‌نویسی نوع ۱ برای برچسب L ایجاد می‌شود که مقصد آن کلاس CL=*'Account'* است.

۲. حاشیه‌نویسی نوع ۲: نمودار فعالیت AD شامل برچسب L می‌باشد که این برچسب شامل نام خصیصه A از کلاس CL می‌باشد و این کلاس با مفهوم C از مجموعه مفاهیم UC تطبیق دارد. این حاشیه‌نویسی را می‌توان به عنوان مسیری به شکل $[AD \rightarrow L \rightarrow A \rightarrow CL \rightarrow C \rightarrow UC]$ به حساب آورد. به عنوان مثال، فرض کنید AD شامل برچسب L=*'enter username'* باشد و کلاس CL=*'Account'* خصیصه‌ای به نام A=*'username'* داشته باشد. در این حالت، یک حاشیه‌نویسی نوع ۲ برای برچسب L ایجاد می‌شود که مقصد آن خصیصه A است.

۳. حاشیه‌نویسی نوع ۳: نمودار فعالیت AD شامل برچسب L است که این برچسب شامل نام کلاس CL است و این کلاس یک ارتباط مستقیم یا غیرمستقیم (مثلا از نوع *Generalization*) با

کلاس *CL'* دارد و کلاس *CL'* با مفهوم *C* از مجموعه مفاهیم *UC* تطبیق دارد. این حاشیه-نویسی را می‌توان در قالب مسیری به شکل *[AD → L → CL → CL' → C → UC]* به حساب آورد. به عنوان مثال، فرض کنید *AD* شامل برچسب *L='notify user'* باشد و کلاس *CL='User'* با کلاس *CL'='Account'* رابطه داشته باشد. در این حالت، یک حاشیه‌نویسی نوع ۳ برای برچسب *L* ایجاد می‌شود که مقصد این حاشیه‌نویسی کلاس *CL='User'* است.

۴. حاشیه‌نویسی نوع ۴: نمودار فعالیت *AD* شامل برچسب *L* می‌باشد که این برچسب شامل نام خصیصه *A* از کلاس *CL* است و این کلاس ارتباطی مستقیم یا غیرمستقیم با کلاس *CL'* دارد که کلاس *CL'* نیز با مفهوم *C* از مجموعه مفاهیم *UC* انطباق دارد. این حاشیه‌نویسی را می‌توان به عنوان یک مسیر به شکل *[AD → L → A → CL → CL' → C → UC]* به حساب آورد. به عنوان مثال، فرض کنید نمودار *AD* شامل برچسب *L='gender == male'* باشد و کلاس CL='User' که خصیصه‌ای به نام *'gender'* دارد با کلاس *CL'='Account'* ارتباط داشته باشد. در این حالت، یک حاشیه‌نویسی نوع ۴ برای برچسب *L* ایجاد می‌شود که مقصد آن خصیصه *A* می‌باشد.

۵. حاشیه‌نویسی نوع ۵: نمودار فعالیت *AD* شامل برچسب *L* است که این برچسب شامل نام عامل *AC* است که این عامل با مورد کاربرد *UC* در ارتباط است. این حاشیه‌نویسی را می‌توان به عنوان یک مسیر به شکل *[AD → L → AC → UC]* به حساب آورد. به عنوان مثال فرض کنید نمودار *AD* شامل برچسب *L='send notification to admin'* باشد و یک عامل با نام *AC='admin'* با مورد کاربرد *UC* در ارتباط باشد. در این حالت یک حاشیه‌نویسی نوع ۵ برای برچسب *L* ایجاد می‌شود که مقصد آن عامل *AC* می‌باشد.

افزون بر پنج نوع حاشیه‌نویسی متصل فوق، دو نوع حاشیه‌نویسی غیرمتصل نیز توسط الگوریتم حاشیه‌نویسی قابل ایجاد است:

۶. حاشیه‌نویسی نوع ۶: نمودار فعالیت *AD* شامل برچسب *L* است که این برچسب شامل نام کلاس *CL* است و این کلاس با هیچیک از عناصر مجموعه مفاهیم *UC* تطبیق ندارد و همچنین، هیچ کلاس *CL'* بگونه‌ای که کلاس *CL* با *CL'* ارتباط داشته باشد و کلاس *CL'* با یکی از عناصر مجموعه مفاهیم *UC* تطبیق داشته باشد وجود ندارد. به بیان ساده، برچسب *L* شامل نام کلاس CL است اما در نام مورد کاربرد *UC* هیچ اشاره‌ای به کلاس *CL* یا کلاس دیگری که با کلاس *CL* در ارتباط باشد وجود ندارد. به عنوان مثال فرض کنید نمودار *AD* شامل برچسب *L='show CAPTCHA'* و کلاسی با نام *CL='CAPTCHA'* وجود دارد. برچسب *L* شامل نام کلاس *CL* است اما این کلاس با هیچیک از مفاهیم مورد کاربرد انطباق ندارد. همچنین این کلاس با هیچیک از کلاس‌هایی که با مفاهیم مورد کاربرد منطبق هستند نیز ارتباطی ندارد. در چنین حالتی یک حاشیه‌نویسی نوع ۶ ایجاد می‌شود که مقصد آن کلاس *CL='CAPTCHA'* است. مشخص است که این نوع حاشیه‌نویسی، هیچ مسیر مشخصی بین *AD* و *UC* ایجاد نمی‌کند.

۷. حاشیه‌نویسی نوع ۷: نمودار فعالیت *AD* شامل برچسب *L* است که این برچسب شامل نام خصیصه *A* از کلاس *CL* است و این کلاس با هیچیک از عناصر مجموعه مفاهیم *UC* تطبیق ندارد و همچنین هیچ کلاس *CL'* بگونه‌ای که کلاس *CL* با *CL'* ارتباط داشته باشد و *CL'* با یکی از عناصر مجموعه مفاهیم *UC* تطبیق داشته باشد وجود ندارد. به بیان ساده، برچسب *L* شامل نام خصیصه *A* از کلاس *CL* است اما در نام مورد کاربرد *UC* هیچ اشاره‌ای به کلاس *CL* یا کلاس دیگری که با کلاس *CL* در ارتباط باشد وجود ندارد. به عنوان مثال فرض کنید نمودار *AD* شامل برچسب *L='complexity_level < 5'* باشد و کلاس *CL='CAPTCHA'* شامل خصیصه *A='complexity_level'* باشد. کلاس *CL* با هیچیک از مفاهیم مورد کاربرد *UC* انطباق ندارد. حال اگر کلاس *CL* با هیچیک از کلاس‌هایی که با مفاهیم *UC* منطبق هستند نیز رابطه نداشته باشد، در این حالت یک حاشیه‌نویسی نوع ۷ ایجاد می‌شود که مقصد آن خصیصه

A='complexity_level' است. مشخص است که این نوع حاشیه‌نویسی نیز هیچ مسیری بین *AD* و *UC* ایجاد نمی‌کند.

با توجه به الگوریتم حاشیه‌نویسی ارائه شده، ممکن است برای برچسب L هیچ حاشیه‌نویسی ایجاد نشود، یا بیش از یک حاشیه‌نویسی ایجاد شود. مثلا برای برچسب *'ShowAddressBook'* ممکن است یک *2-gram* با مقدار *'AddressBook'* با نام کلاس *'AddressBook'* از لیست *classes* تطبیق یابد و همچنین ممکن است یک *1-gram* با مقدار *'Address'* با عنصری به همین نام از لیست *attributes*، مثلا خصیصه *address* از کلاس *User* تطبیق یابد. در چنین مواردی، یعنی در صورت ایجاد بیش از یک حاشیه‌نویسی برای یک برچسب، رویه‌ای تحت عنوان حل تعارض اجرا می‌شود تا مشخص شود کدام حاشیه‌نویسی از اولویت بیشتری برخوردار است و کدامیک قابل حذف است. این رویه در ادامه توضیح داده شده است.

۳-۳-۲-۳ رویه حل تعارض

اگر یک کاربر انسانی بخواهد از بین حاشیه‌نویسی‌های یک برچسب، مناسب‌ترین مورد را انتخاب کند، احتمالا ابتدا مهمترین مفاهیمی که در نمودار فعالیت مربوطه وجود دارند را تشخیص داده و بدین ترتیب زمینه مفهومی نمودار را مشخص می‌کند. سپس حاشیه‌نویسی‌های موجود را بررسی کرده و موردی که به زمینه مورد نظر مرتبط‌تر است را انتخاب می‌کند. الگوریتمی که برای حل تعارض حاشیه-نویسی‌ها ارائه شده است بر این ایده استوار است و سعی می‌کند به روشی مشابه عمل کند.

ورودی الگوریتم، برچسب L بهمراه مجموعه حاشیه‌نویسی‌های تولید شده برای آن می‌باشد و خروجی آن نیز زیرمجموعه‌ای غیرتهی از حاشیه‌نویسی‌های ورودی است. این الگوریتم ابتدا به هر یک از حاشیه‌نویسی‌های ورودی یک سطح اولویت از بین سطوح اولویت "زیاد"، "متوسط" و "کم" اختصاص می‌دهد. این عمل بر اساس یکسری قواعد انتساب اولویت انجام می‌شود. پس از انتساب اولویت، برخی

حاشیه‌نویسی‌ها بر اساس قواعد پالایه[1] حذف می‌شوند. حاشیه‌نویسی‌های باقیمانده، به عنوان خروجی الگوریتم برگردانده می‌شوند.

قواعد انتساب اولویت به حاشیه‌نویسی‌ها بدین شرح می‌باشند:

- اگر مقصد یک حاشیه‌نویسی، خصیصه‌ای از کلاس CL باشد و برچسب L شامل نام کلاس CL باشد، آنگاه این حاشیه‌نویسی دارای سطح اولویت زیاد خواهد بود. این قاعده بر این ایده استوار است که خود برچسب L با ذکر نام کلاس CL بطور صریح ارجاعی به زمینه فراهم کرده است.
- اگر مقصد یک حاشیه‌نویسی، کلاس CL یا یکی از خصیصه‌های آن یا یک عامل AC باشد، آنگاه اگر نام کلاس CL یا عامل AC با یکی از مفاهیم موجود در مورد کاربرد UC مطابقت داشته باشد، سطح اولویت این حاشیه‌نویسی متوسط خواهد بود. این امر مبتنی بر این ایده است که نام مورد کاربرد، با ذکر صریح مفاهیم اصلی که توسط نمودار فعالیت مربوطه پردازش می‌شوند، زمینه نمودار را مشخص کرده است.
- اگر حاشیه‌نویسی با هیچ یک از دو قاعده فوق مطابقت نداشت، سطح اولویت آن برابر اولویت کم در نظر گرفته می‌شود.

پس از انتساب سطح اولویت به همه حاشیه‌نویسی‌های برچسب L، هر یک از این حاشیه‌نویسی‌ها بررسی می‌شود و هر حاشیه‌نویسی $ANNOT_i$، که در حداقل یکی از شرایط زیر صدق کند، حذف می‌شود:

- حاشیه‌نویسی دیگری به نام $ANNOT_j$ وجود دارد بطوری که $Text_j=Text_i$ و $PL_j>PL_i$، که PL_i و PL_j به ترتیب، سطح اولویت حاشیه‌نویسی $ANNOT_i$ و $ANNOT_j$ را مشخص می‌کند.
- حاشیه‌نویسی دیگری به نام $ANNOT_j$ وجود دارد که $Text_j$ شامل $Text_i$ می‌شود.

[1] Filtering Rules

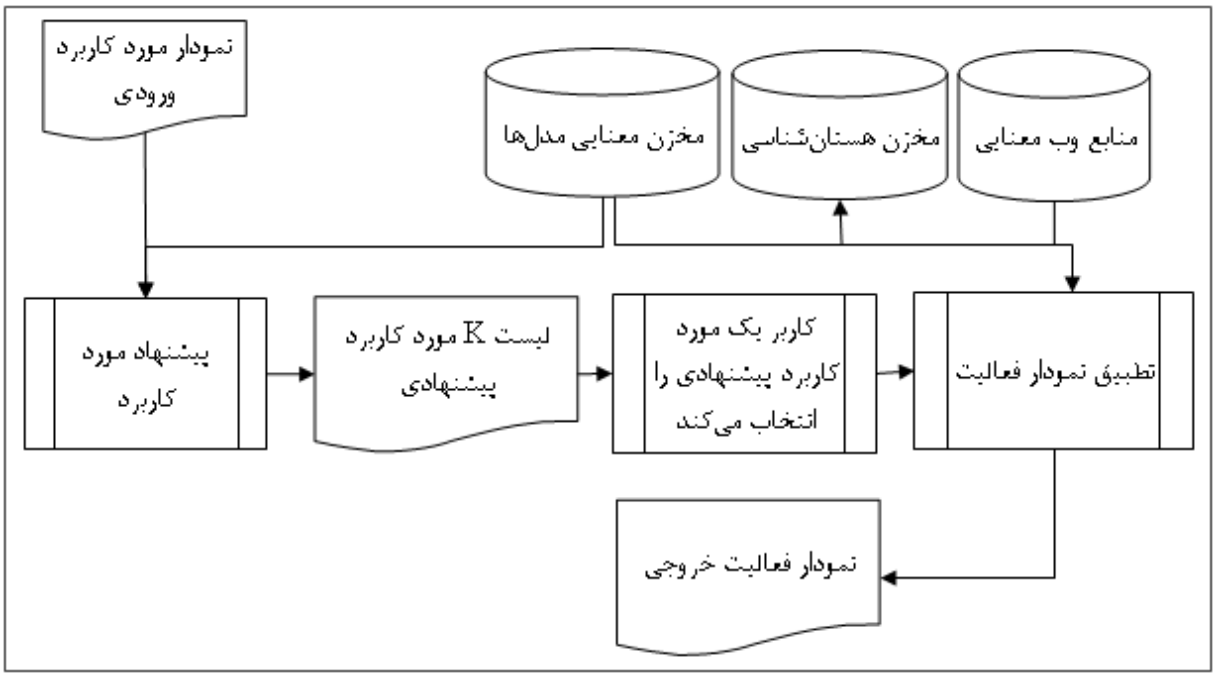
- حاشیه‌نویسی دیگری به نام $ANNOT_j$ وجود دارد بطوری که $Text_j=Text_i$ و مقصد $ANNOT_j$ یک کلاس و مقصد $ANNOT_i$ یک عامل است.

بمنظور درک ساده‌تر الگوریتم حل تعارض و قواعد مربوط به آن، مثال‌هایی از عملکرد آن در بخش ۴-۵ و در طی ارزیابی روش پیشنهادی ارائه خواهد شد.

نکته‌ای که درباره الگوریتم حل تعارض لازم است به آن توجه شود آن است که حتی بعد از اجرای این الگوریتم، ممکن است هنوز بیش از یک حاشیه‌نویسی برای برچسب L باقی مانده باشد. همانطور که در بخش ۲-۳-۳ توضیح داده خواهد شد، حاشیه‌نویسی‌ها نقش مهمی در الگوریتم تطبیق و فرآیند استفاده مجدد دارند. بنابراین وجود چند حاشیه‌نویسی برای برچسب L به معنای شانس بیشتر برای تطبیق این برچسب می‌باشد، چرا که احتمال اینکه الگوریتم تطبیق بتواند حداقل یکی از حاشیه-نویسی‌های برچسب L را تطبیق بدهد افزایش می‌یابد. از طرف دیگر، وجود چند حاشیه‌نویسی در زمان اجرای الگوریتم تطبیق می‌تواند به تولید نتایج متفاوت منجر شده و قطعیت عمل تطبیق را کاهش دهد. در نتیجه، بهتر است حداکثر تعداد حاشیه‌نویسی‌های یک برچسب عدد کوچکی، مانند ۲ یا ۳، باشد. خوشبختانه نتایج ارزیابی که در بخش ۴-۴ گزارش شده است، نشان می‌دهد الگوریتم ارائه شده برای حل تعارض این شرط را احراز می‌کند.

۳-۳ فرآیند استفاده مجدد

نمای کلی فرآیند استفاده مجدد در شکل ۳-۸~~شکل ۳-۸~~ نشان داده شده است. ورودی این فرآیند، توصیف کلی نیازمندی‌های عملیاتی یک برنامه کاربردی جدید WA_{new} در قالب نمودار مورد کاربرد UCD_{new} می‌باشد. خروجی این فرآیند نیز، یک نمودار فعالیت AD_{new} به ازای هر مورد کاربرد UC_{new} از نمودار مورد کاربرد UCD_{new} می‌باشد. نمودارهای فعالیت خروجی که به روشی نیمه‌خودکار و با استفاده از مخزن معنایی مدل‌ها تولید شده‌اند، نسخه ابتدایی توصیف جزئی نیازمندی‌های عملیاتی سیستم WA_{new} می‌باشند.



شکل ۳-۸ نمای کلی فرآیند استفاده مجدد

ایده اصلی فرآیند استفاده مجدد آن است که سیستم‌های مختلف تحت وب، دارای قابلیت‌های مشابه و در نتیجه نیازمندی‌های عملیاتی مشابهی هستند. با دریافت توصیف کلی نیازمندی‌های عملیاتی یک برنامه کاربردی جدید، می‌توان از بین برنامه‌های کاربردی موجود، مواردی که نیازمندی‌های عملیاتی‌شان شبیه برنامه جدید است را پیدا کرده و سپس با استفاده از مدل‌های مربوط به آن برنامه‌ها، توصیف جزئی نیازمندی‌های عملیاتی سیستم جدید را بطور خودکار تولید کرد. فرآیند استفاده مجدد شامل دو مرحله است. مرحله نخست، یعنی پیشنهاد مورد کاربرد، برای پیدا کردن شبیه‌ترین برنامه‌های کاربردی موجود در نظر گرفته شده و هدف مرحله دوم، یعنی تطبیق نمودار فعالیت، استفاده از مدل‌های برنامه‌های مشابه برای تولید نسخه اولیه مدل‌های نیازمندی‌های عملیاتی برنامه جدید می‌باشد.

در ادامه، این دو مرحله به تفکیک معرفی می‌شوند.

۳-۳-۱ پیشنهاد مورد کاربرد

الگوریتم پیشنهاد مورد کاربرد، مورد کاربرد UC_i را به همراه نمودار مورد کاربرد مربوطه، یعنی UCD_i، به عنوان ورودی گرفته و سپس با استفاده از معیار شباهت ارائه شده، که در بخش بعد معرفی

می‌شود، شباهت هر یک از مورد کاربردهای موجود در مخزن معنایی مدل‌ها را با مورد کاربرد ورودی محاسبه کرده و در پایان لیست مرتب (مرتب شده نزولی) K مورد کاربردی که بیشرین شباهت به مورد کاربرد ورودی را دارند را بر می‌گرداند. بدین ترتیب، عنصر اصلی در الگوریتم پیشنهاد مورد کاربرد، معیار شباهتی است که برای محاسبه شباهت دو مورد کاربرد ارائه شده است. این معیار در بخش بعد توضیح داده می‌شود.

۱-۱-۳-۳ معیار شباهت مورد کاربرد

یک ایده ساده برای محاسبه شباهت دو مورد کاربرد آن است که برای هر یک، لیستی شامل کلمات موجود در نام آن مورد کاربرد و همچنین کلمات موجود در نام عامل‌ها یا موارد کاربرد مرتبط با آن مورد کاربرد ایجاد کرده و سپس لیست کلمات مربوط به دو مورد کاربرد را با یکدیگر مقایسه کرده و با استفاده از معیارهای مشابهت متنی، میزان شباهت آن‌ها را محاسبه کرد. چنین روشی در کارهایی نظیر [124~~124~~] استفاده شده است. با این حال، استفاده از شباهت متنی ساده دارای مشکلات شناخته شده‌ای است. به عنوان مثال، ممکن است دو مورد کاربرد، مثلا *'Buy Book'* و *'Purchase Product'*، از نظر معنا و مفهوم خیلی شبیه هم باشند اما به لحاظ متنی شباهت خیلی کمی داشته باشند.

بمنظور محاسبه شباهت دو مورد کاربرد، جنبه‌ها یا اجزای مختلف آن‌ها باید مورد توجه قرار گیرد. این اجزا را می‌توان به دو گروه تقسیم کرد: اجزای متنی و اجزای رابطه‌ای. برای مورد کاربرد UC_i، اجزای متنی عبارتند از S_i، B_i و C_i و اجزای رابطه‌ای عبارتند از E_i، I_i و A_i. معیار پیشنهادی برای محاسبه شباهت دو مورد کاربرد UC_i و UC_j بر اساس رابطه (۳-۱) تعریف می‌شود.

$$sim(UC_i, UC_j) = W_{sem} * semSim_{UC}(UC_i, UC_j) + W_{rel} * relSim(UC_i, UC_j) \quad (۱-۳)$$

که در این رابطه، $semSim_{UC}(UC_i, UC_j)$ شباهت معنایی اجزای متنی دو مورد کاربرد UC_i و Uc_j است و $relSim(UC_i, UC_j)$ شباهت رابطه‌ای یا به بیان دیگر شباهت معنایی اجزای رابطه‌ای دو مورد

کاربرد UC_i و UC_j را مشخص می‌کند. همچنین W_{sem} و W_{rel} پارامترهایی هستند که وزن هر یک از دو شباهت فوق را در شباهت نهایی دو مورد کاربرد مشخص می‌کنند.

در ادامه، نحوه محاسبه شباهت معنایی و همچنین شباهت رابطه‌ای توضیح داده می‌شود.

شباهت معنایی

در رابطه با اجزای متنی توجه به این نکته حائز اهمیت است که تنها با مقایسه متنی این اجزا نمی‌توان شباهت آنها را بدرستی اندازه‌گیری کرد. به عنوان مثال، برای دو مورد کاربرد UC_i و UC_j، اگر داشته باشیم $name_i$='remve user' و $name_j$='delete user' آنگاه B_i={'remove'} و B_j={'delete'}. حال اگر B_i و B_j را از نظر متنی با هم مقایسه کنیم شباهت خیلی کمی به هم دارند، در حالی که از نظر معنایی خیلی شبیه هم هستند و هر دو عمل مشابهی را معرفی می‌کنند. در نتیجه در معیار پیشنهادی، برای جنبه‌های متنی از شباهت معنایی و نه شباهت متنی استفاده می‌شود. برای بدست آوردن شباهت معنایی دو کلمه نیازمند منبعی هستیم که ارتباط کلمه‌های مختلف را به شکلی بیان کرده باشد تا بتوان از روی میزان ارتباط دو کلمه مورد نظر، شباهت معنایی آن‌ها را محاسبه نمود. یکی از معتبرترین منابعی که برای محاسبه شباهت معنایی کلمه‌های انگلیسی استفاده می‌شود شبکه واژگان *WordNet* [212~~212~~] است. الگوریتم‌های مختلفی برای محاسبه شباهت معنایی دو کلمه با استفاده از شبکه واژگانی نظیر *WordNet* ارائه شده که به عنوان نمونه می‌توان به الگوریتم‌های *Lin* [214~~214~~] و *Wu-Palmer* [213~~213~~] اشاره نمود.

با توجه به توضیحات بالا، شباهت معنایی دو مورد کاربرد UC_i و UC_j بر اساس رابطه (۳-۲) محاسبه می‌شود.

$$semSim_{UC}(UC_i, UC_j) = W_S * semSim_S(S_i, S_j) + W_B * semSim_B(B_i, B_j) + W_C * semSim_C(C_i, C_j) \quad (۳-۲)$$

که $semSim_S$، $semSim_B$ و $semSim_C$ به ترتیب برابر شباهت معنایی عنوان سیستم، رفتارهای مورد کاربرد و مفاهیم مورد کاربرد است که محاسبه آنها به ترتیب بر اساس رابطه‌های (۳-۳)، (۳-۴) و (۳-۵) انجام می‌شود. همچنین W_S، W_B و W_C وزن هر یک از این شباهت‌ها می‌باشند.

$$semSim_S(S_i, S_j) = \frac{\sum_{t_m \in S_i} \sum_{t_n \in S_j} (semSim_T(t_m, t_n))}{|S_i| . |S_j|} \qquad (۳-۳)$$

$$semSim_B(B_i, B_j) = \frac{\sum_{t_m \in B_i} \sum_{t_n \in B_j} (semSim_T(t_m, t_n))}{|B_i| . |B_j|} \qquad (۳-۴)$$

$$semSim_C(C_i, C_j) = \frac{\sum_{t_m \in C_i} \sum_{t_n \in C_j} (semSim_T(t_m, t_n))}{|C_i| . |C_j|} \qquad (۳-۵)$$

که $semSim_T(t_m, t_n)$ شباهت معنایی دو کلمه t_m و t_n است که بر اساس یک لغتنامه، هستان-شناسی یا شبکه واژگان محاسبه می‌شود.

شباهت رابطه‌ای

شباهت رابطه‌ای، شباهت دو مورد کاربرد UC_i و UC_j را بر حسب شباهت بین عناصر مرتبط با UC_i و عناصر مرتبط با UC_j محاسبه می‌نماید. این امر مبتنی بر این ایده است که هرچه عناصر مرتبط با UC_i به عناصر مرتبط با UC_j شباهت بیشتری داشته باشند، دو مورد کاربرد UC_i و UC_j نیز شباهت بیشتری به هم دارند. به عنوان نمونه، اگر UC_m عضو E_i و UC_n عضو E_j باشد، آنگاه هرچه UC_m و UC_n شباهت بیشتری به یکدیگر داشته باشند، شباهت UC_i و UC_j نیز بیشتر خواهد بود.

در این رساله، هشت نوع رابطه جهت‌دار با نام‌های R_1 تا R_8 تعریف شده است که یک مورد کاربرد می‌تواند در آن‌ها نقش داشته باشد. یک رابطه از نوع R_k ($1 \le k \le 6$) از UC_i به UC_j وجود دارد اگر و تنها اگر شرط C_k ($1 \le k \le 6$) برقرار باشد، که

C_1: UC_j is a member of EXT_i, i.e. UC_i extends UC_j

C_2: UC_i is a member of EXT_j, i.e. UC_i is extended by UC_j

C_3: UC_j is a member of INC_i, i.e. UC_j is included by UC_i

C_4: UC_i is a member of INC_j, i.e. UC_i includes UC_j

C_5: UC_j is a member of G_i, i.e. UC_j generalizes UC_i

C_6: UC_i is a member of G_j, i.e. UC_i generalizes UC_j

بعلاوه، یک رابطه از نوع R_k ($7 \leq k \leq 8$) از UC_i به یک عامل A وجود دارد اگر و تنها اگر شرط C_k ($7 \leq k \leq 8$) برقرار باشد، که

C_7: there is an association from UC_i to A

C_8: there is an association from A to UC_i

در ادامه، نمادهای زیر را تعریف می‌نماییم:

$UC_{k,i} = \{UC_x \mid \text{there is a } R_k \text{ relation from } UC_i \text{ to } UC_x\}$ for $1 \leq k \leq 6$

$A_{k,i} = \{A_x \mid \text{there is a } R_k \text{ relation from } UC_i \text{ to actor } A_x\}$ for $7 \leq k \leq 8$

با توجه به هشت نوع رابطه‌ای که در بالا تعریف شد، شباهت رابطه‌ای دو مورد کاربرد UC_i و UC_j بر اساس رابطه (۶-۳) محاسبه می‌شود:

$$\text{relSim}(UC_i, UC_j) = \frac{1}{8} \sum_{k=1}^{8} F_k(UC_i, UC_j) \qquad (۶\text{-}۳)$$

که در این رابطه F_k ($1 \leq k \leq 8$) تابعی است که مقداری در بازه [0, 1] بمنزله شباهت دو مورد کاربرد UC_i و UC_j بر اساس رابطه R_k برمی‌گرداند. از بین این هشت تابع، شش تابع نخست توسط رابطه (۷-۳) و دو تابع آخر بر اساس رابطه (۸-۳) تعریف می‌شوند.

$$F_k(UC_i, UC_j) = \frac{\sum_{UC_x \in UC_{k,i}} \sum_{UC_y \in UC_{k,j}} (\text{semSim}_{UC}(UC_x, UC_y))}{|UC_{k,i}| \cdot |UC_{k,j}|} \quad \text{for } 1 \leq k \leq 6 \qquad (۷\text{-}۳)$$

$$F_k(UC_i, UC_j) = \frac{\sum_{A_x \in A_{k,i}} \sum_{A_y \in A_{k,j}} (\text{semSim}_{A}(A_x, A_y))}{|A_{k,i}| \cdot |A_{k,j}|} \quad \text{for } 7 \leq k \leq 8 \qquad (۸\text{-}۳)$$

که $semSim_A(A_x, A_y)$ شباهت معنایی دو عامل A_x و A_y است که بر اساس رابطه (۹-۳) محاسبه می‌شود:

$$semSim_A(A_x, A_y) = semSim_T(\text{name of } A_x, \text{name of } A_y) \quad (۹-۳)$$

۳-۳-۲ تطبیق نمودار فعالیت

پس از آنکه الگوریتم پیشنهاد مورد کاربرد، لیست مرتب k مورد کاربرد پیشنهادی را برای مورد کاربرد ورودی UC_{new} ایجاد کرد، کاربر به بررسی این پیشنهادها پرداخته و موردی را که از نظر خودش مناسب‌تر است انتخاب کرده و به عنوان مورد کاربرد پیشنهادی، UC_{old} به الگوریتم تطبیق نمودار فعالیت می‌دهد. این الگوریتم با دریافت UC_{new} و UC_{old} به عنوان ورودی، نسخه اولیه نمودار فعالیت UC_{new}، یعنی AD_{new}، را با تطبیق نمودار فعالیت مرتبط با UC_{old}، یعنی AD_{old}، تولید می‌نماید. توضیح این الگوریتم در بخش ۱-۲-۳-۳ ارائه می‌شود، اما پیش از آن، مبنای ایده تطبیق نمودار فعالیت به اختصار توضیح داده می‌شود.

چهار حالت کلی در رابطه با مشابهت دو مورد کاربرد UC_{old} و UC_{new} وجود دارد:

۱. دو مورد کاربرد، از نظر رفتار و مفاهیم مربوطه با هم متفاوت هستند.
۲. دو مورد کاربرد، رفتارها و مفاهیم مشابهی دارند.
۳. دو مورد کاربرد، رفتارهای مشابه و مفاهیم متفاوتی دارند.
۴. دو مورد کاربرد، رفتارهای متفاوت و مفاهیم مشابهی دارند.

در حالت نخست چون دو مورد کاربرد هیچ شباهتی به هم ندارند، انتظار نمی‌رود نمودارهای فعالیت مربوط به آن‌ها مشابه هم باشند. در نتیجه انتظار نمی‌رود تطبیق نمودار فعالیت AD_{old} برای ایجاد نمودار فعالیت AD_{new} امیدبخش باشد. در حالت دوم، با توجه به اینکه دو مورد کاربرد از نظر رفتار و مفاهیم مشابه هم هستند، طبیعتا انتظار می‌رود نمودار فعالیت مربوط به آن دو نیز مشابه هم و در نتیجه عمل تطبیق نمودار فعالیت نیز امیدبخش باشد.

در حالت سوم که رفتار دو مورد کاربرد یکسان است با توجه به اینکه اساسا نمودار فعالیت می-خواهد جزئیات یک رفتار را با استفاده از انواع محدودی اجزای گرافیکی توصیف کند انتظار می‌رود نمودارهای فعالیت مربوط به دو مورد کاربرد مشابه، از نظر ساختار و اجزا مشابه هم باشند. البته با توجه به اینکه مفاهیم دو مورد کاربرد متفاوت هستند، انتظار می‌رود تفاوت‌هایی نیز در دو نمودار فعالیت وجود داشته باشد، منتها دیدگاه این رساله آن است که این نقاط تفاوت، عموما خود را در برچسب عناصر موجود در نمودار فعالیت نشان می‌دهند و نه در نوع اجزای بکار رفته در نمودار فعالیت. در نتیجه، نمودار فعالیت UC_{new} باید شبیه نمودار فعالیت UC_{old} باشد با این تفاوت که برچسب‌هایی در نمودار AD_{old} که وابسته به مفاهیم UC_{old} هستند باید برای مفاهیم UC_{new} تطبیق یابند. این ایده پایه‌ای الگوریتم تطبیق نمودار فعالیت است که با جایگزین کردن وابستگی‌های بین AD_{old} و مفاهیم UC_{old} با وابستگی‌های بین مفاهیم UC_{new} می‌خواهد نمودار فعالیت AD_{new} را ایجاد کند.

در حالت چهارم، ایده بر این است که چون دو مورد کاربرد از نظر رفتار با هم متفاوت‌اند در نتیجه ساختار و شمای نمودارهای فعالیت مربوط به آن‌ها نیز متفاوت خواهد بود و این دو نمودار فعالیت قابل تطبیق به یکدیگر نیستند. علیرغم آن که مفاهیم موجود در هر دو مورد کاربرد مشابه هم هستند اما این شباهت منجر به شباهت ساختاری دو نمودار فعالیت نمی‌شود بلکه موجب می‌شود برچسب‌های دو نمودار فعالیت با یکدیگر شباهت‌هایی داشته باشند. در نتیجه در این حالت نیز امیدی نیست که عمل تطبیق نمودارهای فعالیت موفق باشد.

بطور خلاصه، مبنای ایده تطبیق نمودار فعالیت آن است که اگر دو مورد کاربرد دارای رفتارهای مشابه باشند، بواسطه شباهت نمودارهای فعالیت مربوطه‌شان، می‌توان نمودار فعالیت یکی را با تطبیق نمودار فعالیت دیگری ایجاد نمود. این امر بدان معناست که در مرحله پیشنهاد مورد کاربرد، زمانی که مشابهت معنایی مورد کاربرد ورودی با موارد کاربرد موجود در مخزن محاسبه می‌شود بهتر است بیش از مشابهت مفاهیم، به مشابهت رفتار دو مورد کاربرد توجه شود. این نکته، در بخش ۱-۶-۴ که مقادیر وزن-های مربوط به معیار مشابهت ارائه شده مشخص شده است تأیید می‌شود.

۱-۲-۳-۳ الگوریتم تطبیق نمودار فعالیت

الگوریتم تطبیق نمودار فعالیت بر اساس مفهوم حاشیه‌نویسی که پیشتر در بخش ۳-۲-۳ توضیح داده شد کار می‌کند. به بیان ساده اگر هیچ حاشیه‌نویسی برای نمودار فعالیت AD_{old} ایجاد نشده باشد این نمودار فعالیت قابلیت تطبیق ندارد. از سوی دیگر می‌توان گفت هر چه حاشیه‌نویسی‌های بیشتر و دقیق‌تری برای AD_{old} موجود باشد، پتانسیل بیشتری برای تطبیق این نمودار فعالیت برای مورد کاربردهای جدید وجود دارد.

با توجه به انواع مختلف حاشیه‌نویسی‌هایی که الگوریتم حاشیه‌نویسی می‌تواند تولید کند، لازم به ذکر است که در حال حاضر، الگوریتم تطبیق، تنها حاشیه‌نویسی‌های متصل را پوشش می‌دهد. به بیان دیگر هنوز راهکاری برای تطبیق حاشیه‌نویسی‌های غیرمتصل ارائه نشده است و الگوریتم تطبیق به این حاشیه‌نویسی‌ها توجهی نمی‌کند. اگرچه این امر یکی از نقاط ضعف الگوریتم تطبیق ارائه شده بحساب می‌آید، اما ایده این رساله آن است که اگر سیستم مورد نظر بدرستی و با دقت مدل‌سازی شده باشد، بیشتر حاشیه‌نویسی‌های آن باید از نوع متصل باشند. در بخش ارزیابی راهکاری پیشنهادی، با ذکر مثال-هایی، این موضوع بیشتر توضیح داده خواهد شد.

الگوریتم ارائه شده برای تطبیق نمودار فعالیت به شکل شبه کد در شکل ۳-۹ نشان داده شده است. ورودی این الگوریتم عبارت است از:

- UC_{new}: مورد کاربرد مربوط به برنامه کاربردی جدید که قرار است مدل‌سازی شود
- UCD_{new}: نمودار مورد کاربرد مربوط به برنامه کاربردی جدید
- UC_{old}: مورد کاربرد متعلق به برنامه کاربردی موجود که مدل‌های آن در مخزن معنایی مدل‌ها ذخیره شده‌اند.

همچنین، خروجی الگوریتم تطبیق عبارت است از:

- AD_{new}: نسخه اولیه نمودار فعالیت مربوط به UC_{new} که با تطبیق AD_{old}، نمودار فعالیت مربوط به UC_{old}، تولید شده است.

```
Input:
        UC_new: a use case of a new web application WA_new
        UCD_new: use case diagram of UC_new
        UC_old: a use case of an existing web application WA_old
Output:
        AD_new: activity diagram of UC_new
Algorithm
1. retrieve AD_old from model repository
2. AD_new = copy of AD_old
3. labels_old = list of the labels declared in AD_old
4. foreach label L_old in labels_old {
5.      L_new = null
6.      annotations_Lold = list of annotations of L_old
7.      foreach annotation annot_Lold in annotations_Lold {
8.              L = generate equivalent label of L_old based on annot_Lold
9.              if L is not null {
10.                     L_new = L;
11.                     break;
12.             }
13.     }
14.     if L_new is not null {
15.             replace L_old in AD_new with L_new
16.     }
19.}
20. return AD_new
```

شکل ۳-۹. شبه کد الگوریتم تطبیق نمودار فعالیت

الگوریتم تطبیق بدین شرح است: نخست نمودار فعالیت AD_{old} از مخزن بازیابی شده و یک کپی از آن به عنوان نسخه اولیه نمودار فعالیت AD_{new} ایجاد می‌شود (خط‌های ۱-۲). سپس، لیست برچسب-های موجود در نمودار فعالیت AD_{old} بازیابی شده (خط ۳) و برای هر برچسب L_{old} متعلق به این لیست، کلیه حاشیه‌نویسی‌های متصل آن از مخزن معنایی مدل‌ها بازیابی شده و هر یک از این حاشیه‌نویسی‌ها توسط یک زیرالگوریتم تولید برچسب مورد پردازش قرار می‌گیرد (خط‌های ۶ تا ۱۹). زیرالگوریتم تولید برچسب، حاشیه‌نویسی $ANNOT_{old}$ متعلق به برچسب L_{old} را به عنوان ورودی گرفته و برچسب L_{new} را به عنوان خروجی تولید می‌کند (خط ۸). سپس الگوریتم تطبیق، برچسب L_{old} در AD_{new} را با برچسب L_{new} جایگزین می‌کند.

با توجه به توضیح فوق، مشخص می‌شود که قسمت اصلی عملیات تطبیق، زیرالگوریتم تولید برچسب است. همانطور که در بخش ۲-۳-۲-۳ توضیح داده شد، حاشیه‌نویسی متصل $ANNOT_{old}$ را می-

توان بمنزله یک مسیر صریح از AD_{old} به UC_{old} بحساب آورد. ایده زیرالگوریتم تولید برچسب آن است که با پیمایش این مسیر در جهت عکس و تولید یک مسیر معادل از UC_{new} به AD_{new} می‌توان برچسب مورد نیاز، یعنی L_{new}، را ایجاد کرد. این امر، بسته به نوع حاشیه‌نویسی $ANNOT_{old}$، به رویه‌های متفاوتی نیاز دارد که در ادامه توضیح داده می‌شوند.

حاشیه‌نویسی نوع ۱: این نوع حاشیه‌نویسی یک مسیر از AD_{old} به UC_{old} به شکل $[AD_{old} \rightarrow L_{old} \rightarrow CL_{old} \rightarrow C_{old} \rightarrow UC_{old}]$ ایجاد می‌کند. بنابراین مسیر معادل از UC_{new} به AD_{new} باید به شکل $[UC_{new} \rightarrow C_{new} \rightarrow CL_{new} \rightarrow L_{new} \rightarrow AD_{new}]$ باشد که این مسیر از چپ به راست ساخته می‌شود. بدین منظور، ابتدا باید C_{new} ایجاد شود. هر مفهوم متعلق به مجموعه مفاهیم UC_{new} به عنوان یک مقدار مجاز برای C_{new} در نظر گرفته می‌شود. توجیه این امر آن است که C_{old} نیز یکی از عناصر مجموعه مفاهیم UC_{old} بوده است. سپس مقدار CL_{new} نیز برابر مقدار C_{new} در نظر گرفته می‌شود، چرا که CL_{old} نیز با C_{old} تطبیق داشته است. در نهایت، از آنجا که L_{old} شامل نام CL_{old} بوده است، L_{new} نیز با جایگزین کردن CL_{old} با CL_{new} در L_{old} بدست می‌آید.

حاشیه‌نویسی نوع ۲: این نوع حاشیه‌نویسی یک مسیر از AD_{old} به UC_{old} به شکل $[AD_{old} \rightarrow L_{old} \rightarrow A_{old} \rightarrow CL_{old} \rightarrow C_{old} \rightarrow UC_{old}]$ ایجاد می‌کند. در نتیجه مسیر معادل از UC_{new} به AD_{new} باید به شکل $[UC_{new} \rightarrow C_{new} \rightarrow CL_{new} \rightarrow A_{new} \rightarrow L_{new} \rightarrow AD_{new}]$ باشد. ابتدا هر مفهوم متعلق به مجموعه مفاهیم UC_{new} به عنوان یک مقدار مجاز برای C_{new} در نظر گرفته می‌شود. سپس CL_{new} برابر کلاسی در نظر گرفته می‌شود که نامش با نام C_{new} یکسان است. در ادامه، A_{new} برابر هر یک از خصیصه‌های مناسب کلاس CL_{new} در نظر گرفته می‌شود، چرا که A_{old} نیز یکی از خصیصه‌های کلاس CL_{old} بوده است.

چالشی که در اینجا وجود دارد آن است که اگرچه نام کلاس CL_{new} معلوم است (برابر نام C_{new}، یعنی برابر نام هر یک از مفاهیم موجود در مجموعه مفاهیم UC_{new})، اما خصیصه‌های آن ناشناخته‌اند. چرا که هنوز هیچ نمودار کلاسی برای سیستم جدید وجود ندارد (ورودی فرآیند استفاده مجدد، شامل

نمودار کلاس سیستم جدید نمی‌باشد). بنابراین به راهکاری نیاز است که بتواند ابتدا توصیفی از کلاس CL_{new} پیدا کرده و سپس در صورتی که چنین توصیفی بدست آمد، خصیصه‌های این کلاس را بررسی کرده و مناسب‌ترین خصیصه را تشخیص داده و به عنوان نتیجه برگرداند. سپس این خصیصه به عنوان A_{new} در نظر گرفته می‌شود، و از آنجا که L_{old} شامل نام A_{old} بوده است، L_{new} نیز با جایگزین کردن نام A_{old} با A_{new} در L_{old} بدست می‌آید.

دو مشکل در اینجا وجود دارد: نخست، بدست آوردن توصیفی از کلاس مورد نظر و دوم، تشخیص مناسب‌ترین خصیصه این کلاس.

راهکار پیشنهادی برای پیدا کردن کلاس مورد نظر، شامل دو مرحله است. در مرحله نخست، زیرالگوریتم تولید برچسب، به مخزن مراجعه کرده و به دنبال کلاسی با نام مورد نظر می‌گردد. این کلاس ممکن است در نمودار کلاس یکی از سیستم‌های موجود که مدل‌هایش در مخزن ذخیره شده است وجود داشته باشد. اگر چنین کلاسی پیدا شد، به روشی که توضیح داده خواهد شد، خصیصه‌های آن ارزیابی و بهترین مورد انتخاب می‌شود. اما اگر جستجو در مخزن معنایی مدل‌ها نتیجه‌ای نداشت، مرحله دوم اجرا می‌شود.

ایده مرحله دوم، استفاده از فضای داده‌ای است که توسط وب معنایی فراهم شده است. دیدگاه این است که وب معنایی تا حد کافی رشد کرده که بتواند دانش حوزه‌های عمومی و یا تخصصی را به شکلی قابل فهم برای ماشین ارائه کنند. به بیان دیگر انتظار می‌رود دانشی در حد آنچه در یک نمودار کلاس *UML* بازنمایی می‌شود را بتوان از منابع گسترده و غنی موجود در وب معنایی، بخصوص ابر *LOD*، استخراج کرد.

بطور دقیق‌تر، در این مرحله، ابتدا مخزن هستان‌شناسی‌ها مورد جستجو قرار می‌گیرد تا هستان-شناسی‌هایی که کلاسی با نام مورد نظر، یعنی CL_{new}، تعریف کرده‌اند بازیابی شوند. اگر این جستجو نتیجه‌ای نداشت، جستجو در وب معنایی انجام می‌شود. با توجه به اینکه منابع متعدد و متنوعی در وب معنایی وجود دارد، طبیعی است که انجام جستجویی جامع روی همه این منابع قابل انجام نیست. بهمین

دلیل، روش پیشنهادی، در حال حاضر بر روی دو منبع خاص تمرکز دارد که عبارتند از ابر *LOD* و موتور جستجوی *SWOOGLE*. بدین ترتیب ابتدا جستجویی بر روی ابر *LOD* (از طریق درگاه SPARQL آن[۱] می‌باشد) و سپس جستجویی بر روی موتور جستجوی وب معنایی *SWOOGLE*[۲] اجرا می‌شود. نتایج بازیابی شده از این جستجوها، برای استفاده احتمالی در آینده، به عنوان هستان‌شناسی‌های جدیدی در مخزن هستان‌شناسی‌ها ذخیره می‌شوند. مخزن هستان‌شناسی‌ها می‌تواند در ابتدا تهی باشد و یا اینکه از همان ابتدا، هستان‌شناسی‌های مناسبی توسط فرد خبره به آن درون‌بُرد شده باشد. در نهایت با اجرای پرس و جوهایی بر روی مخزن هستان‌شناسی‌ها، کلاس مورد نظر پیدا می‌شود.

پس از آنکه توصیف کلاس مورد نظر پیدا شد، خصیصه‌های آن بازیابی می‌شود. برای تشخیص مناسب‌ترین خصیصه، ابتدا بررسی می‌شود که آیا این کلاس خصیصه‌ای با نام A_{old} دارد یا خیر. اگر جواب مثبت بود این خصیصه به عنوان خصیصه A_{new} در نظر گرفته می‌شود. در غیر اینصورت، خصیصه‌هایی که از نظر نوع داده (مثلا نوع داده رشته‌ای یا عدد صحیح)، با خصیصه A_{old} یکسان هستند، انتخاب شده و از بین آن‌ها خصیصه‌ای که کمترین فاصله معنایی را با خصیصه A_{old} دارد به عنوان خصیصه A_{new} انتخاب می‌شود. فاصله معنایی با استفاده از ترکیبی از فاصله معنایی و فاصله ویرایشی[۳] محاسبه می‌شود. فاصله معنایی توسط ترکیبی از الگوریتم‌های مبتنی بر شبکه واژگان *WordNet* [212~~212~~] نظیر *Wu-Palmer* [213~~213~~] و *Lin* [214~~214~~]، و فاصله ویرایشی نیز توسط الگوریتم *Levenshtein* [215~~215~~] محاسبه می‌شود. اگر هم هیچ خصیصه‌ای که همنوع A_{old} باشد وجود نداشت، از بین همه خصیصه‌های موجود، خصیصه‌ای که کمترین فاصله معنایی با A_{old} را دارد به عنوان خصیصه A_{new} انتخاب می‌شود.

حاشیه‌نویسی نوع ۳: این نوع حاشیه‌نویسی یک مسیر از AD_{old} به UC_{old} به شکل $[AD_{old} \rightarrow L_{old} \rightarrow CL_{old} \rightarrow CL'_{old} \rightarrow C_{old} \rightarrow UC_{old}]$ ایجاد می‌کند. در نتیجه مسیر معادل از UC_{new} به AD_{new} باید به شکل $[UC_{new} \rightarrow C_{new} \rightarrow CL'_{new} \rightarrow CL_{new} \rightarrow L_{new} \rightarrow AD_{new}]$ باشد. ابتدا،

[۱] http://lod.openlinksw.com/
[۲] http://swoogle.umbc.edu/
[۳] Edit distance

هر یک از عناصر موجود در مجموعه مفاهیم UC_{new} به عنوان C_{new} در نظر گرفته می‌شوند. سپس CL'_{new} برابر کلاسی که نامش با نام C_{new} یکسان است در نظر گرفته می‌شود. از آنجا که بین CL_{old} و CL'_{old} رابطه‌ای وجود داشته است، هر کلاسی که با کلاس CL'_{new} همان نوع رابطه را داشته باشد به عنوان CL_{new} در نظر گرفته می‌شود. به عنوان مثال، اگر یک رابطه *Association* از CL'_{old} به CL_{old} وجود داشته باشد، آنگاه باید از CL'_{new} به CL_{new} هم یک رابطه *Association* وجود داشته باشد. در آخر، L_{new} با جایگزین کردن CL_{old} با CL_{new} در L_{old} بدست می‌آید.

روش پیدا کردن توصیف کلاس CL'_{new} پیشتر توضیح داده شد. پیدا کردن کلاس CL_{new} نیز به روش مشابهی انجام می‌شود. از آنجا که CL_{old} و CL'_{old} دو کلاس از نمودار کلاس سیستم SC_{old} هستند، انواع مختلفی از رابطه ممکن است بین آن‌ها وجود داشته باشد: *Association*، *Generalization*، *Aggregation* و *Composition*. برای پیدا کردن CL_{new} که همان نوع رابطه را با CL'_{new} داشته باشد، ابتدا جستجویی در مخزن معنایی مدل‌ها صورت می‌گیرد تا کلاس مورد نیاز از مدل‌های سیستم‌های موجود پیدا شود.

اگر جستجو در مخزن معنایی مدل‌ها نتیجه‌ای نداشته باشد، جستجوی مشابهی در مخزن هستان‌شناسی‌ها انجام می‌شود. با اینحال از آنجا که این مخزن شامل هستان‌شناسی‌هایی است که پیشتر از وب معنایی جمع‌آوری شده‌اند، طبیعی است که از لغات و شِمای متفاوتی، به نسبت مخزن معنایی مدل‌ها، برای توصیف داده‌ها استفاده می‌کنند. در نتیجه، پیدا کردن دو کلاس که رابطه خاصی بین آن‌ها برقرار باشد، در یک هستان‌شناسی که ساختار آن از پیش معلوم نیست، با چالش مواجه است. در حال حاضر، روش پیشنهادی تنها از روابط *Generalization* و *Association* پشتیبانی می‌کند. رابطه نخست با استفاده از مسند *rdfs:subClassOf* قابل جستجو است. به عنوان مثال، پرس و جوهای شکل ۳-۱۰ می‌توانند برای پیدا کردن زیرکلاس‌های کلاس *'Student'* بکار روند.

```
SELECT ?CL2 WHERE {
        ?CL1 rdfs:subClassOf ?CL2  .
        ?CL1 rdfs:label "Student"
}
```

```
SELECT ?CL2 WHERE {
        ?CL1 rdfs:subClassOf ?CL2 .
        FILTER(regex(str(?CL1), "#Student$", "i"))
 }
```

شکل ۳-۱۰. دو نمونه پرس و جوی SPARQL برای بازیابی زیرکلاس‌های کلاس Student

در مورد رابطه *Association*، دیدگاه این است که این نوع رابطه در هستان‌شناسی‌ها، در صفات شیئی[1]بازتاب پیدا می‌کند. به بیان دیگر، یک رابطه *Association* از کلاس CL_1 به کلاس CL_2، در قالب یک صفت شیئی که دامنه و برد آن به ترتیب برابر کلاس‌های CL_1 و CL_2 هستند دیده می‌شود. در نتیجه، می‌توان از پرس و جویی نظیر پرس و جوی شکل ۳-۱۱ برای پیدا کردن کلاس‌هایی که کلاس *'Student'* با آن‌ها رابطه‌ای از نوع *Association* دارد استفاده کرد.

```
SELECT ?CL2 WHERE {
        ?assoc rdf:type         owl:ObjectProperty .
        ?assoc          rdfs:domain  ?CL1 .
        ?assoc rdfs:range  ?CL2 .
        ?CL1   rdfs:label “Student”
}
```

شکل ۳-۱۱. یک نمونه پرس و جوی SPARQL برای بازیابی کلاس‌های مرتبط با کلاس Student

حاشیه‌نویسی نوع ۴: این نوع حاشیه‌نویسی یک مسیر از AD_{old} به UC_{old} به شکل $[AD_{old} \rightarrow L_{old} \rightarrow A_{old} \rightarrow CL_{old} \rightarrow CL'_{old} \rightarrow C_{old} \rightarrow UC_{old}]$ ایجاد می‌کند. در نتیجه مسیر معادل از UC_{new} به AD_{new} باید به شکل $[UC_{new} \rightarrow C_{new} \rightarrow CL'_{new} \rightarrow CL_{new} \rightarrow A_{new} \rightarrow L_{new} \rightarrow AD_{new}]$ باشد. ابتدا، هر یک از مفاهیم موجود در مجموعه مفاهیم UC_{new} به عنوان C_{new} در نظر گرفته می‌شود و CL'_{new} کلاسی در نظر گرفته می‌شود که نامش با نام C_{new} برابر است. سپس هر کلاسی که با CL'_{new} رابطه‌ای از همان نوع رابطه بین CL_{old} و CL'_{old} دارد به عنوان CL_{new} انتخاب می‌شود. هر یک از خصیصه‌های مناسب کلاس CL_{new} به عنوان A_{new} استفاده می‌شود و در نهایت، L_{new} با جایگزین کردن A_{old} با A_{new} در L_{old} بدست می‌آید.

[1] Object properties

حاشیه‌نویسی نوع ۵: این نوع حاشیه‌نویسی یک مسیر از AD_{old} به UC_{old} به شکل $[AD_{old} \rightarrow L_{old} \rightarrow AC_{old} \rightarrow UC_{old}]$ ایجاد می‌کند. در نتیجه مسیر معادل از UC_{new} به AD_{new} باید به شکل $[UC_{new} \rightarrow AC_{new} \rightarrow L_{new} \rightarrow AD_{new}]$ باشد. ابتدا، هر یک از کنش‌گرهایی که از مورد کاربرد UC_{new} استفاده می‌کنند به عنوان AC_{new} انتخاب می‌شوند و سپس L_{new} با جایگزین کردن AC_{old} با AC_{new} در L_{old} بدست می‌آید.

بدین ترتیب، الگوریتم تطبیق نمودار فعالیت، بر اساس قواعدی که در بالا توضیح داده شد، برای هر یک از برچسب‌های AD_{old}، حاشیه‌نویسی‌های آن را استخراج کرده و از روی این حاشیه‌نویسی‌ها، برچسب‌های جدیدی تولید می‌کند. در نهایت با جایگزین کردن برچسب‌های قدیم با برچسب‌های جدید، نمودار فعالیت مربوط به UC_{new} را تولید می‌کند.

۳-۴ خلاصه

در این فصل، رویکرد پیشنهادی این رساله که شامل دو مرحله اصلی است معرفی شده است. در مرحله نخست، یک مخزن معنایی از مدل‌های موجود آماده می‌شود. با بکارگیری فناوری‌های وب معنایی در ایجاد این مخزن، یک لایه معنایی بر روی مدل‌های موجود ایجاد می‌شود که در نتیجه جستجو، بازیابی و دسترسی به اجزای مدل‌های موجود با سهولت و انعطاف‌پذیری بیشتری انجام می‌شود و قابلیت استنتاج خودکار بر روی اطلاعات مدل‌ها فراهم می‌شود. همچنین، یک فرآیند مهم در این مرحله، حاشیه‌نویسی نمودارهای فعالیت موجود در مخزن می‌باشد. این حاشیه‌نویسی‌ها ارتباط بین برچسب‌های نمودارهای فعالیت را با دیگر اجزای مدل‌ها، نظیر نمودار کلاس، بطور صریحی بازنمایی می‌کنند.

در مرحله دوم، فرآیند استفاده مجدد، با استفاده از مخزن معنایی آماده شده اجرا می‌شود. این فرآیند با دریافت نمودار مورد کاربرد یک برنامه کاربردی تحت وب جدید و با استفاده از مخزن مدل‌ها، به ایجاد نمودارهای فعالیت مورد نیاز می‌پردازد. ایجاد نمودارهای فعالیت، بر اساس الگوریتم تطبیق نمودار فعالیت انجام می‌شود که این الگوریتم نیز از وب معنایی و داده‌های پیوندی استفاده می‌کند. همانطور که

در این فصل توضیح داده شد، حاشیه‌نویسی‌های انجام شده در مرحله نخست، اساس کار الگوریتم تطبیق هستند.

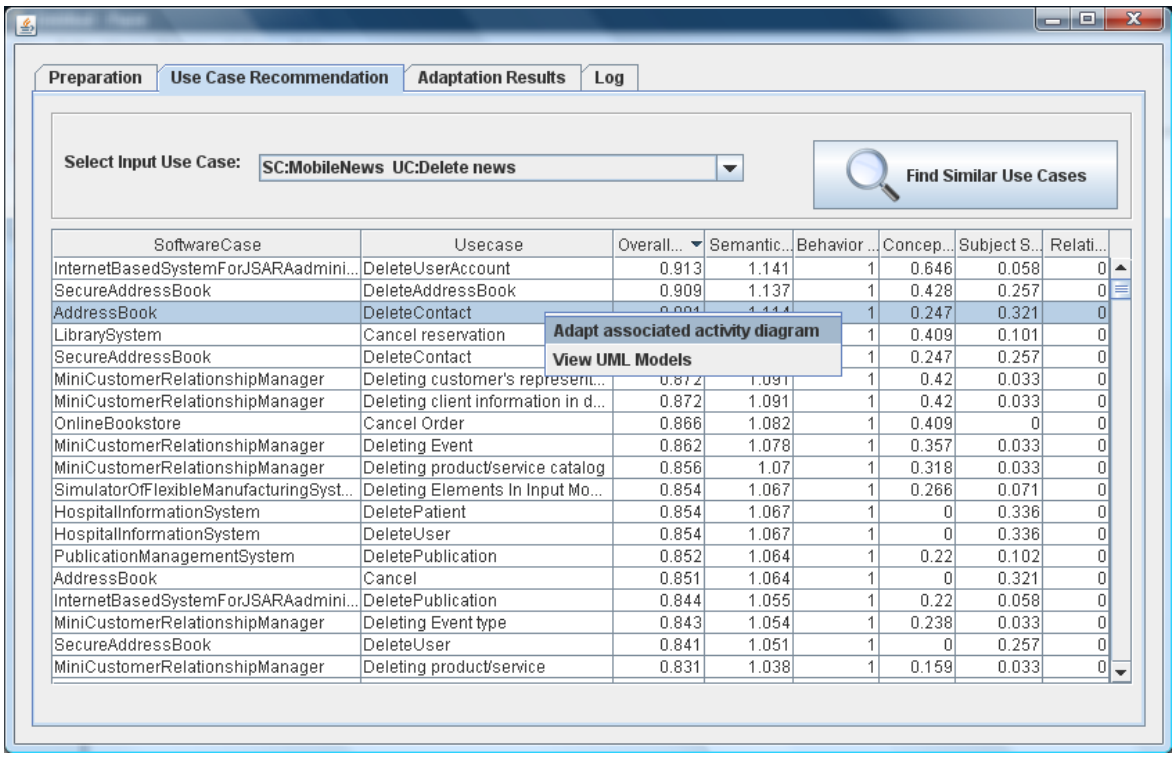
در کل، روش ارائه شده، از دیدگاه کاربر روشی سبک است چرا که الزامات بسیار اندکی بر کاربر تحمیل می‌کند. در واقع، روش ارائه شده تا حد زیادی خودکار می‌باشد. کاربر پس از فراهم کردن نمودار مورد کاربرد ورودی، تنها در یک مرحله، یعنی انتخاب مورد کاربرد مناسب‌تر از بین موارد کاربرد پیشنهاد شده توسط روش ارائه شده، نقش دارد. با توجه به اینکه موارد کاربرد پیشنهادی بر حسب میزان مشابهت با مورد کاربرد ورودی مرتب و به کاربر ارائه می‌شوند، پردازش ذهنی که کاربر باید انجام دهد سنگین و پیچیده نمی‌باشد. در نهایت، پس از آنکه نسخه‌های اولیه نمودارهای فعالیت توسط روش پیشنهادی ایجاد شد، کاربر می‌تواند با اعمال تغییراتی در این نسخه‌ها، نمودارهای فعالیت مورد نظر خودش را ایجاد نماید.

ایده استفاده از مخزن معنایی برای ذخیره مصنوعات نرم‌افزاری ایده جدیدی نیست، اما مباحثی نظیر حاشیه‌نویسی نمودارهای فعالیت و استفاده از پایگاه قوانین در طراحی روش پیشنهادی، جزء جنبه-های نوآوری این رساله می‌باشد. همچنین ارائه یک معیار برای محاسبه شباهت معنایی دو مورد کاربرد *UML* برای نخستین بار انجام شده است. در مرحله دوم، یعنی فرآیند استفاده مجدد نیز الگوریتم‌های ارائه شده برای تطبیق نمودار فعالیت که از داده‌های پیوندی استفاده می‌کنند جزء جنبه‌های نوآوری این رساله می‌باشد. همچنین بر اساس مطالعات انجام شده، روش پیشنهادی، نخستین روش استفاده مجدد از مدل‌ها است که ورودی آن تنها یک نمودار مورد کاربرد است. روش‌های موجود، یا بطور کلی از مورد کاربرد به عنوان مبدأ استفاده مجدد استفاده نمی‌کنند یا توصیف جزئیات مورد کاربرد را نیز به عنوان ورودی از کاربر دریافت می‌کنند.

روش ارائه شده که جزئیات آن در این فصل تشریح شد، می‌تواند برای تولید خودکار مدل نیازمندی‌های عملیاتی یک برنامه کاربردی تحت وب بر اساس نمودار مورد کاربرد مربوط به آن مورد استفاده قرار گیرد. با توجه به نقش و اهمیت این مدل‌ها در کل فرآیند توسعه، فراهم کردن امکان استفاده مجدد بدین شکل می‌تواند هزینه و پیچیدگی کل فرآیند توسعه را کاهش دهد.

۴- ارزیابی

بمنظور ارزیابی روش و الگوریتم‌های پیشنهادی، نمونه اولیه‌ای[۱] به زبان جاوا پیاده‌سازی و آزمایش‌هایی انجام شده است. این نمونه اولیه، از کتابخانه *Jena*[۲] برای پردازش داده‌های معنایی، از کتابخانه *JAWS*[۳] برای محاسبه شباهت معنایی لغات بر اساس *WordNet*، از کتابخانه *SAX*[۴] برای پردازش فایل‌های مدل‌ها و از کتابخانه *OpenRDF*[۵] برای پیاده‌سازی مخزن معنایی مدل‌ها استفاده می‌کند. در شکل ۴-۱ و شکل ۴-۲ دو نما از واسط گرافیکی این نمونه اولیه نشان داده شده است. در شکل اول، لیست مورد کاربردهای پیشنهادی برای مورد کاربرد انتخاب شده توسط کاربر نمایش داده شده است و کاربر با انتخاب یکی از پیشنهادها، عمل تطبیق را اجرا کرده است. نتیجه عمل تطبیق در شکل دوم نشان داده شده است.

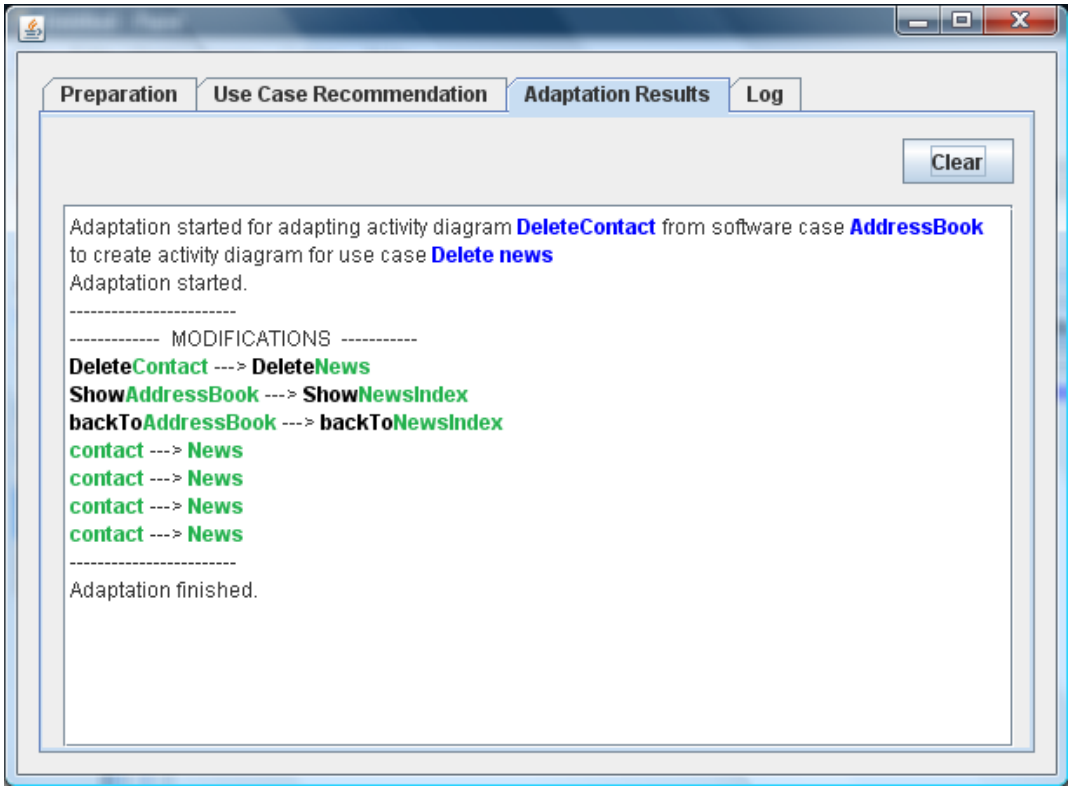


SoftwareCase	Usecase	Overall...	Semantic...	Behavior ...	Concep...	Subject S...	Relati...
InternetBasedSystemForJSARAadmini...	DeleteUserAccount	0.913	1.141	1	0.646	0.058	0
SecureAddressBook	DeleteAddressBook	0.909	1.137	1	0.428	0.257	0
AddressBook	DeleteContact			1	0.247	0.321	0
LibrarySystem	Cancel reservation			1	0.409	0.101	0
SecureAddressBook	DeleteContact			1	0.247	0.257	0
MiniCustomerRelationshipManager	Deleting customer's represent...	0.872	1.091	1	0.42	0.033	0
MiniCustomerRelationshipManager	Deleting client information in d...	0.872	1.091	1	0.42	0.033	0
OnlineBookstore	Cancel Order	0.866	1.082	1	0.409	0	0
MiniCustomerRelationshipManager	Deleting Event	0.862	1.078	1	0.357	0.033	0
MiniCustomerRelationshipManager	Deleting product/service catalog	0.856	1.07	1	0.318	0.033	0
SimulatorOfFlexibleManufacturingSyst...	Deleting Elements In Input Mo...	0.854	1.067	1	0.266	0.071	0
HospitalInformationSystem	DeletePatient	0.854	1.067	1	0	0.336	0
HospitalInformationSystem	DeleteUser	0.854	1.067	1	0	0.336	0
PublicationManagementSystem	DeletePublication	0.852	1.064	1	0.22	0.102	0
AddressBook	Cancel	0.851	1.064	1	0	0.321	0
InternetBasedSystemForJSARAadmini...	DeletePublication	0.844	1.055	1	0.22	0.058	0
MiniCustomerRelationshipManager	Deleting Event type	0.843	1.054	1	0.238	0.033	0
SecureAddressBook	DeleteUser	0.841	1.051	1	0	0.257	0
MiniCustomerRelationshipManager	Deleting product/service	0.831	1.038	1	0.159	0.033	0

شکل ۴-۱. نمایی از واسط گرافیکی نمونه اولیه

---

۱ Prototype
۲ http://jena.apache.org
۳ Java API for WordNet Searching: http://lyle.smu.edu/~tspell/jaws/index.html
۴ Simple API for XML
۵ http://www.aduna-software.com/



شکل ۴-۲. نمایی از واسط گرافیکی نمونه اولیه

اگر چه نمونه اولیه پیاده‌سازی شده از دیدگاه واسط گرافیکی و کاربر پسند بودن، نیاز به بهبودهای زیادی دارد، اما می‌تواند برای ارزیابی روش پیشنهادی مورد استفاده قرار گیرد. در این بخش به گزارش ارزیابی‌هایی که با استفاده از این نمونه اولیه انجام شده است پرداخته می‌شود. آزمایش‌ها بر روی یک سیستم با مشخصات پردازنده از نوع *Intel Core2 Due* (۲,۲۶ و ۲,۲۷ گیگاهرتز)، ۴ گیگابایت حافظه *RAM* و سیستم عامل *Windows Vista* ۶۴ بیتی اجرا شده است.

در ادامه، ابتدا سوالاتی که قرار است در این ارزیابی به آن‌ها پاسخ داده شود مطرح می‌شود. سپس در بخش ۲-۴ نحوه آماده‌سازی مخزن معنایی مدل‌ها که در ارزیابی مورد استفاده قرار گرفته است توضیح داده می‌شود. پس از آن، با ارزیابی قسمت‌های مختلف روش پیشنهادی، سوالات مورد نظر پاسخ داده می‌شود.

۴-۱ سوال‌های تحقیق

یکی از جنبه‌های روش ارائه شده آن است که از وب معنایی به عنوان منبعی برای بدست آوردن اطلاعات مورد نیاز درباره کلاس‌ها و خصیصه‌های آن‌ها استفاده می‌کند. در نتیجه سوالی که مطرح می-شود آن است که:

سوال ۱: آیا می‌توان اطلاعات مربوط به کلاس‌ها و خصیصه‌هایی که در یک نمودار کلاس UML توصیف شده‌اند را با کارایی مناسبی از وب معنایی جمع‌آوری کرد؟

همچنین با توجه به اینکه هدف روش پیشنهادی از جستجوی اطلاعات کلاس‌ها و خصیصه‌ها در وب معنایی، استفاده از این اطلاعات در الگوریتم تطبیق می‌باشد، سوال دیگری که ایجاد می‌شود آن است که:

سوال ۲: استفاده از وب معنایی چقدر بر نتایج الگوریتم تطبیق تأثیر می‌گذارد؟

با توجه به اینکه هدف اصلی روش پیشنهادی، استفاده مجدد از مدل‌های موجود از طریق بکارگیری الگوریتم‌های حاشیه‌نویسی و تطبیق است، سوال اصلی که باید به آن پاسخ داد عبارت است از:

سوال ۳: آیا روش ارائه شده می‌تواند به شکل کارایی، امکان استفاده مجدد از مدل‌های موجود برای ایجاد نمودار فعالیت متناظر با یک مورد کاربرد جدید را فراهم کند؟

برای پاسخ به سوال ۳، لازم است اجزای اصلی روش پیشنهادی که در فرآیند استفاده مجدد دخالت و تأثیر دارند مورد ارزیابی قرار گیرند. بهمین منظور، الگوریتم حاشیه‌نویسی، معیار شباهت مورد کاربرد و الگوریتم تطبیق در بخش‌های جداگانه مورد ارزیابی قرار گرفته‌اند.

۴-۲ آماده‌سازی مخزن معنایی مدل‌ها

بمنظور آماده‌سازی مخزن معنایی مدل‌ها باید مجموعه‌ای از نمونه برنامه‌های کاربردی تحت وب شناسایی و انتخاب شده و مدل‌های *UML* آن‌ها براساس فرآیند پیشنهادی به بازنمایی معنایی تبدیل شود و در نهایت در یک مخزن با ویژگی‌های مورد نظر ذخیره گردد. ضمنا عمل حاشیه‌نویسی نیز باید بر

روی این مدل‌ها انجام شود. در این بخش، ابتدا مجموعه داده‌ای که برای انجام آزمایش‌ها آماده شده است معرفی و سپس فرآیند حاشیه‌نویسی مدل‌ها ارزیابی می‌شود.

۴-۲-۱ مجموعه داده

علیرغم اینکه *UML* رایج‌ترین زبان مدل‌سازی در مهندسی نرم‌افزار است، متأسفانه هیچ مجموعه داده استانداردی از مدل‌های *UML* وجود ندارد که بتوان برای انجام آزمایش‌ها از آن استفاده کرد. در نتیجه، بمنظور ارزیابی روش پیشنهادی، یک مجموعه داده شامل مدل‌های برنامه‌های کاربردی تحت وب جمع‌آوری شد. این مدل‌ها عبارتند از:

- مدل‌های مربوط به ۱۳ برنامه کاربردی تحت وب که بر اساس روشگان *UWE* مدل‌سازی شده‌اند. این مدل‌ها از پایگاه *UWE*[۱] دانلود شده‌اند.
- توصیف مربوط به ۲۷ برنامه کاربردی دیگر از منابعی نظیر کتاب‌های مربوط به *UML* و مدل-سازی شیءگرا [216~~216~~, 217~~217~~, 218~~218~~] جمع‌آوری شده و مدل‌های مربوطه بصورت دستی در ابزار *MagicDraw*[۲] ایجاد شده‌اند.
- توصیف نیازمندی‌های مربوط به ۱۰ برنامه کاربردی دیگر از توسعه‌دهندگان *UCDB* [219~~219~~]دریافت شده و مدل‌های آن‌ها بصورت دستی در ابزار *MagicDraw* ایجاد شده‌اند.

در نهایت، مجموعه داده‌ای شامل مدل‌های مربوط به ۵۰ نمونه برنامه کاربردی تحت وب آماده شد و مترجم توسعه داده شده بر روی فایل‌های مربوط به این مدل‌ها اجرا شده و سه‌گانه‌های *RDF* حاصل، در مخزن معنایی مدل‌ها ذخیره شده است.در جدول ۴-۱~~جدول ۴-۱~~ آماری درباره این فرآیند ارائه شده است.

جدول ۴-۱ آماری درباره فرآیند بازنمایی معنایی مدل‌ها

[۱] http://uwe.pst.ifi.lmu.de/
[۲] http://www.nomagic.com/products/magicdraw.html

تعداد نمونه برنامه‌های کاربردی ورودی	۶۰
تعداد سه‌گانه‌های حاصل از بازنمایی معنایی	۱۵۶۳۶۴
زمان ایجاد بازنمایی معنایی (میلی‌ثانیه)	۸۵۳۲۰

۴-۳ آمادگی وب معنایی

از آنجا که روش پیشنهادی از وب معنایی به عنوان بستری برای تأمین نیازهای اطلاعاتی خود استفاده می‌کند، لازم است میزان آمادگی منابع وب معنایی برای پاسخگویی به این نیازها مورد سنجش قرار گیرد. بدین منظور، آزمایشی انجام شده است تا مشخص شود برای چند درصد از کلاس‌های موجود در مخزن می‌توان یک تعریف مناسب بر روی منابع وب معنایی پیدا نمود. همچنین برای چند درصد از این کلاس‌ها، می‌توان خصیصه‌های مناسبی از وب معنایی استخراج کرد. از آنجا که وب معنایی شامل منابع متعدد و گسترده‌ای است، امکان جستجو بر روی کل وب معنایی وجود ندارد. در نتیجه در این آزمایش، دو منبع مشخص به عنوان نماینده وب معنایی مورد استفاده قرار گرفته‌اند:

- موتور جستجوی *SWOOGLE*[۱]: این موتور جستجو، اسناد وب معنایی را از وب جمع‌آوری و نمایه کرده و امکانات خوبی برای جستجو بر روی آن‌ها فراهم می‌کند. در این آزمایش، برای پیدا کردن یک کلاس خاص به اسم X و همچنین خصیصه‌های آن، بطور خودکار جستجویی بر روی *SWOOGLE* انجام می‌شود تا هستان‌شناسی‌هایی که شامل تعریفی از مفهوم X هستند بازیابی شود. در واقع، جستجویی بر اساس کلمه کلیدی X انجام می‌شود. هستان‌شناسی‌هایی که بدین طریق بازیابی شده‌اند، در یک مخزن محلی ذخیره شده و سپس، با استفاده از پرس و جوهای *SPARQL* شبیه آنچه در شکل ۴-۳~~شکل ۴-۳~~ نشان داده شده است، جستجو بر روی این مخزن انجام می‌شود و کلاس‌های مورد نظر بهمراه خصیصه‌های‌شان بازیابی می‌شوند. لازم به ذکر است که پرس و جوی شکل ۴-۳~~شکل ۴-۳~~، شکل ساده‌ای از پرس و جویی است که در این آزمایش مورد استفاده قرار گرفته است. برخی جزئیات فنی وجود دارد که بر اساس آن می‌توان این پرس و جو را اصلاح کرد تا نتایج بیشتری برگردانده شود. به عنوان مثال، در برخی از هستان‌شناسی‌ها

[۱] http://swoogle.umbc.edu/

برای تعریف یک کلاس، بجای مسند *rdfs:Class*، از مسند *owl:Class* استفاده شده است. می-توان پرس و جوی شکل ۴-۳~~شکل ۴-۳~~ را تغییر داد و با استفاده از عملگر اجتماع[۱] عمل جستجو را برحسب هر دو مسند فوق انجام داد. چنین تغییراتی، اندازه و پیچیدگی پرس و جو را افزایش و میزان *recall* آن را بهبود می‌دهند.

- ابر پروژه *LOD*[۲]. یکی از منابع غنی بر روی وب معنایی، ابر *LOD* است که می‌توان با ارسال پرس و جوهای *SPARQL* به درگاه آن[۳]به جستجو بر روی داده‌هایش پرداخت. برای پیدا کردن یک کلاس خاص بهمراه خصیصه‌هایش، پرس و جویی مشابه آنچه در شکل ۴-۳~~شکل ۴-۳~~ نشان داده شده است بطور خودکار به درگاه *LOD* ارسال شده و نتایج آن که شامل شناسه‌های *URI* مربوط به کلاس‌ها و خصیصه‌های مورد نظر هستند، بازیابی شده است.

```
SELECT ?CL ?attribute WHERE {
        ?CL    rdf:type              rdfs:Class .
        ?CL    rdfs:label      “Student”
OPTIONAL { ?attribute rdfs:domain ?CL . }
}
```

شکل ۴-۳. یک پرس و جوی SPARQL برای بازیابی کلاس ’Student‘ بهمراه خصیصه‌های آن

در این آزمایش، یک مجموعه شامل ۵۰ کلاس از کلاس‌های موجود در مخزن معنایی مدل‌ها به عنوان داده‌های تست انتخاب شده‌اند. برای هر کلاس، هر دو منبع فوق مورد جستجو قرار گرفته و کلاس‌های معادل کلاس مورد نظر، بهمراه خصیصه‌هایشان بازیابی شده‌اند. نتایج این آزمایش در جدول ۴-۲~~جدول ۴-۲~~ نشان داده شده است. همانطور که در این جدول دیده می‌شود، برای ۹۴٪ کلاس‌های مورد آزمون، حداقل یک کلاس معادل از طریق جستجو بر روی *SWOOGLE* بدست آمده است. همچنین برای ۸۸٪ از کلاس‌های مورد آزمون، حداقل یک خصیصه از این طریق پیدا شده است. در مورد جستجو بر روی ابر *LOD*، این مقادیر به ترتیب برابر ۷۲٪ و ۵۰٪ می‌باشد. به عنوان مثال، برای دو کلاس

[۱] Union
[۲] Linking Open Data
[۳] http://lod.openlinksw.com/sparql

آزمون با نام‌های *'Date'* و *'Artist'*، برخی از خصیصه‌هایی که از طریق موتور جستجوی *SWOOGLE* بازیابی شده‌اند در جدول ۴-۳~~جدول ۴-۳~~ نشان داده شده است.

جدول ۴-۲ نتایج آزمایش آمادگی وب معنایی

	جستجو از طریق موتور جستجوی SWOOGLE	جستجو بر روی ابر LOD
کلاس‌های دارای معادل در وب معنایی	۹۴٪	۷۲٪
کلاس‌هایی با حداقل یک خصیصه در وب معنایی	۸۸٪	۵۰٪

جدول ۴-۳ برخی از نتایج جستجو بر روی موتور جستجوی SWOOGLE

نام کلاس	نام خصیصه‌ها
Date	day, month, year, laterThan, date
Artist	artist_description, birthday, collaborated_with, currentProject, gender, hasAddress, hasFax, nativeLanguage, phone, weblog

علاوه بر نتایجی که در جدول ۴-۲~~جدول ۴-۲~~ نشان داده شده است، لازم است زمان اجرای رویه‌هایی که برای جستجو بر روی دو منبع وب معنایی ارائه شده است نیز ارزیابی شود. در مورد جستجو بر روی *SWOOGLE*، زمان اجرا شامل زمان اجرای هر یک از مراحل زیر می‌باشد:

۱. جستجو برای هستان‌شناسی‌هایی که شامل کلمه کلیدی مورد نظر می‌باشند.
۲. دانلود هستان‌شناسی‌های حاصل از جستجو
۳. ذخیره هستان‌شناسی‌ها در مخزن محلی
۴. اجرای پرس و جوی *SPARQL* بر روی مخزن معنایی

لازم به ذکر است که در این آزمایش، در مرحله دوم، لیست هستان‌شناسی‌ها تا زمانی که ۱۰ هستان‌شناسی با موفقیت دانلود شوند یا اینکه لیست به انتها برسد، پردازش شده است. برای دانلود هر هستان‌شناسی، نسخه‌ای که بصورت محلی در پایگاه *SWOOGLE* ذخیره شده دانلود شده است. این امر، در مقایسه با دانلود هستان‌شناسی از پایگاه اصلی آن، دارای چند مزیت است:

- از آنجایی که *SWOOGLE* هستان‌شناسی‌هایی که اندازه‌شان از حد مشخصی کمتر است را بصورت محلی ذخیره می‌کند[1] دانلود نسخه محلی، تضمینی در رابطه با زمان دانلود هستان-شناسی‌ها فراهم می‌کند. در نتیجه از اینکه زمان اجرا بواسطه دانلود هستان‌شناسی‌های خیلی بزرگ هدر رود، جلوگیری می‌شود، چرا که اساسا این هستان‌شناسی‌ها نسخه محلی ندارند و در نتیجه واحد دانلود کننده از آن‌ها صرف نظر می‌کند.
- از آنجا که همه هستان‌شناسی‌ها از یک منبع (پایگاه *SWOOGLE*) دانلود می‌شوند، زمان دانلود نسبتا یکنواخت و قابل تخمین خواهد بود.
- دانلود نسخه محلی احتمال مواجهه با پیوندهای شکسته[2]یا مرده[3]را کاهش می‌دهد.

زمان اجرای هر مرحله برای هر یک از ۵۰ کلاس مورد آزمون اندازه‌گیری شده و مقدار متوسط محاسبه شده است. نتایج در جدول ۴-۴~~جدول ۴-۴~~ نشان داده شده است. همانطور که در این جدول دیده می‌شود، متوسط زمان اجرا برای جستجو از طریق *SWOOGLE* در حدود ۱۵ ثانیه است و زمان دانلود هستان‌شناسی‌ها، سهم عمده‌ای در این زمان دارد. زمان متوسط جستجو بر روی *LOD*، که صرفا نیازمند ارسال پرس و جوی *SPARQL* و بازیابی جواب می‌باشد، برابر ۱۳ ثانیه است که کمی بهتر از جستجو در *SWOOGLE* است. با این حال، اگر بنا باشد نتایج جستجوی هر یک از این دو رویه، به مخزن هستان‌شناسی‌ها افزوده و در آینده مورد استفاده قرار گیرد، جستجو در *SWOOGLE* مفیدتر خواهد بود چرا که بواسطه آن، هستان‌شناسی‌های کاملی بازیابی و به مخزن افزوده می‌شوند، در حالیکه نتیجه جستجو در *LOD* صرفا تعدادی سه‌گانه *RDF* است و نه اسناد یا هستان‌شناسی‌های کامل.

این آزمایش نشان می‌دهد که رویه‌های ارائه شده برای جستجو بر روی وب معنایی، برای استفاده در حالت تعاملی، خیلی مناسب نیستند. اما با این حال، دیدگاه ما این است که با استفاده از روش ارائه شده، به مرور که مخزن هستان‌شناسی‌ها غنی شده و نتایج جستجوهای قبلی بر روی وب معنایی به آن

[1] در زمان اجرای آزمایش‌ها، این حد آستانه برابر ۵ مگابایت می‌باشد.

[2] Broken links
[3] Dead links

افزوده می‌شود، قابلیت پاسخگویی آن نیز افزایش یافته و نیاز به جستجو بر روی وب معنایی کاهش می-یابد. هرچه مخزن محلی غنی‌تر باشد، احتمال اینکه نیازهای اطلاعاتی آتی را پاسخ دهد بیشتر می‌شود و در نتیجه، زمان اجرای جستجو بر روی وب معنایی، اثر کمتری خواهد داشت.

جدول ۴-۴ زمان اجرای دو رویه جستجو بر روی وب معنایی

متوسط زمان اجرا (میلی‌ثانیه)					
جستجو بر روی ابر LOD	جستجو از طریق موتور جستجوی SWOOGLE				
۱۳۱۶۰	مرحله ۱	مرحله ۲	مرحله ۳	مرحله ۴	مجموع
	۲۱۴۵	۱۴۳۸۷	۵۳۴	۱۱۸	۱۵۱۸۴

در مجموع، نتایج این آزمایش، پاسخ مثبتی به سوال شماره ۱ (بخش ۱-۴) می‌دهد. به بیان دیگر، فضای داده‌ای که وب معنایی فراهم می‌کند، برای پاسخ به نیازهای اطلاعاتی روش پیشنهادی امیدوارکننده است. در این آزمایش، تنها دو نمونه از منابع وب معنایی مورد استفاده قرار گرفت، در حالی که منابع دیگری نظیر موتور جستجوی *Sindice*[۱] وجود دارد که آن‌ها نیز می‌توانند در این امر مورد استفاده قرار گیرند. همچنین، دیدگاه ما این است که می‌توان با بهبود پرس و جوهای مورد استفاده، نتایج جستجو را تقویت کرد. برای این منظور لازم است پرس و جوهایی ایجاد شود که روش‌های مختلفی که برای تعریف یک کلاس یا خصیصه می‌تواند استفاده شود را پوشش دهند.

۴-۴ حاشیه‌نویسی مدل‌ها

بمنظور ارزیابی الگوریتم ارائه شده برای حاشیه‌نویسی مدل‌ها، آزمایشی انجام شده است که در این بخش توضیح داده می‌شود. در این آزمایش، ۲۰ نمودار فعالیت از ۳ نمونه برنامه کاربردی تحت وب، از بین مدل‌های موجود در مخزن معنایی مدل‌ها انتخاب شده است. ابتدا این نمودارهای فعالیت بصورت دستی و توسط شرکت کنندگان در آزمایش، حاشیه‌نویسی شدند. برای این کار از هر فرد خواسته شده برچسب‌های موجود در هر نمودار فعالیت را با موجودیت‌های مناسب از مدل‌های مربوط به آن برنامه

[۱] http://sindice.com

کاربردی حاشیه‌نویسی کند. موجودیت‌های مجاز نیز عبارتند از کلاس‌هایی که در نمودار کلاس تعریف شده‌اند، خصیصه‌های این کلاس‌ها و همچنین عامل‌های موجود در نمودار مورد کاربرد مربوطه.

نتایج حاشیه‌نویسی دستی به عنوان استاندارد طلایی[۱] در نظر گرفته شده است که برخی اطلاعات مربوط به آن در جدول ۴-۵~~جدول ۴-۵~~ و جدول ۴-۶~~جدول ۴-۶~~ نشان داده شده است. لازم به ذکر است که بطور متوسط، هر نمودار فعالیت موجود در استاندارد طلایی دارای ۱۰ حاشیه‌نویسی می‌باشد. همانطور که در جدول ۴-۶~~جدول ۴-۶~~ دیده می‌شود، در حدود ۹۱٪ حاشیه‌نویسی‌های استاندارد طلایی از نوع حاشیه‌نویسی‌های متصل، یعنی از انواع ۱ تا ۵، هستند که تطبیق آن‌ها توسط الگوریتم تطبیق ارائه شده پشتیبانی می‌شود. همچنین، تقریبا ۷۹٪ این حاشیه‌نویسی‌های متصل (که ۷۲٪ کل حاشیه‌نویسی‌ها هستند) از انواع ۱ و ۲ هستند. این بدان معناست که تطبیق موفق این نوع حاشیه‌نویسی‌ها نقش بسزایی در موفقیت کل الگوریتم تطبیق دارد.

[۱] Golden Standard

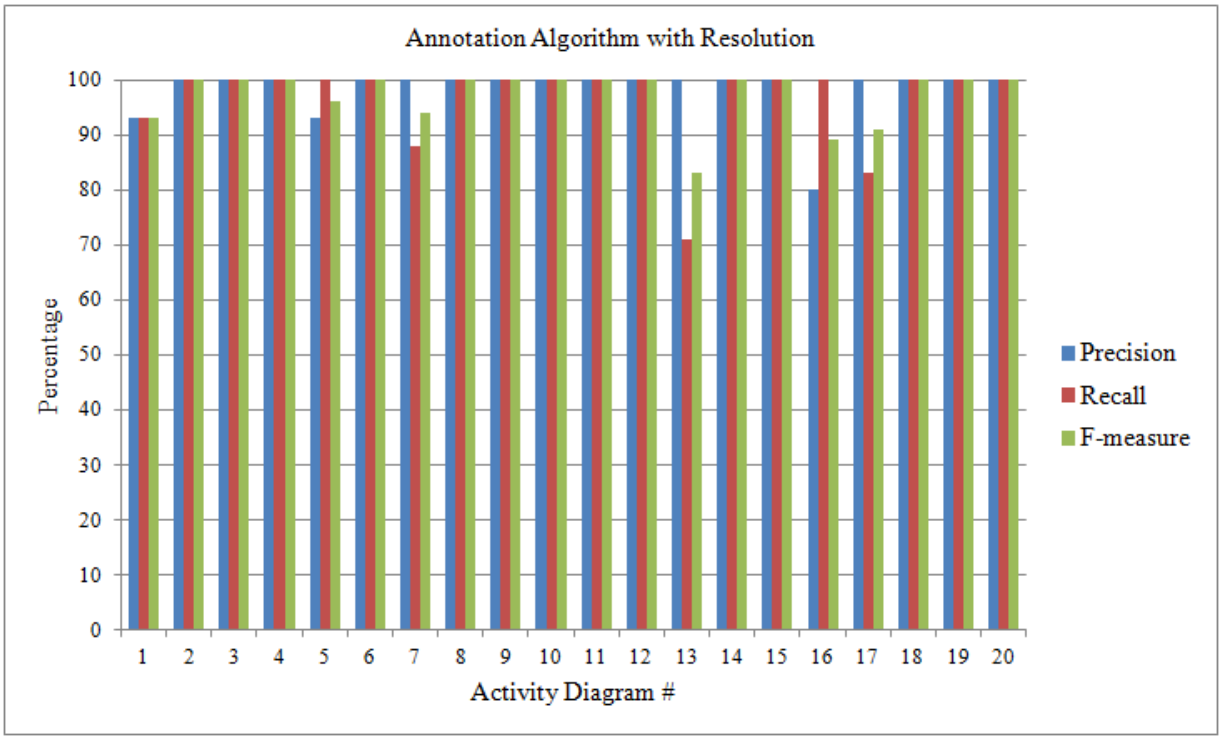
جدول ۴-۵ اطلاعاتی درباره بخشی از استاندارد طلایی برای ارزیابی الگوریتم حاشیه‌نویسی

نام برنامه کاربردی تحت وب	عنوان نمودار فعالیت	تعداد حاشیه‌نویسی-ها
Philoponella	Browse LinkInfos	۱۴
	Add Image	۵
	Add Friend	۷
	Add Favorite	۶
	Add Comment	۲۶
	Adapt Suggestions	۶
	ChangeBrowseSettings	۹
	Adapt LinkList	۹
IMDB	VoteMovie	۸
	SelectTheaters	۴
	Register	۱۰
	Register2	۱۴
	Rate Comment	۶
	Logout	۱
	Login	۵
	CommentMovie	۸
	Buy Ticket	۱۲
Address Book with Content Updates	CreateContact	۱۸
	DeleteContact	۷
	UpdateContact	۱۸

جدول ۴-۶ اطلاعاتی درباره استاندارد طلایی برای ارزیابی الگوریتم حاشیه‌نویسی

	تعداد	درصد
حاشیه‌نویسی‌های نوع ۱	۶۲	۳۲
حاشیه‌نویسی‌های نوع ۲	۷۷	۴۰
حاشیه‌نویسی‌های نوع ۳	۱۸	۹
حاشیه‌نویسی‌های نوع ۴	۱۳	۷
حاشیه‌نویسی‌های نوع ۵	۶	۳
حاشیه‌نویسی‌های از انواع دیگر	۱۷	۹
کل حاشیه‌نویسی‌ها	۱۹۳	۱۰۰

پس از آماده‌سازی استاندارد طلایی، الگوریتم حاشیه‌نویسی ارائه شده بر روی نمودارهای فعالیت مورد آزمون اجرا شده و نتایج آن با استاندارد طلایی مقایسه شده است. همانطور که در شکل ۴-۴ نشان داده شده است، کمترین مقدار *precision* و *recall*، به ترتیب برابر ۸۰٪ و ۷۱٪ است. همچنین مقدار متوسط *precision*، *recall* و *f-measure* به ترتیب برابر ۹۸٪، ۹۷٪ و ۹۷٪ می‌باشد که انحراف معیار آن‌ها نیز به ترتیب برابر ۵٪، ۸٪ و ۵٪ است. این بدان معناست که الگوریتم حاشیه‌نویسی ارائه شده، با توجه به استاندارد طلایی، از دقت و کارایی بسیار خوبی برخوردار است. بعلاوه، زمان متوسط برای حاشیه‌نویسی یک نمودار فعالیت برابر ۲۳۰ میلی‌ثانیه است که نشان می‌دهد الگوریتم ارائه شده از دیدگاه زمان اجرا، کارایی خوبی دارد. نکته آخر اینکه همانطور که در بخش ۲-۳-۲-۳ اشاره شد، بهتر است تعداد حاشیه‌نویسی‌های ایجاد شده برای یک برچسب، عدد کوچکی باشد. در ارتباط با این موضوع، لازم به ذکر است که در این آزمایش، برای برچسب‌هایی که دارای حداقل یک حاشیه‌نویسی بوده‌اند، تعداد متوسط حاشیه‌نویسی‌های ایجاد شده برابر ۱٫۱ حاشیه‌نویسی بوده است که نشان می‌دهد الگوریتم ارائه شده، از این منظر نیز امیدوارکننده است.



شکل ۴-۴. نتایج ارزیابی الگوریتم حاشیه‌نویسی نمودار فعالیت

به عنوان مثال، نمودار فعالیت *'CreateContact'* از برنامه کاربردی *'AddressBook'* در شکل ۴-۵~~شکل ۴-۵~~ نشان داده شده است. حاشیه‌نویسی‌های انجام شده بصورت دستی (توسط فرد خبره) در این شکل شماره‌گذاری شده‌اند و جزئیات‌شان در جدول ۴-۷~~جدول ۴-۷~~ ذکر شده است. خوشبختانه، الگوریتم حاشیه‌نویسی ارائه شده در تشخیص همه این حاشیه‌نویسی‌ها موفق عمل کرده است.

اگرچه نتایج ارزیابی الگوریتم حاشیه‌نویسی نشان از کارایی خوب و دقت بالای آن دارد اما این الگوریتم دو نقطه ضعف اصلی دارد. نخست آنکه چون الگوریتم حاشیه‌نویسی از تطبیق *N-gram* برای مقایسه برچسب عناصر نمودار فعالیت و نام کلاس‌ها، خصیصه‌ها و عامل‌ها استفاده می‌کند، نسبت به تفاوت‌های متنی حساسیت دارد. به عنوان مثال، اگر در نمودار فعالیت نشان داده شده در شکل ۴-۵~~شکل ۴-۵~~ برچسب *'postalAddrMain'* با برچسب *'postalAddressMain'* جایگزین شود، آنگاه الگوریتم حاشیه‌نویسی نمی‌تواند برچسب جدید را با خصیصه *'postalAddrMain'* از کلاس *'Contact'* حاشیه-نویسی کند، در حالی که چنین حاشیه‌نویسی می‌تواند درست تلقی شود.

نقطه ضعف دوم الگوریتم حاشیه‌نویسی آن است که اگر در مدل‌های یک برنامه کاربردی عناصری با نام‌های یکسان وجود داشته باشد آنگاه ممکن است مقصد یک حاشیه‌نویسی بطور نادرست تشخیص داده شود. به عنوان نمونه فرض کنید نمودار کلاس یک برنامه کاربردی شامل دو کلاس *'Book'* و *'DVD'* می‌باشد و هر دو کلاس نیز خصیصه‌ای با نام *'title'* دارند. حال اگر برچسب یک نمودار فعالیت شامل کلمه *'title'* باشد، ممکن است الگوریتم حاشیه‌نویسی نتواند بدرستی تشخیص دهد که کلمه *'title'* باید با خصیصه *'title'* از کدام یک از دو کلاس فوق حاشیه‌نویسی شود.

برای رفع نقطه ضعف اول، لازم است بجای استفاده از تطبیق قطعی *N-gram* ها از یک روش تطبیق غیرقطعی استفاده شود که بتواند از تغییرات محدود و جزئی چشمپوشی کند. همچنین برای آنکه این نقطه ضعف اثر کمتری روی نتایج الگوریتم حاشیه‌نویسی داشته باشد، لازم است که عبارات و لغاتی که در برچسب‌های نمودارهای فعالیت استفاده شده‌اند، حتی الامکان با نام‌های کلاس‌ها، خصیصه‌ها و عامل‌ها سازگار و منطبق باشند. به عنوان مثال، اگر نام خصیصه یک کلاس برابر *'last_name'* است، در

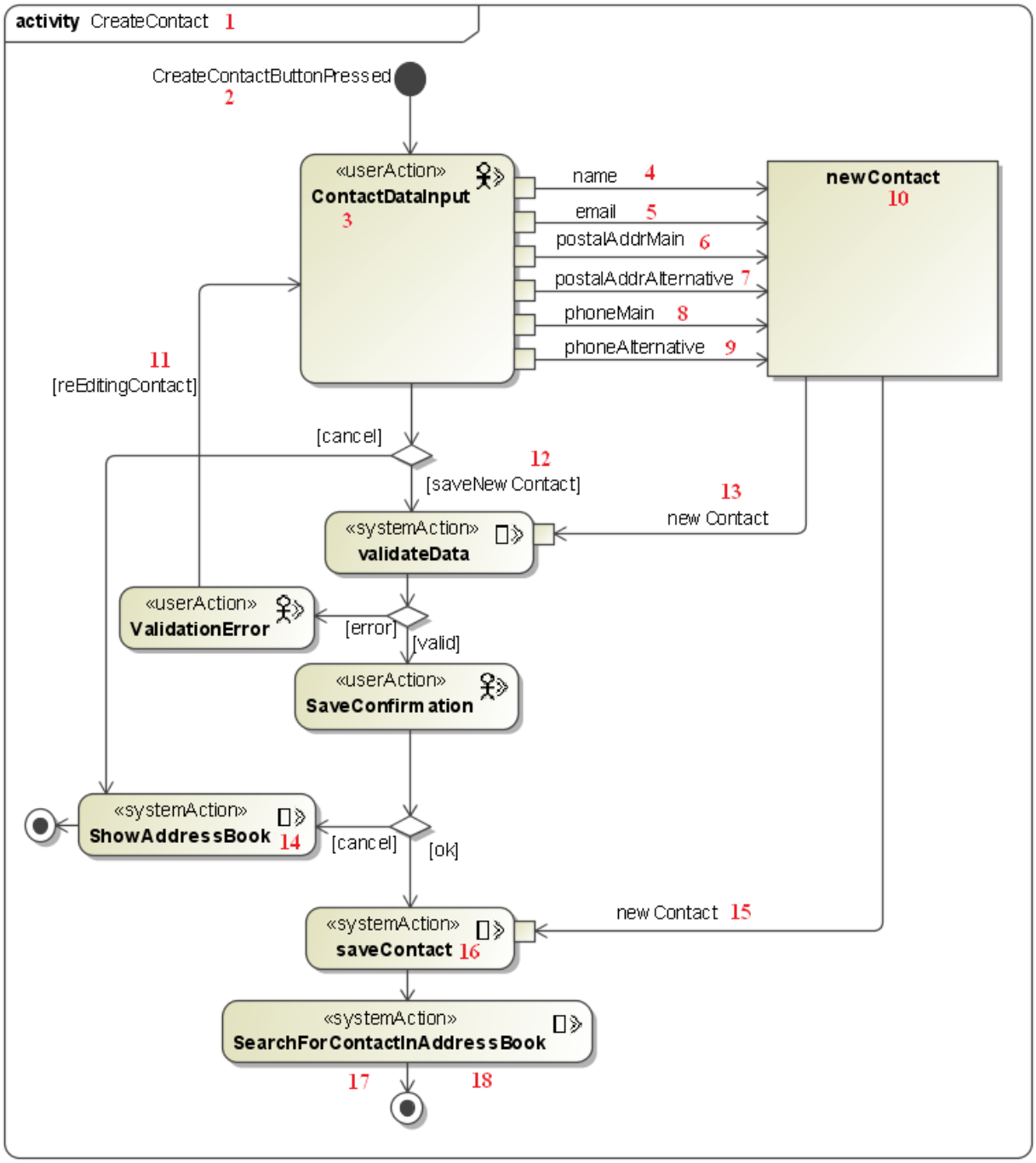
نمودارهای فعالیت مرتبط با آن، برای ارجاع به این خصیصه از عبارت *'family'* یا *'surname'* استفاده نشده باشد.

باور ما بر این است که این الزامات، انتظارات فوق‌العاده‌ای نیستند و رعایت آن‌ها توسط طراحان مدل‌ها امری سخت، پیچیده و نامتداول نیست. همچنین، این نیازمندی‌ها تنها مربوط به روش ارائه شده نمی‌باشد و هر روش دیگری هم که بخواهد پردازش‌های خودکاری را در این سطح، بر روی مدل‌ها فراهم کند نسبت به سازگاری بین عناوین و لغات بکاررفته در مدل‌ها حساس خواهد بود. حتی اگر قرار باشد نمودارهای فعالیت مورد نظر توسط برنامه‌نویسان به کد تبدیل شوند، وجود این قبیل ناسازگاری‌ها، منجر به ابهام و پیاده‌سازی غیردقیق می‌شود. در نتیجه، هرچه مدل‌هایی که در مخزن معنایی مدل‌ها قرار می‌گیرند از سازگاری و دقت بیشتری برخوردار باشند، عمل حاشیه‌نویسی نیز موفق‌تر خواهد بود و در نتیجه کل روش ارائه شده امیدوارکننده‌تر می‌باشد.

برای رفع مشکل دوم، یعنی تشخیص مقصد یک حاشیه‌نویسی، الگوریتم حل تعارض ارائه شده است که از مفهوم اولویت استفاده می‌کند. قواعد ساده‌ای که برای انتساب اولویت به حاشیه‌نویسی‌ها ارائه شده مبتنی بر این ایده است که برای تشخیص درست مقصد یک حاشیه‌نویسی، باید نه تنها به برچسب مورد نظر، بلکه به زمینه و موضوع کل نمودار فعالیت توجه کرد و در ضمن، نام یک نمودار فعالیت انعکاس‌دهنده زمینه و موضوع آن می‌باشد. اگر دو عنصر برای مقصد یک حاشیه‌نویسی قابل انتخاب باشند، آنگاه عنصری که با زمینه مرتبط‌تر و از لحاظ مفهومی به آن نزدیک‌تر باشد از اولویت بیشتری برخوردار است.

به عنوان نمونه در مثالی که پیشتر ذکر شد، اگر نام مورد کاربرد مربوط به نمودار فعالیت مورد نظر برابر *'Buy DVD'* باشد آنگاه مقصد حاشیه‌نویسی، خصیصه *'title'* از کلاس *'DVD'* است و اگر نام این مورد کاربرد برابر *'Buy Book'* باشد آنگاه مقصد حاشیه‌نویسی برابر خصیصه *'title'* از کلاس *'Book'* است. مشخص است که اگر نام مورد کاربرد برابر *'Buy Product'* باشد، آنگاه هر دو گزینه موجود می‌توانند به عنوان مقصد حاشیه‌نویسی انتخاب شوند و گزینه صحیح برای الگوریتم قابل تشخیص نمی‌باشد.

بهر حال، با توجه به اهمیت و نقش الگوریتم حل تعارض، لازم است عملکرد این الگوریتم مورد ارزیابی قرار گیرد. این امر، در قسمت بعد توضیح داده شده است.



شکل ۴-۵. نمودار فعالیت ‘CreateContact’ به همراه حاشیه‌نویسی‌های آن

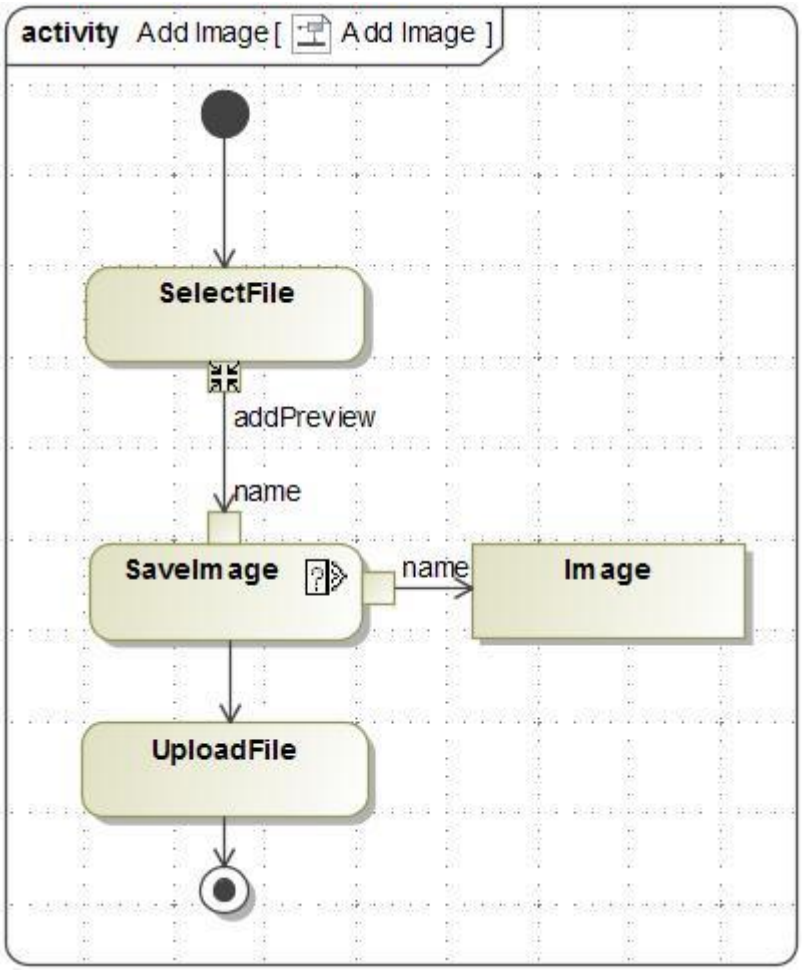
جدول ۴-۷ جزئیات حاشیه‌نویسی‌های نمودار فعالیت 'CreateContact'

شماره	برچسب	مقصد حاشیه‌نویسی	نوع حاشیه‌نویسی
۱	CreateContact	کلاس Contact	۱
۲	CreateContactButtonPressed	کلاس Contact	۱
۳	ContactDateInput	کلاس Contact	۱
۴	name	خصیصه name از کلاس Contact	۲
۵	email	خصیصه email از کلاس Contact	۲
۶	postalAddrMain	خصیصه postalAddrMain از کلاس Contact	۲
۷	postalAddrAlternative	خصیصه postalAddrAlternative از کلاس Contact	۲
۸	phoneMain	خصیصه phoneMain از کلاس Contact	۲
۹	phoneAlternative	خصیصه phoneAlternative از کلاس Contact	۲
۱۰	newContact	کلاس Contact	۱
۱۱	reEditingContact	کلاس Contact	۱
۱۲	saveNewContact	کلاس Contact	۱
۱۳	newContact	کلاس Contact	۱
۱۴	showAddressBook	کلاس AddressBook که با کلاس Contact رابطه‌ای از نوع Association دارد	۳
۱۵	newContact	کلاس Contact	۱
۱۵	saveContact	کلاس Contact	۱
۱۷	searchForContactInAddressBook	کلاس Contact	۱
۱۸	searchForContactInAddressBook	کلاس AddressBook که با کلاس Contact رابطه‌ای از نوع Association دارد	۳

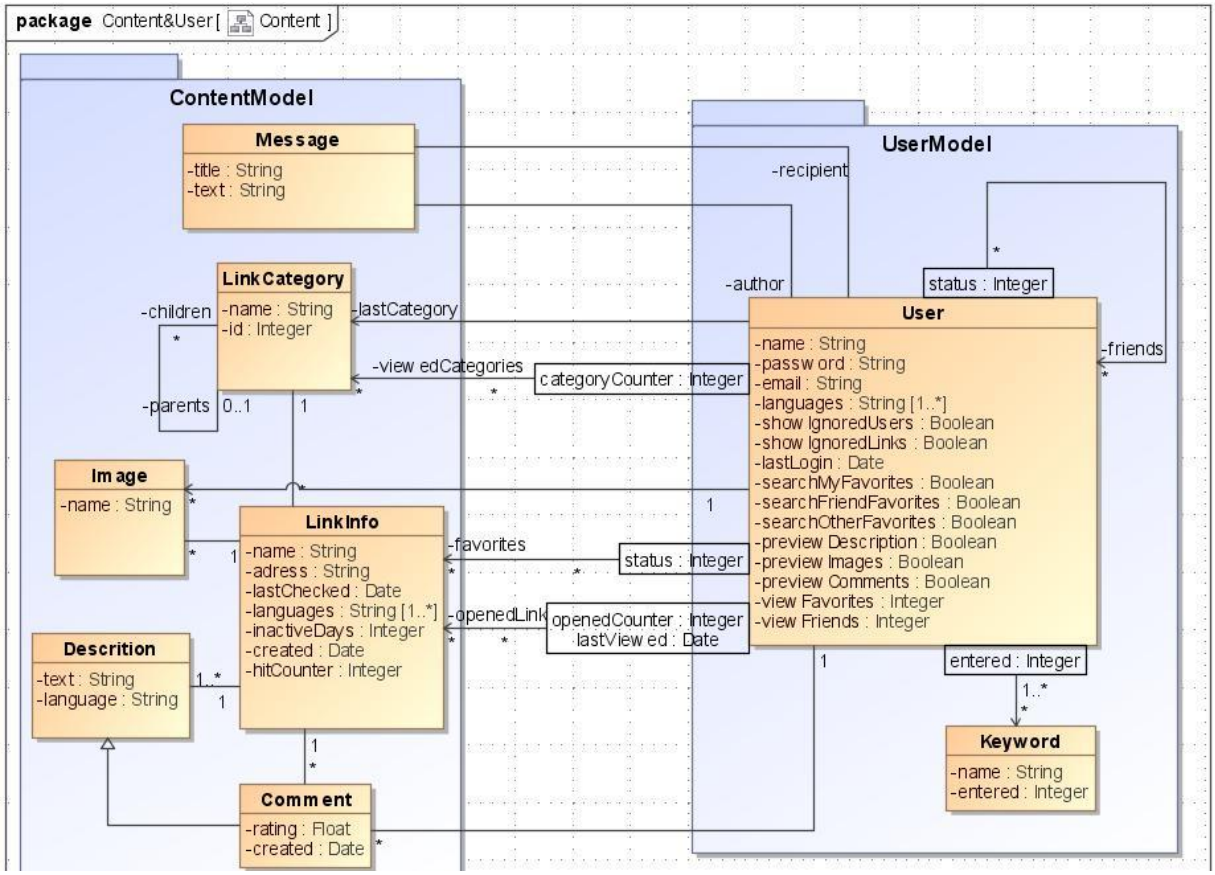
۴-۵ الگوریتم حل تعارض حاشیه‌نویسی

در این بخش، جزئیات بیشتری درباره عملکرد الگوریتم ارائه شده برای حل تعارض حاشیه-نویسی‌ها بیان می‌شود. آزمایشی که در بخش قبل گزارش شده است شامل حاشیه‌نویسی ۲۰ نمودار فعالیت می‌باشد. در این آزمایش، الگوریتم حل تعارض، در مجموع ۵۳ مرتبه (یعنی برای ۵۳ برچسب مختلف) فراخوانی شده است. به بیان دیگر، بطور متوسط برای هر نمودار فعالیت، این الگوریتم ۳ مرتبه فراخوانی شده است. همچنین این فراخوانی‌ها، در مجموع منجر به حذف ۸۴ حاشیه‌نویسی شده است.

به عنوان یک مثال، نمودار فعالیت *'Add Image'* در شکل ۴-۶ ~~شکل ۴-۶~~ نشان داده شده است. این نمودار شامل دو برچسب *'name'* است که الگوریتم حاشیه‌نویسی، ابتدا برای هر کدام از این برچسب‌ها ۵ حاشیه‌نویسی ایجاد کرده است. تفاوت این حاشیه‌نویسی‌ها در مقصدهای آن‌ها است که عبارتند از: خصیصه *'name'* از کلاس‌های *'Image'*، *'LinkCategory'*، *'LinkInfo'*، *'Keyword'* و *'User'*. جهت سهولت درک این حاشیه‌نویسی‌ها، نمودار کلاس مربوطه در شکل ۴-۷ ~~شکل ۴-۷~~ نشان داده شده است. اگر چه همه این ۵ کلاس دارای خصیصه‌ای با نام *'name'* هستند اما الگوریتم حل تعارض، به حاشیه‌نویسی اول سطح اولویت متوسط را انتساب داده است، چرا که مفهوم *'Image'* در نام مورد کاربرد مربوطه (مورد کاربرد *'Add Image'*) ذکر شده است. بعلاوه، الگوریتم حل تعارض، به ۴ حاشیه نویسی باقیمانده سطح اولویت کم را انتساب داده است. در نتیجه این ۴ حاشیه‌نویسی حذف شده و تنها حاشیه‌نویسی اول باقی مانده است که نتیجه درستی هم می‌باشد.



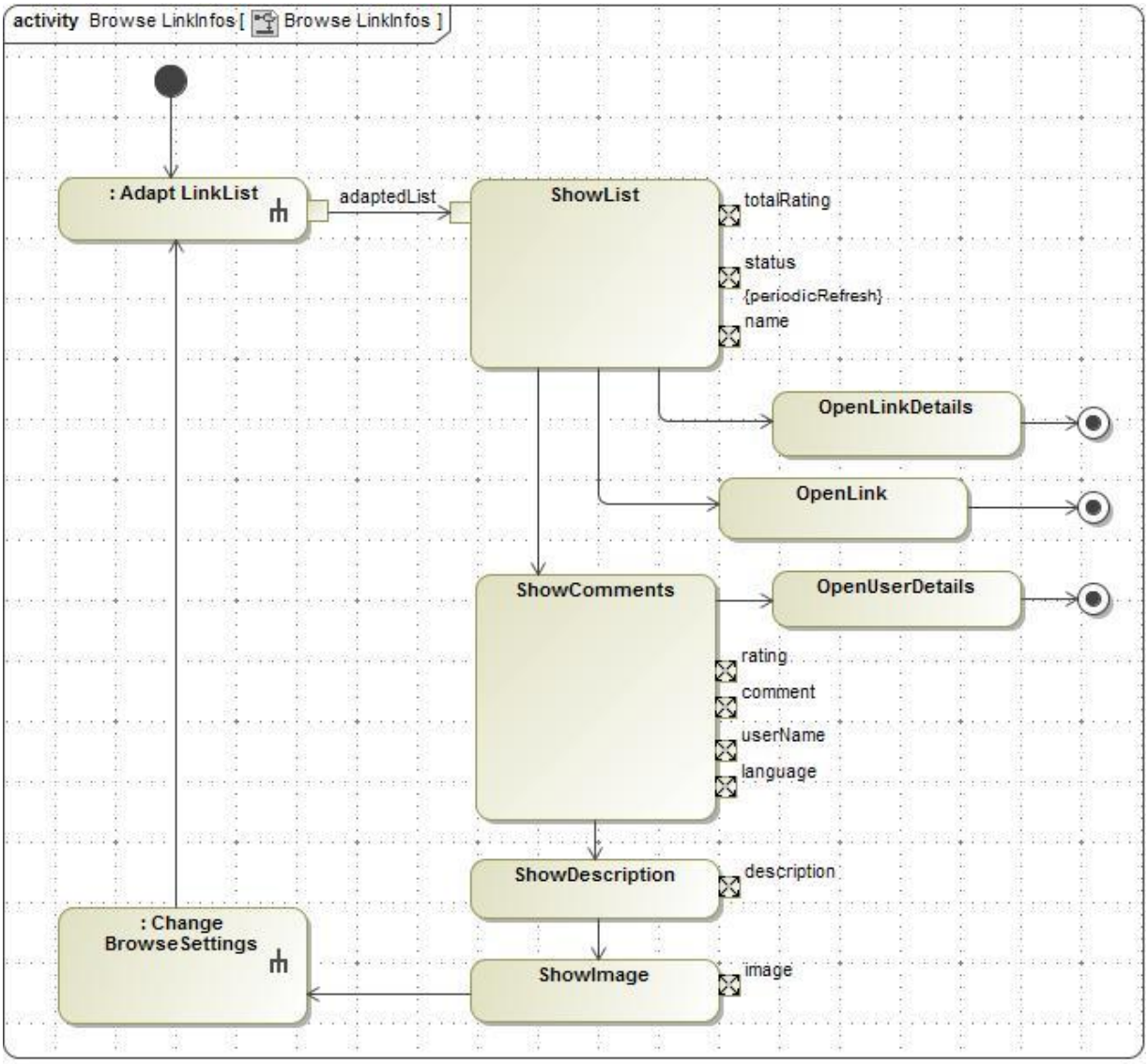
شکل ۴-۶. نمودار فعالیت ‘Add Image’



شکل ۴-۷. نمودار کلاس مربوط به سیستم ‘Philoponella’

به عنوان یک مثال دیگر، نمودار فعالیت *‘Browse LinkInfos’* در **Error! Reference source not found.** ~~شکل ۴-۸~~ نشان داده شده است. این نمودار شامل یک برچسب *‘userName’* است که الگوریتم حاشیه‌نویسی، ۷ حاشیه‌نویسی مختلف برای آن ایجاد کرده است: ۲ حاشیه نویسی برای کلمه *‘user’* و ۵ حاشیه‌نویسی هم برای کلمه *‘Name’*. در مورد کلمه *‘user’* مقصد یکی از حاشیه‌نویسی‌ها کلاس *‘User’* و مقصد حاشیه‌نویسی دوم عامل *‘User’* است. الگوریتم حل تعارض، به هر دو حاشیه-نویسی سطح اولویت کم انتساب داده است. با این حال، بر اساس قاعده سوم از قواعد مربوط به پالایش حاشیه‌نویسی‌ها، حاشیه‌نویسی دوم حذف شده و حاشیه‌نویسی نخست باقی مانده است که درست هم می‌باشد. در مورد کلمه *‘name’*، ۵ حاشیه‌نویسی ایجاد شده که مقصدهای آن‌ها خصیصه *‘name’* از کلاس‌های *‘Image’*، *‘LinkCategory’*، *‘LinkInfo’*، *‘Keyword’* و *‘User’* می‌باشد. از بین این ۵

حاشیه‌نویسی، مورد سوم دارای سطح اولویت متوسط است چرا که نام کلاس *'LinkInfo'* در مجموعه مفاهیم مورد کاربرد مربوطه وجود دارد. حاشیه‌نویسی پنجم دارای سطح اولویت زیاد می‌باشد چرا که برچسب *'userName'* شامل نام کلاس *'User'* که دارای خصیصه *'name'* است می‌باشد. همچنین ۳ حاشیه‌نویسی باقیمانده دارای سطح اولویت کم می‌باشند. در نتیجه الگوریتم حل تعارض، ۴ حاشیه‌نویسی اول را حذف کرده و مورد پنجم را به عنوان حاشیه‌نویسی مناسب‌تر انتخاب کرده است. در این مثال هم، الگوریتم حل تعارض درست عمل کرده است. لازم به ذکر است که در این مثال، هر دو کلاس *'LinkInfo'* و *'User'* دارای خصیصه *'name'* هستند اما نام کلاس *'User'* در خود برچسب L و در کنار نام خصیصه *'name'* ظاهر شده است در حالیکه نام کلاس *'LinkInfo'* در نام مورد کاربرد و نمودار فعالیت مربوطه ظاهر شده است. با توجه به اینکه فاصله فیزیکی محل ظهور کلاس *'User'* با محل وقوع خصیصه *'name'*، در مقایسه با فاصله کلاس *'LinkInfo'* و خصیصه *'name'* کمتر است، کلاس *'User'* را می‌توان نماینده بهتری برای زمینه برچسب L بحساب آورد و در نتیجه آن را مالک این خصیصه دانست. این امر، در واقع ایده‌ای که الگوریتم حل تعارض بر اساس آن رفتار کرده است را توضیح می‌دهد.

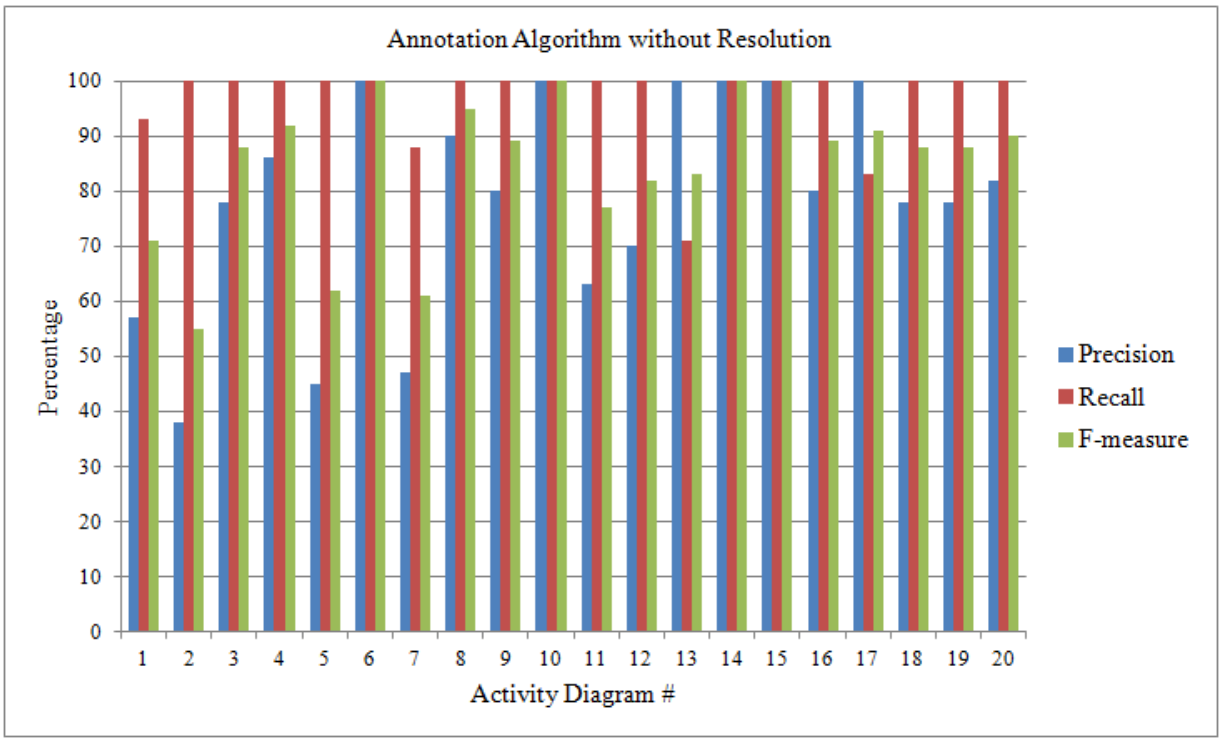


شکل ۴-۸ نمودار فعالیت ‘Browse LinkInfos’

برای آنکه نقش الگوریتم حل تعارض در کیفیت نتایج الگوریتم حاشیه‌نویسی مشخص شود، آزمایش دیگری انجام شده است. در این آزمایش، الگوریتم حاشیه‌نویسی بر روی همان مجموعه داده آزمون مربوط به آزمایش قبل تکرار شده است، اما این بار از الگوریتم حل تعارض استفاده نشده است و در نتیجه همه حاشیه‌نویسی‌های تولید شده توسط الگوریتم حاشیه‌نویسی، حفظ شده و به عنوان جواب برگردانده شده و با استاندارد طلایی آزمایش قبل مقایسه شده‌اند. نتایج این آزمایش در شکل ۴-۹ و جدول ۴-۸ نشان داده شده است.

تحلیل نتایج نشان می‌دهد که تعداد زیادی حاشیه‌نویسی‌های نادرست در بین نتایج وجود دارد. در نتیجه در مقایسه با آزمایش قبلی، مقدار متوسط *precision* و *f-measure* به ترتیب از ۹۸٪ به ۷۹٪ و از ۹۷٪ به ۸۵٪ کاهش یافته است، اما *recall* تغییری نکرده است. عدم تغییر *recall* به معنای آن است که هیچ یک از حاشیه‌نویسی‌هایی که در آزمایش قبلی توسط الگوریتم حل تعارض حذف شده بودند، حاشیه‌نویسی‌های درستی نبوده‌اند. به بیان دیگر، الگوریتم حل تعارض، هیچ حاشیه‌نویسی درستی را به اشتباه حذف نکرده است.

این آزمایش نشان می‌دهد الگوریتم حل تعارض، تأثیر قابل توجهی روی افزایش *precision* الگوریتم حاشیه‌نویسی دارد و در عین حال تأثیر خاصی روی کاهش *recall* ندارد. در نتیجه می‌توان گفت استفاده از الگوریتم ارائه شده برای حل تعارض در الگوریتم حاشیه‌نویسی موجب افزایش دقت و کیفیت آن می‌شود.



شکل ۴-۹. نتایج ارزیابی الگوریتم حاشیه‌نویسی بدون استفاده از الگوریتم حل تعارض حاشیه‌نویسی

جدول ۴-۸ ارزیابی نقش الگوریتم حل تعارض در الگوریتم حاشیه‌نویسی

	الگوریتم حاشیه‌نویسی بهمراه الگوریتم حل تعارض	الگوریتم حاشیه‌نویسی بدون الگوریتم حل تعارض
مقدار متوسط Precision (٪)	۹۸	۷۹
مقدار متوسط Recall (٪)	۹۷	۹۷
مقدار متوسط F-measure (٪)	۹۷	۸۵
کمترین مقدار Precision (٪)	۸۰	۳۸
کمترین مقدار Recall (٪)	۷۱	۷۱
کمترین مقدار F-measure (٪)	۸۳	۵۵

۴-۶ الگوریتم تشخیص مفاهیم و رفتار مورد کاربرد

بمنظور ارزیابی الگوریتم ارائه شده برای تشخیص مفاهیم و رفتار مورد کاربرد، آزمایشی انجام شد که در این قسمت توضیح داده می‌شود. در این آزمایش، مجموعه‌ای شامل ۴۷۵ مورد کاربرد از مخزن مدل‌ها استخراج شده و از شرکت کنندگان خواسته شده است که برای هر مورد کاربرد، کلمه‌ای را در نام آن مشخص کنند که مشخص کننده رفتار آن مورد کاربرد باشد. همچنین کلمه یا کلماتی در نام مورد کاربرد که مفاهیم مورد کاربرد را مشخص می‌کنند. از شرکت کنندگان خواسته شده است که در تشخیص مفاهیم مورد کاربرد، این نکته را در نظر داشته باشند: *"یک لغت C در نام مورد کاربرد، یک مفهوم برای آن مورد کاربرد می‌باشد، بشرطی که لغت C گزینه مناسبی برای نامگذاری یک کلاس در نمودار کلاس سیستم مربوطه، یا خصیصه‌ای از یک کلاس و یا نام یک عامل در نمودار مورد کاربرد باشد"*.

لازم به ذکر است که در بین موارد کاربرد انتخاب شده برای آزمایش، ۹ مورد کاربرد وجود دارد که فاقد رفتار صریح می‌باشند، به عنوان مثال مورد کاربرد *'Home'* یا *'Customer Statistics'*. اگرچه ممکن است گفته شود که رفتار این موارد کاربرد بطور ضمنی بیان شده و برای انسان قابل حدس است و احتمالا منظور از *'Home'* همان *'Show Home'* است و رفتار *'show'* مد نظر است، اما بهرحال از آنجا که در نام این موارد کاربرد هیچ اشاره صریحی به رفتارشان نشده است طبیعی است که الگوریتم تشخیص رفتار که بر اساس نام موارد کاربرد عمل می‌کند قادر به تشخیص این رفتارهای ضمنی و پنهان

نمی‌باشد. به همین دلیل در ارزیابی نتایج، این موارد کاربرد فاقد رفتار در نظر گرفته شده و چنانچه الگوریتم مورد نظر هیچ رفتاری تشخیص نداده است عملکرد آن درست در نظر گرفته شده است. به عنوان مثال برای دو مورد کاربرد *'Home'* و *'Customer Statistics'* هیچ رفتاری تشخیص داده نشده است. در برخی موارد نیز الگوریتم تشخیص رفتار، اشتباه عمل کرده است و این موارد در حکم مثبت نادرست[۱] در نظر گرفته شده‌اند. مثلا برای دو مورد کاربرد *'damage statistics'* و *'lost password'* الگوریتم تشخیص رفتار به اشتباه کلمات *'damage'* و *'lost'* را به عنوان رفتار مورد کاربرد تشخیص داده است. همچنین در موارد محدودی، الگوریتم پیشنهادی به اشتباه، هیچ رفتاری را برای مورد کاربرد *'Unarchive Clinical Records'* یا *'Backup data'* تشخیص نداده است، مثلا برای مورد کاربرد *'backup'* منجر به چنین اشتباهاتی شده است. بررسی دقیق‌تر نشان می‌دهد که ضعف ابزار برچسب‌زننده نقش کلمات و عدم پوشش کلماتی نظیر

نتایج بدست آمده از شرکت کنندگان به عنوان استاندارد طلایی در این آزمایش مورد استفاده قرار گرفته است. الگوریتم ارائه شده برای تشخیص مفاهیم و رفتار مورد کاربرد، بر روی تک تک مورد کاربردهای آزمون اجرا شده و نتایج آن با استاندارد طلایی مقایسه شده است. با توجه به اینکه برای استفاده از این الگوریتم، ابتدا باید مقدار پارامتر چاشنی را تنظیم کرد، در این آزمایش، ۹ مقدار مختلف به عنوان چاشنی مورد بررسی قرار گرفته است. این مقادیر بهمراه نتایج آزمایش در جدول ۴-۹~~جدول ۴-۹~~ نشان داده شده است (در هر ستون، خانه‌هایی که دارای بیشترین مقدار هستند به رنگ خاکستری نشان داده شده‌اند).

در حالت اول، مقدار چاشنی یک رشته تهی انتخاب شده است تا بتوان ضرورت استفاده از چاشنی را بررسی کرد. در هر یک از ۸ حالت دیگر، مقدار چاشنی یک رشته مشخص است. در همه ۹ حالت، از ابزار *Stanford*[۲] برای برچسب زنی نقش کلمات استفاده شده است.

---

۱ False positive

۲ http://nlp.stanford.edu/software/tagger.shtml

همانطور که در جدول ۴-۹~~جدول ۴-۹~~ دیده می‌شود، در تشخیص رفتار مورد کاربرد، همه ۹ حالت از دقت خیلی بالایی، بین ۹۲٪ تا ۹۴٪، برخوردارند، اما مقدار فراخوانی برای حالت‌های مختلف متفاوت است و از ۵۵٪ تا ۹۳٪ تغییر می‌کند. در حالت اول، مقدار فراخوانی خیلی کم است و می‌توان نتیجه گرفت که استفاده از چاشنی مناسب ضروری است. در بین ۸ حالت دیگر، در برخی حالات مقدار فراخوانی کم است (حالت چهارم و نهم)، در حالیکه حداقل ۳ حالت با مقدار فراخوانی خیلی خوب وجود دارد، یعنی حالت‌های دوم، پنجم و هفتم که مقدار فراخوانی‌شان بین ۹۲٪ تا ۰۳٪ است.

در مورد تشخیص مفاهیم، در همه حالات، مقدار فراخوانی خیلی بالا (بین ۹۲٪ تا ٪۹۴) است، اما مقدار دقت کمتر است و در بازه ۶۹٪ تا ۸۷٪ تغییر می‌کند. در حالت اول که عملا از چاشنی استفاده نشده است، دقت الگوریتم تقریبا کمترین مقدار خود را دارد که می‌تواند به عنوان تأکیدی دوباره بر ضرورت استفاده از چاشنی قلمداد شود. همچنین، سه حالت دوم، پنجم و هفتم که در تشخیص رفتار مورد کاربرد، از بقیه موفق‌تر هستند، در تشخیص مفاهیم نیز بهترین نتایج را تولید کرده‌اند. در نتیجه می‌توان نتیجه گرفت که مقدار چاشنی استفاده شده در این حالات، مقدار مناسب و قابل اطمینانی است.

این آزمایش نشان می‌دهد که الگوریتم ارائه شده برای تشخیص مفاهیم و رفتار مورد کاربرد، کارایی و دقت خیلی خوبی دارد. همچنین می‌توان گفت با در نظر گرفتن مقدار چاشنی یکسان، عمل تشخیص رفتار، در مقایسه با تشخیص مفاهیم، از دقت بیشتری برخوردار است، اما با انتخاب بهترین چاشنی، هر دو عمل، از فراخوانی تقریبا مشابهی، یعنی ۹۳٪، برخوردار هستند.

نتیجه دیگری که از این آزمایش بدست می‌آید آن است که انتخاب یک چاشنی مناسب، سخت نیست و همچنین بیش از یک مقدار مناسب وجود دارد. دیدگاه ما این است که انتخاب چاشنی مناسب، با حدس زدن چند مقدار و سپس آزمایش کردن آن‌ها می‌تواند بدون چالش خاصی انجام شود. مقادیری که در این آزمایش بررسی شدند، بر اساس این ایده حدس زده شده‌اند که نام یک مورد کاربرد، معمولا با یک فعل شروع می‌شود و بنابراین چون مقدار چاشنی به ابتدای نام مورد کاربرد افزوده می‌شود، پس چاشنی باید رشته‌ای باشد که بتواند با یک فعل دنبال شده و معنای مناسبی داشته باشد.

در مجموع با توجه به نتایج این آزمایش، رشته *"I want to "* به عنوان بهترین گزینه انتخاب شده و در بقیه آزمایش‌ها به عنوان چاشنی مورد استفاده قرار گرفته است.

اینکه چرا تشخیص مفاهیم مورد کاربرد، در مقایسه با تشخیص رفتار، دقت کمتری دارد، مورد بررسی قرار گرفته است. تحلیل جزئیات نتایج بدست آمده از الگوریتم نشان می‌دهد که تعدادی کلمه وجود دارد که در نام مورد کاربردها ذکر شده‌اند اما عام‌تر از آن هستند که به عنوان مفاهیم مورد کاربرد شناخته شوند، در حالیکه ابزار برچسب‌زنی نقش کلمات، این کلمات را به عنوان مفاهیم مورد کاربرد تشخیص می‌دهد و این امر منجر به افزایش تعداد مثبت نادرست[1] و در نتیجه کاهش دقت الگوریتم می‌شود. به عنوان مثال، در مورد کاربرد *'Save customer information'*، الگوریتم تشخیص مفاهیم، هر دو کلمه *'customer'* و *'information'* را به عنوان مفاهیم مورد کاربرد تشخیص داده است در حالیکه همه شرکت کنندگان در آزمایش، تنها کلمه *'customer'* را انتخاب کرده‌اند. در واقع، نظر شرکت کنندگان این بوده است که کلمه *'information'* یک کلمه عمومی و تکرارشونده است و باید از آن صرفنظر کرد، چرا که *'information'* گزینه مناسبی برای نامگذاری یک کلاس، خصیصه یا یک عامل نمی‌باشد.

از نتایج این تحلیل اینطور نتیجه گرفته می‌شود که استفاده از یک ~~لیست~~ فهرست ایست‌واژه احتمالا به بهبود دقت در تشخیص مفاهیم مورد کاربرد منجر می‌شود. بدین ترتیب می‌توان برخی کلمات پرتکرار و عمومی را از لیست نتایج الگوریتم حذف کرد تا دقت آن افزایش یابد. با این حال، دیدگاه ما این است که این لیست باید بطور خاص و با در نظر گرفتن زمینه بحث، یعنی مورد کاربرد *UML*، تنظیم شود و یک لیست عمومی که در کاربردهای مختلف پردازش متن استفاده می‌شود، مثلا لیست ارائه شده در [220~~220~~]، احتمالا برای استفاده در الگوریتم ارائه شده مناسب نمی‌باشد. برای ارزیابی این موضوع، تغییری در الگوریتم تشخیص مفاهیم اعمال شد تا کلمات موجود در لیست ایست‌واژه‌های [220~~220~~] را از نتایج حذف کند. پس از این تغییر، آزمایش دوباره تکرار شد، اما بهبود خاصی در نتایج آزمایش دیده نشد و دقت و فراخوانی تقریبا ۱٪ کاهش یافت. به عنوان مثال، برای حالت دوم، مقدار دقت از ۸۷٪ به ۸۶٪ و

[1] False Positive

مقدار فراخوانی از ۹۳٪ به ۹۲٪ کاهش یافت. این امر، تأیید کننده این نکته است که در صورت استفاده از لیست ایست‌واژه‌ها، این لیست باید به دقت و با در نظر گرفتن زمینه کاربرد آن تولید شود.

جدول ۴-۹ نتایج اجرای الگوریتم تشخیص مفاهیم و رفتار مورد کاربرد

حالت آزمایش	مقدار رشته چاشنی	تشخیص رفتار			تشخیص مفاهیم		
		Precision (%)	Recall (%)	F-Measure (%)	Precision (%)	Recall (%)	F-Measure (%)
۱	---	۹۲	۵۷	۷۰	۷۰	**۹۴**	۸۰
۲	"I want to "	**۹۴**	**۹۳**	**۹۳**	**۸۷**	۹۳	**۹۰**
۳	"I want to do "	۹۳	۶۵	۷۷	۷۴	**۹۴**	۸۳
۴	"Please "	۹۲	۵۵	۶۹	۷۰	**۹۴**	۸۰
۵	"I "	۹۳	**۹۳**	**۹۳**	**۸۷**	۹۲	۸۹
۶	"I do "	۹۳	۷۳	۸۲	۷۷	۹۳	۸۴
۷	"Do you "	**۹۴**	۹۲	**۹۳**	۸۶	۹۳	۸۹
۸	"Are you "	۹۳	۸۰	۸۶	۸۰	۹۲	۸۶
۹	"Let's "	۹۲	۵۵	۶۹	۶۹	**۹۴**	۸۰

از آنجا که در فرآیند استفاده مجدد، عمل پیشنهاد مورد کاربرد، بر اساس محاسبه شباهت موارد کاربرد انجام می‌شود و در معیار ارائه شده برای محاسبه شباهت دو مورد کاربرد نیز، تشخیص رفتار و مفاهیم مورد کاربرد نقش کلیدی دارد، این مسأله که الگوریتم تشخیص مفاهیم و رفتار، دقت و کارایی خیلی خوبی دارد، این انتظار را ایجاد می‌کند که معیار ارائه شده نیز در تشخیص شباهت موارد کاربرد موفق باشد. این مسأله در آزمایش بعدی مورد بررسی قرار گرفته است.

۴-۶-۱ معیار شباهت مورد کاربرد

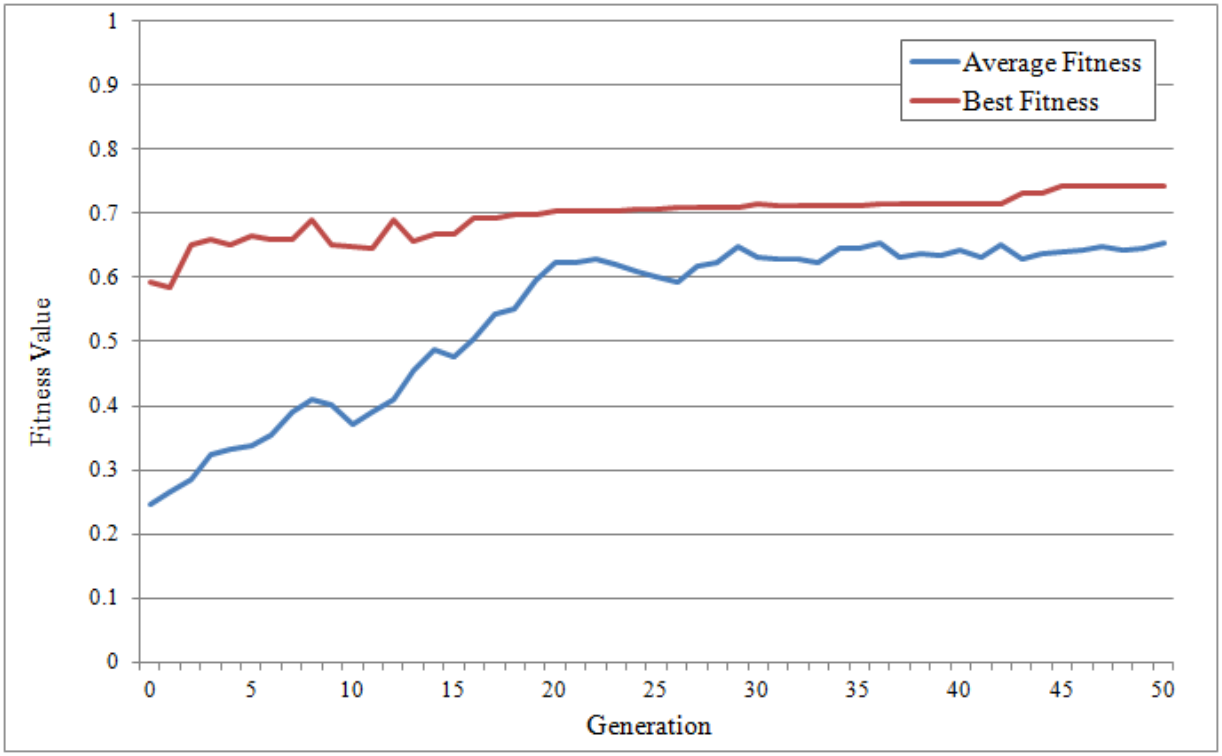
بمنظور ارزیابی معیار ارائه شده برای محاسبه شباهت معنایی دو مورد کاربرد، آزمایشی انجام شده است که در این قسمت توضیح داده می‌شود. اما پیش از آن، لازم است نحوه تنظیم مقدار پارامتر-های بکاررفته در این معیار مشخص شود.

معیار ارائه شده از ۵ پارامتر استفاده می‌کند که هر یک وزن یک عامل در محاسبه شباهت معنایی را مشخص می‌کند. از آنجا که عمل تنظیم وزن‌ها عملی ساده و بدیهی نیست، نمی‌توان این کار را بصورت موردی و ذهنی انجام داد. به همین دلیل، از رویکرد الگوریتم‌های تکاملی برای این منظور استفاده شده است. به بیان دیگر، یک الگوریتم ژنتیک پیاده‌سازی شده است که کارش، بدست آوردن مقادیر مناسب برای پارامترهای مورد نظر می‌باشد. در ادامه، جزئیات این الگوریتم توضیح داده می‌شود.

در الگوریتم ژنتیک مورد نظر، هر کروموزوم دارای ۵ ژن است که هر ژن متناظر با مقدار یکی از پارامترهای هدف می‌باشد. جمعیت نسل اول، شامل ۱۰۰ کروموزوم است که مقدار ژن‌های هر یک بطور تصادفی از بازه [0, 1] انتخاب شده است. ارزیابی برازندگی هر کروموزوم بدین ترتیب انجام می‌شود: هر یک از ژن‌های کروموزوم به عنوان یکی از پارامترها در نظر گرفته می‌شود و بدین ترتیب امکان استفاده از معیار ارائه شده فراهم می‌شود. مجموعه آزمونی شامل ۵ مورد کاربرد از پیش آماده شده، به همراه ۱۰ مورد کاربردی که از نظر افراد خبره، شبیه‌ترین موارد کاربرد به این مورد کاربرد هستند، می‌باشد. سپس برای هر یک از این ۵ مورد کاربرد، بقیه موارد کاربرد موجود در مخزن به کمک معیار ارائه شده ارزیابی و بر حسب میزان شباهت بصورت نزولی مرتب می‌شوند. سپس با مقایسه لیست موارد کاربرد مشابه که توسط فرد خبره تهیه شده است با لیست حاصل از معیار شباهت، میزان برازندگی کروموزوم مربوطه مشخص می‌شود. این مقایسه بر حسب معیار *11pIAP* انجام می‌شود که مقدار متوسط *precision* را برای مقادیر مختلف *recall* از ۰ تا ۱,۰ (با فاصله ۰,۱) محاسبه می‌کند. پس از محاسبه *11pIAP* برای هر ۵ مورد کاربرد، میانگین آن به عنوان مقدار برازندگی کروموزوم مربوطه در نظر گرفته می‌شود.

پس از ارزیابی جمعیت هر نسل، نسب بعدی به این ترتیب ایجاد می‌شود: ابتدا ۵۰ عدد از بهترین کروموزوم‌ها انتخاب شده و به نسل بعد منتقل می‌شوند. سپس، ۵۰ کروموزوم دیگر با استفاده از عمل تقاطع بر روی دو کروموزومی که بطور تصادفی از بین جمعیت نسل جاری انتخاب شده‌اند بدست می‌آیند. در آخر، هر یک از ژن‌های هر کروموزوم، با یک احتمال کوچک، مورد جهش قرار می‌گیرد. مقدار احتمال جهش در ابتدا برابر ۰,۱ است که در طی زمان طبق الگوی مشخصی کاهش می‌یابد.

الگوریتم ژنتیک برای ۵۰ نسل اجرا شده و در آخر، بهترین کروموزوم موجود در آخرین نسل به عنوان نتیجه نهایی انتخاب شده است و ژن‌های آن به عنوان مقدار پارامترهای معیار ارائه شده در نظر گرفته شده‌اند. در شکل ۴-۱۰ ~~شکل ۴-۱۰~~ متوسط برازندگی کروموزوم‌های هر نسل، به همراه بهترین برازندگی مشاهده شده در آن نسل نشان داده شده است. همچنین مقادیر پارامترها که بر اساس الگوریتم ژنتیک مشخص شده‌اند در جدول ۴-۱۰ ~~جدول ۴-۱۰~~ ذکر شده است.



شکل ۴-۱۰ . نتایج الگوریتم ژنتیک برای تعیین مقدار پارامترها

جدول ۴-۱۰ مقدار پارامترهای بدست آمده از الگوریتم ژنتیک

پارامتر	W_{sem}	W_{rel}	W_S	W_B	W_C
مقدار	۰,۹۶	۰,۱۹	۰,۱۸	۰,۹۱	۰,۲۴

بعد از آنکه مقدار پارامترها به کمک الگوریتم ژنتیک تنظیم شد، بمنظور ارزیابی معیار ارائه شده، آزمایشی ترتیب داده شد. در این آزمایش، ۱۰ نمودار مورد کاربرد از مجموعه داده آزمایش انتخاب شده و از هر نمودار نیز یک مورد کاربرد به عنوان مورد آزمون در نظر گرفته شده است. سپس برای هر یک از

۱۰ مورد آزمون، ۱۰ مورد کاربردی که از بقیه موارد کاربرد موجود در مخزن به آن شبیه‌تر هستند توسط افراد خبره مشخص شده است. نتایج بدست آمده در حکم استاندارد طلایی آزمایش هستند. بعد از آن، برای هر مورد کاربرد مورد آزمون، شباهت بقیه موارد کاربرد موجود در مخزن، با استفاده از معیار ارائه شده محاسبه شده و لیست این موارد کاربرد برحسب مقدار شباهت بطور نزولی مرتب شده است. در آخر، با مقایسه لیست مرتب حاصل از معیار ارائه شده و استاندارد طلایی مربوطه، کیفیت معیار ارائه شده مورد ارزیابی قرار گرفته است.

در این آزمایش، دو روش مختلف مورد مقایسه قرار گرفته است. در روش اول، معیار ارائه شده به همراه وزن‌هایی که توسط الگوریتم ژنتیک بدست آمده استفاده شده است. در روش دوم، مقدار همه پارامترهای وزن برابر عدد ثابت ۱,۰ در نظر گرفته شده است. با توجه به نتایج روش دوم می‌توان مشخص کرد که انتساب مقادیر متفاوت به هر یک از پارامترهای وزن و در نتیجه تفاوت قائل شدن بین عوامل مختلف مؤثر در معیار ارائه شده چقدر ضرورت دارد.

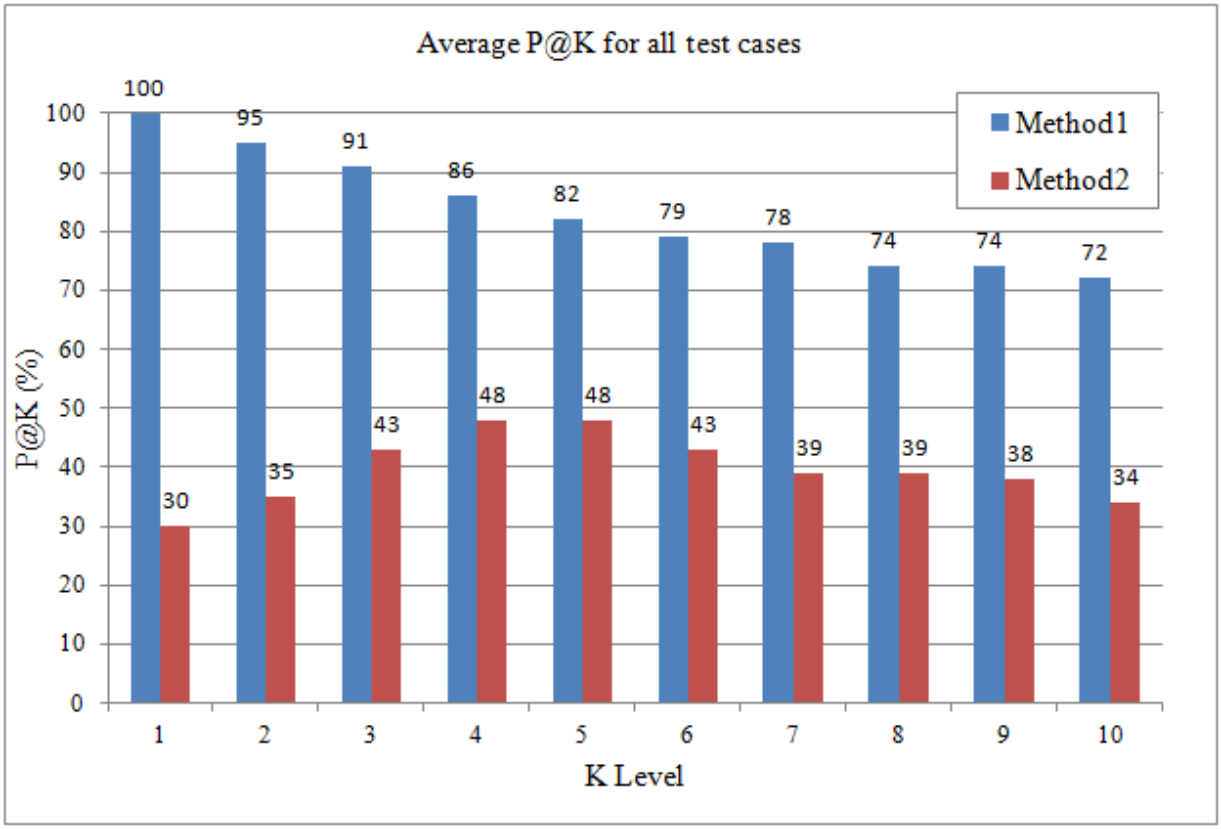
معیارهای مختلفی در این آزمایش مورد استفاده قرار گرفته است. نخست، معیار $P@k$[1] است که *precision* را برای k عنصر ابتدایی لیست مرتب اندازه‌گیری می‌کند. برای هر یک از ۱۰ مورد آزمون، $P@k$ به ازای ۱۰ مقدار متفاوت $k=1$ تا $k=10$ اندازه‌گیری شده است.

در شکل ۴-۱۱~~شکل ۴-۱۱~~، مقدار متوسط $P@k$ برای همه ۱۰ مورد آزمون نشان داده شده است. همانطور که در این شکل دیده می‌شود، در روش اول، متوسط *precision* برای $k=5$ و $k=10$ به ترتیب برابر ۸۲٪ و ۷۲٪ است که یعنی بطور متوسط، چهار جواب درست در بین ۵ جواب اول و ۷ جواب درست در بین ۱۰ جواب اول این روش وجود دارد، که در نتیجه می‌توان کیفیت روش اول را از نظر *precision*، خوب قلمداد کرد.

[1] Precision at k

برای روش دوم، متوسط *precision* برابر *k=5* و *k=10* به ترتیب برابر ۴۸٪ و ۳۴٪ است که این به معنای آن است که بطور متوسط، ۲ جواب درست در بین ۵ جواب اول و سه جواب درست در بین ۱۰ جواب اول وجود دارد که در نتیجه می‌توان کیفیت این روش را از نظر *precision*، ضعیف دانست.

با مقایسه دو روش فوق، مشخص می‌شود که انتساب مقادیر متفاوت به پارامترهای وزن معیار ارائه شده، ضروری است و در نتیجه انتساب صحیح این مقادیر نیز از اهمیت خاصی برخوردار است. با توجه به اینکه *precision* روش اول خوب است می‌توان گفت وزن‌های بدست آمده از الگوریتم ژنتیک، خوب و قابل قبول هستند.



شکل ۴-۱۱. ارزیابی معیار ارائه شده از نظر P@k

معیار دوم که در این آزمایش استفاده شده است *MRR*[1] است که مشخص می‌کند معمولا اولین جواب درست، در چه رتبه‌ای در لیست مرتب خروجی قرار دارد. در مورد روش اول، برای همه ۱۰ مورد آزمون، نخستین جواب صحیح در رتبه ۱ ظاهر شده است و به بیان دیگر مقدار *MRR* برابر ۱ بوده است.

[1] Mean Reciprocal Rank

اطلاعات مربوط به نخستین جواب درست هر مورد آزمون در جدول ۴-۱۱~~جدول ۴-۱۱~~ نشان داده شده است. همچنین رتبه متوسط دومین جواب درست برای روش اول، برابر ۲,۰۹ بوده است که یعنی بطور متوسط، دومین جواب درست در رتبه دوم ظاهر شده است. برای روش دوم، مقدار *MRR* تقریبا برابر ۳ است (در واقع برابر ۲,۸۱) که یعنی بطور متوسط، اولین جواب درست در رتبه سوم ظاهر می‌شود. همچنین برای روش دوم، رتبه متوسط دومین جواب درست برابر ۵,۹۰ است که به معنای آن است که دومین جواب درست، بطور متوسط، در رتبه ۶ ظاهر می‌شود.

نتیجه ارزیابی بر حسب *MRR* نیز نشان‌دهنده آن است که تمایز قائل شدن بین وزن پارامترهای مختلف، به بهبود کیفیت معیار ارائه شده منجر می‌شود و همچنین مؤید این مطلب است که وزن‌های بدست آمده از الگوریتم ژنتیک خوب هستند.

جدول ۴-۱۱ نخستین جواب بدست آمده برای هر مورد آزمون با استفاده از روش اول

مورد آزمون		نتیجه اول	
نام مورد کاربرد	نام برنامه کاربردی	نام مورد کاربرد	نام برنامه کاربردی
Add Comment	Philoponella	Add Item	Library System
Buy Ticket	IMDB	Buy Album	Simple Music Portal
Delete User	SecureAddressBook	Delete User	HospInfo
Search Users	SecureAddressBook	Search Patients	HospInfo
Create New Account	e-retail system	Create a new Blog Account	Content Management System
View Movie	IMDB	ViewPublicationDetails	PVS
ViewAlbumDetails	Simple Music Portal	ViewPersonalDetails	Philoponella
Browse Items	Library Systems	BrowseLinkInfos	Philoponella
Verify Security Permission	Clinical Records System	Check PIN	ATM
Order Books	Online Bookstore	Order Product	Online Shopping

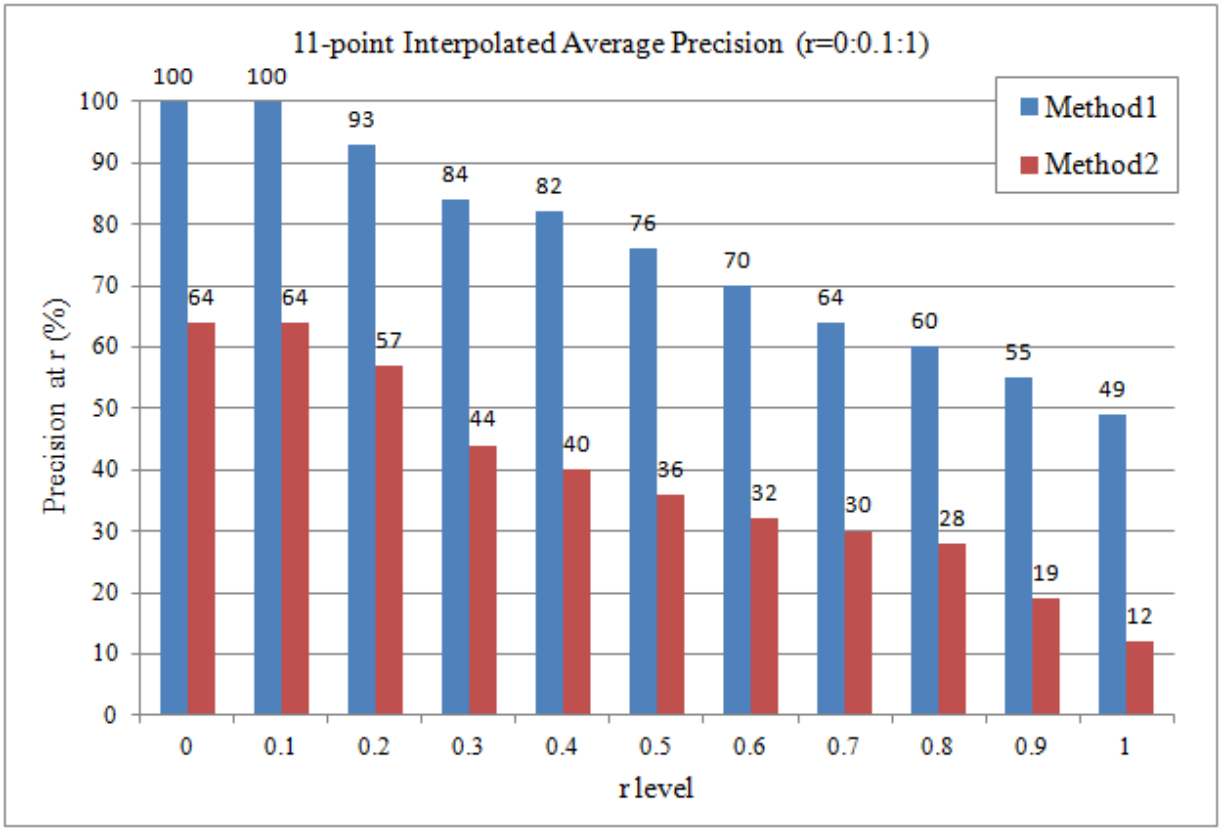
به عنوان سومین معیار ارزیابی، از *11pIAP*[۱] استفاده شده است که مقدار *precision* را در ۱۱ سطح مختلف از *recall*، از سطح *recall=0.0* تا *recall=1.0* محاسبه می‌کند. ابتدا، برای هر مورد آزمون، مقدار *Pinterp(r)*[۲] محاسبه شده است که برابر بیشترین مقدار *precision* بدست آمده برای همه سطوح *recall* برابر *r'* که *r'≥r* می‌باشد. سپس *11pIAP* با محاسبه میانگین *Pinterp(r)* بر روی ۱۰ مورد آزمون مختلف و به ازای ۱۱ سطح مختلف *r* بدست می‌آید.

[۱] 11-point Interpolated Average Precision
[۲] Interpolated precision at level r

نتایج ارزیابی بر حسب معیار *11pIAP* در شکل ۴-۱۲~~شکل ۴-۱۲~~ نشان داده شده است. همانطور که در این شکل دیده می‌شود، در مورد روش اول، حداکثر *precision* قابل حصول در سطح *recall* برابر ۰٫۶، برابر ۷۰٪ است. به بیان دیگر، در سطحی که ۶۰٪ جواب‌های درست برگردانده می‌شوند، حداکثر، ۷۰٪ جواب‌های برگردانده شده درست هستند. همچنین بیشترین سطح *precision* قابل حصول در سطح *recall* برابر ۱٫۰، برابر ۵۰٪ است که یعنی برای آنکه ۱۰۰٪ جواب‌های درست بازیابی شوند، لازم است *precision* در حد ۵۰٪ را پذیرفت (که یعنی تنها نیمی از نتایج برگردانده شده، درست هستند).

درباره روش دوم، مقدار متوسط *precision* در سطح *recall* برابر ۰٫۶ برابر ۳۲٪ و در سطح *recall* برابر ۱٫۰ برابر ۱۲٪ است. به بیان دیگر، در حالتی که ۶۰٪ جواب‌های درست بازیابی می‌شوند، تنها ۳۲٪ جواب‌های بازیابی شده درست هستند. همچنین زمانی که ۱۰۰٪ جواب‌های درست بازیابی می‌شوند، حداکثر، ۱۲٪ جواب‌های بازیابی شده درست هستند.

نتیجه ارزیابی بر حسب معیار *11pIAP* نیز نتیجه بدست آمده بر اساس دو معیار دیگر را تأیید می‌کند. بنابراین می‌توان گفت انتساب وزن‌های متفاوت به پارامترهایی که در معیار ارائه شده در نظر گرفته شده‌اند، ضروری است و در ضمن، الگوریتم ژنتیک ارائه شده، مقادیر این وزن‌ها را بخوبی مشخص کرده است. بنابراین می‌توان گفت معیار ارائه شده، به همراه وزن‌های انتساب داده شده، در مقایسه با نتایج بدست آمده از افراد خبره، نتایج خوبی تولید می‌کند و از *precision* و *recall* خوبی برخوردار است، با اینحال، بهتر است آزمایش‌های انجام شده بر روی مجموعه آزمایشی بزرگتری اجرا شود تا نتایج آن با قطعیت بیشتری قابل پذیرش باشد. یک چالش اساسی در این راستا، عدم وجود یک مجموعه داده بزرگ و استاندارد از مدل‌های *UML* می‌باشد.



شکل ۴-۱۲ ارزیابی معیار ارائه شده از نظر 11pIAP

۴-۷ الگوریتم تطبیق

برای ارزیابی الگوریتم تطبیق ارائه شده، آزمایشی با ۱۰ مورد آزمون انجام شد. هر مورد آزمون شامل یک مورد کاربرد UC_{new} متعلق به برنامه کاربردی WA_{new} به همراه نمودار مورد کاربرد مرتبط با آن و همچنین یک مورد کاربرد UC_{old} متعلق به برنامه کاربردی WA_{old} می‌باشد که از مخزن معنایی مدل‌ها انتخاب شده است. برای هر مورد آزمون، پس از انتخاب مورد کاربرد UC_{new}، مورد کاربرد UC_{old} توسط فرد خبره انتخاب شده است. به بیان دقیق‌تر، از فرد خبره خواسته شده که با بررسی مدل‌های موجود در مخزن معنایی مدل‌ها (البته بجز مدل‌های مربوط به WA_{new})، مناسب‌ترین مورد کاربرد برای تطبیق مورد کاربرد UC_{new} را به عنوان UC_{old} پیدا و معرفی کند.

بعد از آماده‌سازی موارد آزمون، الگوریتم تطبیق بر روی هر یک از آن‌ها اجرا شده و نسخه اولیه نمودار فعالیت AD_{new} بر اساس نمودار فعالیت AD_{old} برای هر مورد آزمون ایجاد شده است. سپس نتایج هر مورد آزمون به دو فرد خبره داده شده تا بصورت همکارانه درباره آن قضاوت نمایند. بطور خاص، از

آن‌ها خواسته شده نمودار فعالیت AD_{new} را بررسی کرده و چنانچه لازم است، با ویرایش برچسب‌هایش آن را اصلاح کنند. به بیان دیگر، افزودن عناصر جدید به AD_{new} یا حذف برخی عناصر آن مجاز نبوده است. پس از اعمال اصلاحات توسط افراد خبره، برچسب‌های ایجاد شده توسط الگوریتم تطبیق که توسط افراد خبره تغییر داده نشده‌اند به عنوان تطبیق‌های درست قلمداد شده و نتایج الگوریتم تطبیق مورد ارزیابی قرار گرفته است.

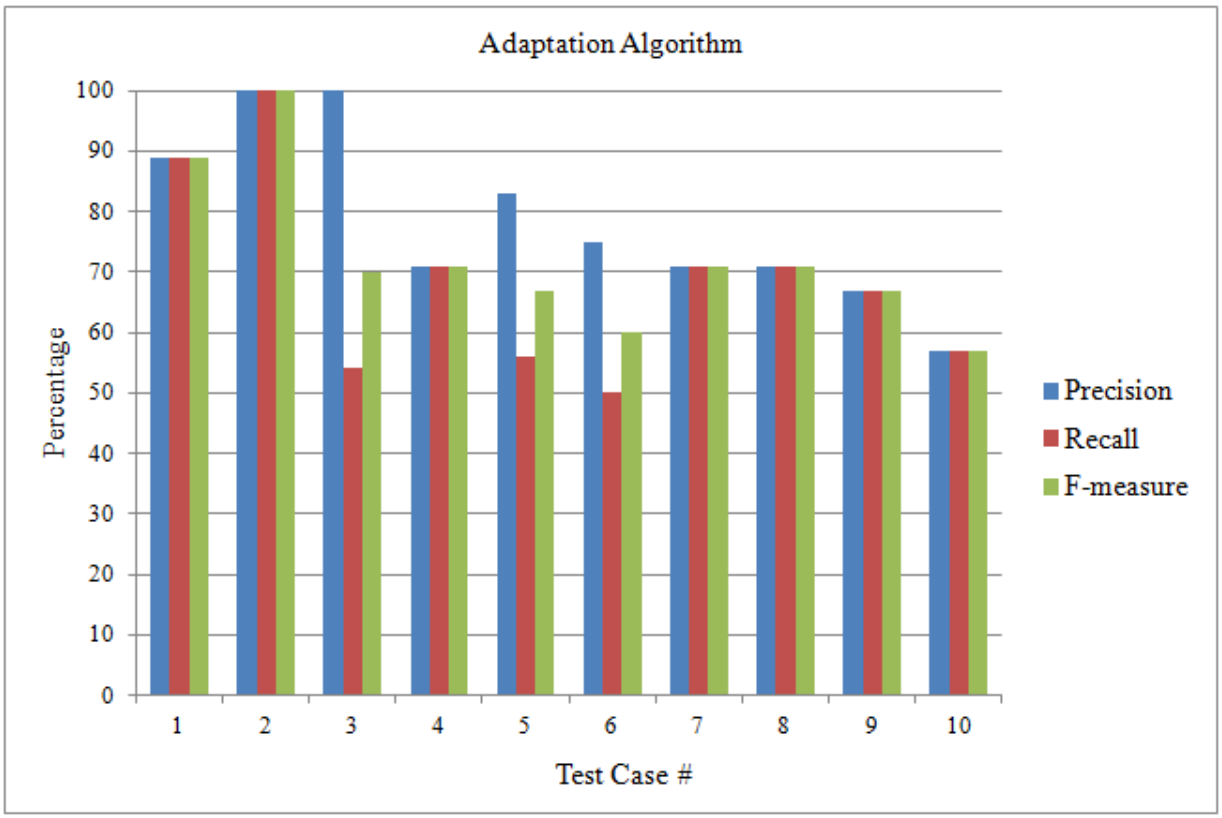
نتایج این آزمایش در جدول ۴-۱۲~~جدول ۴-۱۲~~ و شکل ۴-۱۳~~شکل ۴-۱۳~~ نشان داده شده است. مقدار متوسط *precision*، *recall* و *F-measure* به ترتیب برابر ۷۸٪، ۶۹٪ و ۷۲٪ و مقادیر انحراف معیار نیز به ترتیب برابر ۱۴٪، ۱۶٪ و ۱۳٪ بوده است. همچنین کمترین مقدار *precision* و *recall* به ترتیب برابر ۵۷٪ و ۵۰٪ است.

نتایج آزمایش، به تفکیک نوع حاشیه‌نویسی، در جدول ۴-۱۳~~جدول ۴-۱۳~~ نشان داده شده است. همانطور که در این جدول دیده می‌شود، تطبیق حاشیه‌نویسی‌های نوع ۱ و ۲ به شکل خوب و مؤثری انجام شده است. اما تطبیق حاشیه‌نویسی‌های نوع ۳، ۴ و ۵ با موفقیت کمتری همراه بوده است. همانطور که در بخش ۴-۴ توضیح داده شد، در حدود ۷۲٪ کل حاشیه‌نویسی‌های ایجاد شده برای مجموعه داده آزمون، از نوع ۱ و ۲ هستند که تطبیق آن‌ها با موفقیت و با نرخ ۹۴٪ و ۸۲٪ انجام شده است.

با توجه به اینکه هدف الگوریتم تطبیق، ایجاد نسخه اولیه نمودار فعالیت AD_{new} و نه نسخه نهایی و دقیق آن بوده است، از نتایج بدست آمده می‌توان اینطور نتیجه‌گیری کرد که الگوریتم تطبیق ارائه شده از دقت و کارایی قابل قبولی برخوردار است.

جدول ۴-۱۲ نتایج ارزیابی الگوریتم تطبیق

شماره	نام مورد کاربرد جدید	عنوان برنامه کاربردی جدید	نام مورد کاربرد مورد تطبیق	عنوان برنامه کاربردی مورد تطبیق	تعداد حاشیه‌نویسی‌ها	تعداد حاشیه‌نویسی‌های تطبیق‌یافته	تعداد تطبیق‌های درست
۱	CreatePatient	HospInfo	CreateContact	AddressBook	۱۸	۱۸	۱۶
۲	CreateContact	AddressBook	CreatePatient	HospInfo	۴	۴	۴
۳	DeleteContact	AddressBook	DeletePublication	PVS	۱۳	۷	۷
۴	DeletePublication	PVS	DeleteContact	AddressBook	۷	۷	۵
۵	BuyAlbum	SimpleMusicPortal	Buy Ticket	IMDB	۹	۶	۵
۶	Buy Ticket	IMDB	BuyAlbum	SimpleMusicPortal	۱۲	۸	۶
۷	DeleteContact	AddressBook	RemoveFriend	Philoponella	۷	۷	۵
۸	RemoveFriend	Philoponella	DeleteContact	AddressBook	۷	۷	۵
۹	Update Friend	Philoponella	UpdateContact	AddressBook	۱۸	۱۸	۱۲
۱۰	UpdateContact	AddressBook	Update Friend	Philoponella	۷	۷	۴



شکل ۴-۱۳. نتایج ارزیابی الگوریتم تطبیق

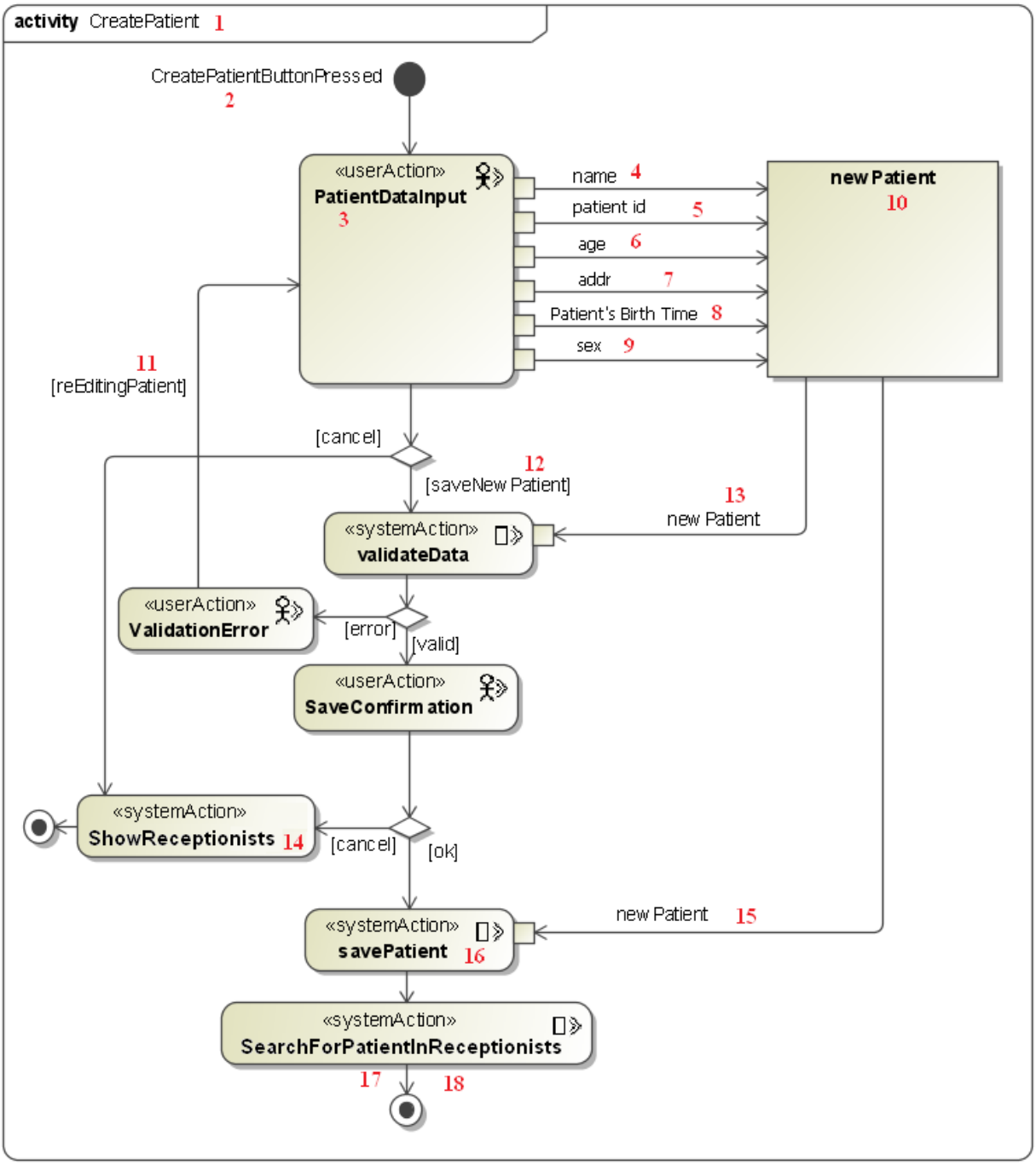
جدول ۴-۱۳ نتایج ارزیابی الگوریتم تطبیق به تفکیک نوع حاشیه‌نویسی

نوع حاشیه‌نویسی	تطبیق درست (٪)	تطبیق نادرست (٪)	عدم تطبیق (٪)
۱	۹۴	۴	۲
۲	۸۲	۱۴	۴
۳	۱۸	۸۲	۰
۴	۲۰	۳۰	۵۰
۵	۰	۳۳	۶۷

به عنوان نمونه، نتایج مربوط به نخستین مورد آزمون در جدول ۴-۱۴ ~~جدول ۴-۱۴~~ و شکل ۴-۱۴ ~~شکل ۴-۱۴~~ نشان داده شده است. همانطور که در جدول ۴-۱۴ ~~جدول ۴-۱۴~~ دیده می‌شود، در مورد آزمون اول، ۲ مورد از ۱۸ مورد تطبیق صورت گرفته، از دیدگاه فرد خبره نادرست بوده است که این تطبیق‌ها مرتبط با حاشیه‌نویسی‌های از نوع ۳ می‌باشند. لازم به ذکر است که برای تطبیق‌های شماره ۴ تا ۹، نتیجه تطبیق بر اساس جستجو در وب معنایی بدست آمده است. در واقع، جستجویی بر روی وب معنایی انجام شده تا خصیصه‌های موجودیتی از نوع *'Patient'* بدست آید. برخی از نتایج این جستجو در جدول ۴-۱۵ ~~جدول ۴-۱۵~~ نشان داده شده است، که شناسه *URI* موجودیت‌هایی در وب معنایی که کلاسی با نام *'Patient'* را توصیف می‌کنند، بهمراه خصیصه‌های آن را مشخص می‌کند.

جدول ۴-۱۴ نتایج الگوریتم تطبیق برای مورد آزمون اول

شماره	برچسب اولیه	برچسب تطبیق‌یافته	نوع حاشیه‌نویسی	تطبیق درست
۱	CreateContact	CreatePatient	۱	✓
۲	CreateContactButtonPressed	CreatePatientButtonPressed	۱	✓
۳	ContactDateInput	PatientDateInput	۱	✓
۴	name	name	۲	✓
۵	email	patient id	۲	✓
۶	postalAddrMain	age	۲	✓
۷	postalAddrAlternative	addr	۲	✓
۸	phoneMain	Patient's Birth Time	۲	✓
۹	phoneAlternative	sex	۲	✓
۱۰	newContact	newPatient	۱	✓
۱۱	reEditingContact	reEditingPatient	۱	✓
۱۲	saveNewContact	saveNewPatient	۱	✓
۱۳	newContact	newPatient	۱	✓
۱۴	showAddressBook	ShowReceptionists	۳	×
۱۵	newContact	newPatient	۱	✓
۱۶	saveContact	savePatient	۱	✓
۱۷	searchFor*Contact*InAddressBook	searchFor*Patient*InReceptionists	۱	✓
۱۸	searchForContactIn*AddressBook*	searchForPatientIn*Receptionists*	۳	×



شکل ۴-۱۴. نمودار فعالیت ‘createPatient’ (تطبیق یافته از روی نمودار فعالیت ‘createContact’)

جدول ۴-۱۵ برخی از نتایج جستجو بر روی وب معنایی برای مفهوم 'Patient'

شناسه (URI) کلاس	برخی خصیصه‌ها
http://sw.opencyc.org/2008/06/10/concept/en/MedicalPatient	age, Primary Cause Of Death For Individual
http://www.loria.fr/~coulet/ontology/sopharm/version2.0/#SOPHARM_51000	patient id, sex, presents clinical item
http://purl.org/healthcarevocab/v1#IE.Patient	Patient Comments, Patient's Birth Time
http://veggente.berlios.de/ns/RIMOntology#Patient	veryImportantPersonCode, name, id, addr,

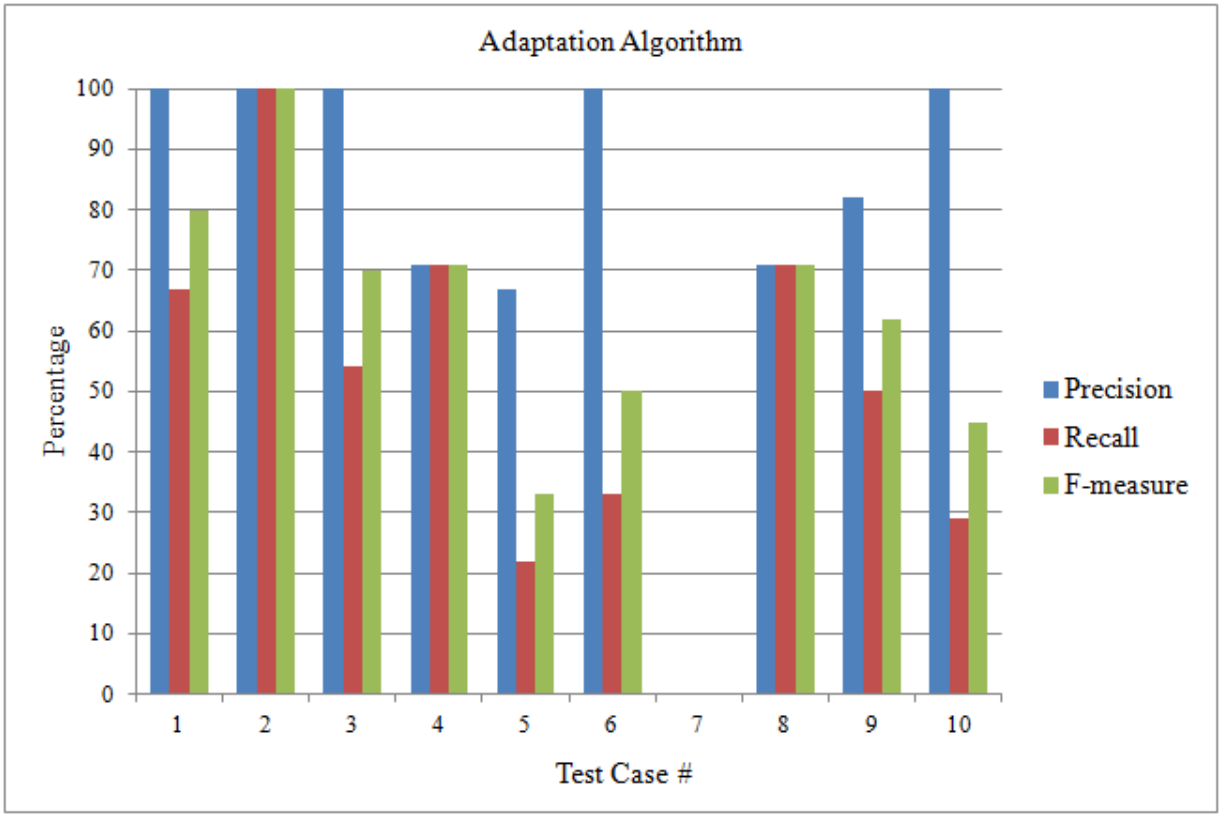
نتایج آزمایش نشان می‌دهد که همانطور که انتظار می‌رود، چالش اصلی الگوریتم تطبیق، مربوط به تطبیق حاشیه‌نویسی‌های از نوع ۲ و ۳ می‌باشد. به بیان دیگر، در همه ۱۰ مورد آزمون، همه حاشیه-نویسی‌های از نوع ۱ با موفقیت و بدرستی تطبیق داده شده‌اند.

۴-۸ نقش وب معنایی

از آنجا که الگوریتم تطبیق از وب معنایی برای تأمین اطلاعات مورد نیاز خود استفاده می‌کند، آزمایش‌هایی انجام شده است که مشخص می‌کند استفاده از وب معنایی تا چه حد بر کیفیت این الگوریتم تأثیر می‌گذارد. بدین منظور، آزمایش قسمت قبل دوباره تکرار شده است اما این بار، ماژول جستجو در وب معنایی غیرفعال شده است. به بیان دیگر، هر بار که الگوریتم تطبیق، ماژول مربوطه را برای جستجو در وب معنایی فراخوانی می‌کند، این ماژول بلافاصله مجموعه جواب تهی را به عنوان نتیجه جستجو برمی‌گرداند.

نتایج این آزمایش در جدول ۴-۱۶~~جدول ۴-۱۶~~ و شکل ۴-۱۵~~شکل ۴-۱۵~~ نشان داده شده است. تحلیل این نتایج نشان می‌دهد که متوسط *precision* از ۷۸٪ به ۷۹٪ افزایش یافته است. دلیل این امر آن است که چون ماژول جستجو در وب معنایی غیرفعال شده است، حاشیه‌نویسی‌هایی که تطبیق آن‌ها نیازمند این جستجو بوده است عملا نادیده گرفته شده و تطبیق داده نشده‌اند. در نتیجه تعداد تطبیق‌ها و به دنبال آن تعداد تطبیق‌های نادرست نیز کاهش یافته و این امر، *precision* را افزایش داده است. از سوی دیگر، چون این حاشیه‌نویسی‌ها تطبیق داده نشده‌اند، تعداد تطبیق‌های درست نیز کاهش یافته است. در واقع، مقدار متوسط *recall* از ۶۹٪ به ۵۰٪ کاهش یافته است که نرخ کاهش برابر ۲۷٪ است.

مقدار متوسط *f-measure* نیز از ۷۲٪ به ۵۸٪ کاهش یافته که نشان از نرخ کاهش ۱۹٪ است. این امر بدان معناست که غیرفعال کردن قابلیت جستجو در وب معنایی، به شکل قابل ملاحظه‌ای کیفیت الگوریتم تطبیق را کاهش می‌دهد. در نتیجه پاسخ سوال دومی که در بخش ۱-۴ مطرح شد، مثبت است و ایده استفاده از وب معنایی به عنوان یک فعال‌ساز برای الگوریتم تطبیق ارائه شده، ایده‌ای درست و امیدوارکننده است.



شکل ۴-۱۵. نتایج ارزیابی الگوریتم تطبیق بدون استفاده از قابلیت جستجو در وب معنایی

جدول ۴-۱۶ نتایج ارزیابی الگوریتم تطبیق بدون استفاده از قابلیت جستجو در وب معنایی

شماره	نام مورد کاربرد جدید	عنوان برنامه کاربردی جدید	نام مورد کاربرد مورد تطبیق	عنوان برنامه کاربردی مورد تطبیق	تعداد حاشیه‌نویسی‌ها	تعداد حاشیه‌نویسی‌های تطبیق‌یافته	تعداد تطبیق‌های درست
۱	CreatePatient	HospInfo	CreateContact	AddressBook	۱۸	۱۲	۱۲
۲	CreateContact	AddressBook	CreatePatient	HospInfo	۴	۴	۴
۳	DeleteContact	AddressBook	DeletePublication	PVS	۱۳	۷	۷
۴	DeletePublication	PVS	DeleteContact	AddressBook	۷	۷	۵
۵	BuyAlbum	SimpleMusicPortal	Buy Ticket	IMDB	۹	۳	۲
۶	Buy Ticket	IMDB	BuyAlbum	SimpleMusicPortal	۱۲	۴	۴
۷	DeleteContact	AddressBook	RemoveFriend	Philoponella	۷	۲	۰
۸	RemoveFriend	Philoponella	DeleteContact	AddressBook	۷	۷	۵
۹	Update Friend	Philoponella	UpdateContact	AddressBook	۱۸	۱۱	۹
۱۰	UpdateContact	AddressBook	Update Friend	Philoponella	۷	۲	۲

۴-۹ خلاصه

در این فصل تک تک اجزا و الگوریتم‌های روش پیشنهادی بطور جداگانه مورد ارزیابی قرار گرفته است. دقت و کارآیی هر یک از این عناصر در موفقیت کل روش پیشنهادی مؤثر است. از آنجا که حاشیه-نویسی‌های ایجاد شده توسط الگوریتم حاشیه‌نویسی اساس کار الگوریتم تطبیق هستند، هر چه الگوریتم حاشیه‌نویسی دقیق‌تر و موفق‌تر باشد، پتانسیل بیشتری برای موفقیت الگوریتم تطبیق فراهم می‌شود. خوشبختانه ارزیابی‌های صورت گرفته نشان می‌دهد الگوریتم حاشیه‌نویسی با داشتن *precision* برابر ٪۹۸ و *recall* برابر ٪۹۷ از دقت و کارایی خوبی برخوردار است. همچنین تحلیل مجموعه داده آزمایش نشان می‌دهد تقریبا ٪۹۱ کل حاشیه‌نویسی‌ها از نوع متصل هستند که ساز و کار لازم برای تطبیق آن‌ها در الگوریتم تطبیق پیشنهادی در نظر گرفته شده است. بدین ترتیب، الگوریتم حاشیه‌نویسی با داشتن دقت و کارایی خوب، شانس زیادی را برای موفق و مؤثر بودن الگوریتم تطبیق فراهم می‌کند.

در فرآیند استفاده مجدد و در مرحله پیشنهاد مورد کاربرد از معیار جدیدی برای محاسبه شباهت معنایی موارد کاربرد استفاده شده است. هرچه این معیار بهتر عمل کند و موارد کاربرد بهتری را به کاربر پیشنهاد کند کاربر سریع‌تر می‌تواند مورد کاربرد مورد نظرش را از بین موارد کاربرد موجود در مخزن پیدا کرده و در نتیجه مورد کاربرد بهتری برای عمل تطبیق انتخاب می‌شود. اگر معیار شباهت مورد کاربرد کیفیت خوبی نداشته باشد، آنگاه وظیفه کاربر در پردازش و ارزیابی موارد کاربرد موجود سنگین‌تر و پیچیده‌تر شده و احتمال انتخاب مناسب‌ترین مورد کاربرد برای عمل تطبیق کاهش می‌یابد.

خوشبختانه نتایج ارزیابی نشان می‌دهد معیار شباهت ارائه شده و الگوریتم‌های مربوط به آن کیفیت خوبی دارند. الگوریتم تشخیص رفتار، در صورت استفاده از چاشنی مناسبی که در ارزیابی‌ها انتخاب شده است، از *precision* برابر ۹۴٪ و *recall* برابر ۹۳٪ و الگوریتم تشخیص مفاهیم از *precision* برابر ۸۷٪ و *recall* برابر ۹۳٪ برخوردار است. همچنین معیار ارائه شده در صورت انتساب وزن‌های مناسب، در سطح $K=5$ دارای $P@K$ برابر ۸۲٪ می‌باشد، یعنی بطور متوسط، در بین ۵ پیشنهاد اولی که بر اساس این معیار ارائه می‌شود ۴ پیشنهاد درست وجود دارد. مقدار $P@K$ برای $K=10$ برابر ۷۲٪ می‌باشد که یعنی بطور متوسط در بین ۱۰ پیشنهاد نخست، ۷ پیشنهاد درست وجود دارد. همچنین مقدار *MRR* برای معیار ارائه شده برابر ۱ است که یعنی بطور متوسط، نخستین پیشنهاد درست در رتبه ۱ پیشنهادها ظاهر می‌شود. بدین ترتیب می‌توان گفت موارد کاربردی که در طی فرآیند استفاده مجدد به کاربر پیشنهاد می‌شوند، پیشنهادهای مناسبی هستند و در نتیجه کاربر می‌تواند با بررسی مختصر آن‌ها، مناسب‌ترین مورد را برای عمل تطبیق انتخاب نماید.

مقدار متوسط *precision* برای الگوریتم تطبیق نمودار فعالیت برابر ۷۸٪ و مقدار متوسط *recall* برابر ۶۹٪ است که مقادیر خوب و قابل قبولی هستند. اگرچه هنوز آزمایش‌های بیشتری لازم است اما نتایج آزمایش‌های صورت گرفته نشان می‌دهد در کل، روش پیشنهادی قابل استفاده و عملکرد آن قابل قبول است، البته با در نظر داشتن این نکته که اساسا هدف الگوریتم تطبیق این نبوده است که نسخه نهایی و دقیقی ایجاد کند که نیاز به هیچ تغییری نداشته باشد. بلکه هدف این بوده که کاربر با هزینه

کمی نسخه اولیه نمودار فعالیت مورد نیاز خود را بدست آورد و سپس با صرف وقت نه چندان زیادی تغییرات لازم را روی نمودار فعالیت اعمال کرده و نسخه نهایی آن را آماده کند.

مجموعه داده‌ای که برای آزمایش‌های این فصل مورد استفاده قرار گرفته است مجموعه داده بزرگی نیست و متأسفانه هیچ مجموعه داده استاندارد با حجم قابل توجه در این زمینه وجود ندارد. در صورتی که چنین مجموعه داده‌ای فراهم شود امکان آزمایش‌های دقیق‌تر و بیشتر وجود دارد، با این حال در غیاب آن، می‌توان درباره شرایطی که طی آن، روش پیشنهادی مؤثرتر است بحث نمود.

دیدگاه این رساله آن است که چنانچه روش پیشنهادی، توسط یک سازمان یا تیم نرم‌افزاری که بر توسعه خانواده خاصی از برنامه‌های کاربردی تحت وب تمرکز دارد استفاده شود، مزایای آن بیشتر و ملموس‌تر می‌شود.

به عنوان مثال، چنانچه یک تیم نرم‌افزاری بر توسعه برنامه‌های کاربردی تحت وب برای سیستم آموزشی دانشگاه تمرکز داشته باشد، به مرور زمان که محصولات نرم‌افزاری این تیم تکامل می‌یابد، مخزن معنایی مدل‌ها غنی شده و با توجه به اینکه مفاهیم موجود در آن حوزه محدود می‌باشند، با گذشت زمان، مدل این مفاهیم ایجاد شده و احتمال آنکه اطلاعات مورد نیاز برای ایجاد مدل‌های جدید، در این مخزن موجود باشد افزایش می‌یابد. همچنین با توجه به اینکه نمودارهای فعالیت موجود در مخزن، توسط همان تیم ایجاد شده‌اند، احتمال آنکه تطبیق‌های صورت گرفته توسط روش پیشنهادی، برای طراحان تیم قابل قبول باشد افزایش می‌یابد چرا که این نمودارها بر اساس نمودارهایی تطبیق یافته‌اند که توسط همان افراد ایجاد شده است.

۵- نتیجه‌گیری و کارهای آینده

در این رساله روشی برای استفاده مجدد از مدل نیازمندی‌های عملیاتی یک برنامه کاربردی تحت وب ارائه شده است که هم بازیابی مصنوعات قابل استفاده مجدد و هم استفاده از آن‌ها در تولید مصنوعات جدید را مورد پوشش قرار می‌دهد.

روش ارائه شده شامل دو مرحله اصلی است. در مرحله نخست، مخزنی ایجاد می‌شود که اطلاعات مدل‌های موجود، در قالب داده‌های معنایی در آن ذخیره شده است. این اطلاعات شامل حاشیه‌نویسی-هایی نیز می‌باشد که بر اساس تعریف و الگوریتم‌های ارائه شده در این رساله ایجاد می‌شوند. در مرحله دوم، یعنی پس از آنکه مخزن معنایی مدل‌ها آماده شد، فرآیند استفاده مجدد قابل استفاده است. طی این فرآیند، پس از دریافت توصیف کلی نیازمندی‌های عملیاتی یک برنامه کاربردی جدید در قالب نمودار مورد کاربرد *UML*، ابتدا برای هر مورد کاربرد ورودی، شبیه‌ترین موارد کاربرد موجود در مخزن معنایی به کاربر پیشنهاد شده و بعد از آن که کاربر، مورد کاربرد مناسب‌تر را انتخاب کرد، نمودار فعالیت مربوط به آن مورد کاربرد، برای مورد کاربرد ورودی تطبیق داده شده و بدین ترتیب نمودار فعالیت مورد کاربرد ورودی ایجاد می‌شود.

روش ارائه شده شامل الگوریتم‌های جدیدی نظیر الگوریتم تشخیص مفاهیم و رفتار مورد کاربرد، الگوریتم حاشیه‌نویسی نمودار فعالیت و الگوریتم تطبیق نمودار فعالیت می‌باشد. همچنین از معیار جدیدی برای محاسبه شباهت معنایی موارد کاربرد استفاده شده است.

ارزیابی‌های انجام شده نشان از دقت و کارایی خوب الگوریتم‌های ارائه شده دارد و در مجموع می‌توان نتیجه می‌گرفت روش ارائه شده روشی درست، منطقی و امیدوارکننده است و با توجه به نتایج بدست آمده از آزمایش‌ها می‌توان چنین نتیجه‌گیری کرد:

- ارائه یک روش استفاده مجدد که تنها بر اساس توصیف کلی نیازمندی‌های عملیاتی در قالب نمودار مورد کاربرد *UML* عمل کند انجام‌پذیر است. به بیان دیگر، مورد کاربرد *UML* می‌تواند به

عنوان مبنای استفاده مجدد از مدل‌ها استفاده شود. این نکته را شاید بتوان مهم‌ترین نتیجه حاصل از این پژوهش بحساب آورد.

- با استفاده از فضای داده غنی فراهم شده توسط وب معنایی می‌توان فرآیند استفاده مجدد را تقویت کرده و دانش مورد نیاز برای این فرآیند را تا حد مطلوبی تأمین و در نتیجه این فرآیند را تا میزان مناسبی خودکارسازی نمود.
- معیار ارائه شده برای اندازه‌گیری شباهت معنایی موارد کاربرد *UML* از دقت خوبی برخوردار است و با استفاده از این معیار، تشخیص مدل‌هایی که می‌توان از آن‌ها در تولید مدل‌های یک برنامه کاربردی جدید استفاده کرد با دقت خوبی قابل انجام است.
- استفاده از فناوری‌های وب معنایی، بواسطه مزایایی نظیر قابلیت استنتاج خودکار، موجب کاهش پیچیدگی و افزایش انعطاف‌پذیری در پیاده‌سازی فرآیند استفاده مجدد می‌شود.

علیرغم امیدوارکننده بودن نتایج آزمایش‌های انجام شده، رویکرد ارائه شده هنوز با چالش‌هایی مواجه است و الگوریتم‌های ارائه شده نیازمند بهبود هستند. برخی از مهمترین مواردی که به عنوان کارهای آتی در راستای این تحقیق مورد توجه می‌باشد عبارتند از:

- بهبود الگوریتم حاشیه‌نویسی و تقویت آن بمنظور مقابله با مشکلات ناشی از تطبیق متنی و همچنین ناسازگاری متنی در عبارات بکار رفته در برچسب‌های نمودارهای فعالیت. با توجه به اینکه کاربران در ایجاد نمودارهای فعالیت، برچسب‌ها را آزادانه انتخاب و مقداردهی می‌کنند و محدودیت خاصی بر عملکرد آن‌ها اعمال نمی‌شود، امکان اینکه برچسب‌ها با عناوین متناظرشان دقیقا تطبیق نداشته باشند زیاد است. مثلا ممکن است نام خصیصه یک کلاس برابر *‘creadit’* باشد اما کاربر برای ارجاع به آن از رشته *‘card’* استفاده کند. الگوریتم حاشیه‌نویسی ارائه شده در حال حاضر قادر به تشخیص چنین تناظرهایی نمی‌باشد. بهبود این الگوریتم برای آنکه چنین مواردی را بدرستی پردازش کرده و حاشیه‌نویسی‌های لازم را تشخیص دهد می‌تواند به افزایش شانس تطبیق نمودار فعالیت مربوطه منجر شود.

- بهبود الگوریتم تطبیق نمودار فعالیت بمنظور
  - پشتیبانی از الگوهای ساختاری و انجام تطبیق‌های ساختاری. الگوریتم تطبیق در حال حاضر تنها قادر به انجام تطبیق‌های متنی می‌باشد. به بیان دیگر این الگوریتم هیچ عنصری به نمودار فعالیت اضافه یا از آن کم نمی‌کند بلکه فقط برچسب‌های نمودار فعالیت را تغییر می‌دهد. به نظر می‌رسد بتوان الگوهای ساختاری تکرار شونده‌ای را در نمودارهای فعالیت مختلف شناسایی کرد و سپس در تطبیق نمودارهای فعالیت، از این الگوها نیز استفاده نمود. به عنوان مثال در نمودار فعالیت نشان داده شده در شکل ۴-۵شکل ۴-۵، شش یال خروجی از گره‌ای با برچسب *ContactDataInput* به گره‌ای با برچسب *newContact* وارد می‌شوند. هر یک از این یال‌ها معادل یکی از خصیصه‌های کلاس *Contact* می‌باشد. اگر این کلاس بجای شش خصیصه دارای ۳ خصیصه بود بجای این ۶ یال، تنها ۳ یال وجود داشت. بدین ترتیب در این مثال می‌توان یک الگو را شناسایی کرد که بر اساس آن، تعداد یال‌های خروجی از یک گره، برابر تعداد خصیصه-های یک کلاس است. اگر الگوریتم تطبیق، نسبت به این الگو حساس باشد در زمانی که می‌خواهد این نمودار فعالیت را برای مورد کاربرد دیگری که با مفاهیم دیگری سر و کار دارد استفاده کند، می‌تواند تطبیق‌های مناسب‌تری ایجاد کند، یعنی بسته به اینکه در مورد کاربرد جدید، چه مفهومی جایگزین *Contact* شده است، به تعداد خصیصه‌های آن مفهوم یال ایجاد کند. در حال حاضر، الگوریتم تطبیق فاقد آگاهی نسبت به الگوها می-باشد. یکی از جهت‌های مهم در کارهای آتی این تحقیق، بررسی و شناسایی الگوهای رایج در نمودارهای فعالیت و نحوه پشتیبانی از آن‌ها توسط الگوریتم تطبیق می‌باشد.
  - پشتیبانی از حاشیه‌نویسی‌های غیرمتصل. در حال حاضر الگوریتم تطبیق، از حاشیه-نویسی‌های غیرمتصل پشتیبانی نمی‌کند و در واقع این حاشیه‌نویسی‌ها را تطبیق نمی-

دهد. ارائه راهکاری که بتواند این مشکل را حل کند، یکی دیگر از کارهای آتی در راستای این تحقیق می‌باشد.

- بررسی و ارزیابی منابع وب معنایی بمنظور شناسایی منابعی که برای استفاده در روش ارائه شده مناسب‌تر هستند. با توجه به گستردگی منابع موجود در ابر *LOD*، تحقیق درباره اینکه کدامیک از این منابع برای استفاده در روش پیشنهادی مؤثرتر است موضوع مهمی است که تحقیق حاضر به آن نپرداخته است.
- ارائه راهکارهای جدیدی برای رفع نیازهای اطلاعاتی از بستر وب معنایی و داده‌های پیوندی. در این رساله، ایده‌هایی مکاشفه‌ای برای استفاده از وب معنایی و داده‌های پیوندی بمنظور تأمین دانش مورد نیاز الگوریتم تطبیق مطرح شده است. بهبود و تکمیل این ایده‌ها از گام‌های مهم در راستای بهبود کیفیت روش پیشنهادی می‌باشد.
- پیاده‌سازی یک نسخه عملیاتی و کاربرپسند. در حال حاضر نسخه اولیه‌ای از راهکار پیشنهادی پیاده‌سازی شده است اما این نسخه برای استفاده توسط کاربران نهایی مناسب نمی‌باشد. پیاده-سازی یک نسخه عملیاتی و کاربرپسند، یکی دیگر از فعالیت‌هایی است که در ادامه تحقیق جاری می‌توان به آن پرداخت.

٦- مراجع

1. Murugesan, S., Web Application Development: Challenges and the Role of Web Engineering. Chap. 2 in Web Engineering: Modelling and Implementing Web Applications, edited by Gustavo Rossi, Oscar Pastor, Daniel Schwabe and Luis Olsina. Springer-Verlag, 2008.
2. Kappel, G., Proll, B., Reich, S., Retschitzegger, W., "An Introduction to Web Engineering", In Web Engineering - Systematic Development of Web Applications. Kappel, G., Pröll, B., Reich, S., and Retschitzegger W., Eds. Wiley, 2006.
3. Valderas, P., Pelechano, V., "A Survey of Requirements Specification in Model-Driven Development of Web Applications", ACM Transactions on the Web (ACM) 5, no. 2, 1-51, 2011.
4. Atkinson, C., Kuhne, T., "Model-Driven Development: A Metamodeling Foundation", IEEE Software 20, no. 5, 36-41 (2003)
5. Kusel, A., Schonbock, J., Wimmer, M., Kappel, G., Retschitzegger, W., Wchwinger, W., "Reuse in Model-to-model transformation languages: are we there yet?", Journal of Software & Systems Modeling, vol. 12, issue 2, 2013.
6. McCarey, F., Cinneide, M.O., Kushmerick, N., "Knowledge Reuse for Software Reuse", Journal of Web Intelligence and Agent Systems, vol. 6, issue 1, pp. 59-81, 2008.
7. Sen, A., "The Role of Opportunism in the Software Design Reuse Process", IEEE Transactions on Software Engineering 23, no. 7, 418-436, 1997.
8. Prasad, A., Park, E.K., "Reuse System: An Artificial Intelligence-based Approach", Journal of Systems and Software, vol. 27, 207-221, 1994.
9. Alnusair, A., Zhao, T., "Retrieving Reusable Software Components Using Enhanced Representation of Domain Knowledge. Recent Trends in Information Reuse and Integration", Lecture Notes in Computer Science (LNCS), 363-379, 2012.
10. Robles, K., Fraga, A., Morato, J., Llorens, J., "Towards an Ontology-based Retrieval of UML Class Diagrams. Information and Software Technology", vol. 54, issue 1, 72-86, 2012.
11. Alnusair, A., Zhao, T., "Component Search and Reuse: An Ontology-based Approach", IEEE International Conference on Information Reuse and Integration (IRI), 258-261, 2010.
12. Bislimovska, B., Bozzon, A., Brambilla, M., Fraternali, P., "Graph-Based Search over Web Application Model Repositories", the 11th International Conference on Web Engineering (ICWE). Paphos, Cyprus, 2011.

13. Lucredio, D., Fortes, R. P.M., Whittle, J., "Moogle: A Model Search Engine", The 11th International Conference on Model Driven Engineering Languages and Systems (MoDELS '08). Springer-Verlag, 296-310, 2008.
14. Keivanloo, I., Roostapour, L., Schugerl, P., Rilling, J., "Semantic Web-based Source Code Search", the 6th International Workshop on Semantic Web Enabled Software Engineering (SWESE 2010). San Francisco, CA, 2010.
15. Cai, S., Zou, Y., Wang, L., Xie, B., Shao, W., "A Semi-Supervised Approach for Component Recommendation Based on Citations", Proceedings of the 12th International conference on software reuse (ICSR'11), pp. 78-86, 2011.
16. Hartig, O., Kost, M., Freytag, J.C., "Automatic Component Selection with Semantic Technologies", the 4th International Workshop on Semantic Web Enabled Software Engineering (SWESE). Karlsruhe, 2008.
17. Bajracharya, S., Ossher, J., et al., "Sourcerer: An Internet-Scale Software Repository", in Proceedings of the 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation, pp. 1-4, IEEE Computer Society, 2009.
18. Takuya, W., Masuhara, H., "A Spontaneous Code Recommendation Tool Based on Associative Search", in Proceedings of the 3rd International Workshop on Search-driven Development: Users, Infrastructure, Tools and Evaluation, SUITE'11, pp.17-20, New York, NY, USA, 2011.
19. Grechanik, M., Fu, C., Xie, Q.,McMillan, C.,Poshyvanyk, D., Cumby, C., "Exemplar: EXEcutable exeMPLes Archiver", in Proceedings of the 32th ACM/IEEE International Conference on Software Engineering- vol. 2, pp. 259-262, ACM, 2010.
20. Tappolet, J., Kiefer, C., Bernstein, A.: Semantic Web Enabled Software Analysis. Journal of Web Semantics 8, 225-240 (2010)
21. Reifer, D.J., "Web Development: Estimating Quick-to-market Software", IEEE Software, November/December, pp. 57-64, 2000.
22. Offutt, J., "Quality Attributes of Web Software Applications", IEEE Software, March/April, 19(2), pp. 25-32, 2002.
23. Standing, C., "Methodologies for Developing Web Applications", Journal of Information and Software Technology, 44(3), pp. 151-160, 2002.
24. Ginige, A., "Web Engineering: Managing the Complexity of Web Systems Development", In Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, 2002.
25. Deshpande, Y., Murugesan, S., Ginige, A., Hansen, S., Schwabe, D., Gaedke, M., White, B., "Web Engineering", Journal of Web Engineering, October, 1(1), 3-17, 2002.

26. Taylor, M.J., Mc William, J., Forsyth, H., Wade, S., "Methodologies and Web Site Development: a Survey of Practice", Journal of Information and Software Technology, 44(6), pp. 381-391, 2002.
27. Lee, S.C., Shirani, A.I., "A Component Based Methodology for Web Application Development", Journal of Systems and Software, 71(1-2), pp. 177-187, 2004.
28. Murugesan, S.: Web Engineering: A New Discipline for Development of Web-based Systems. First ICSE Workshop on Web Engineering. Los Angeles, 1-9 (1999)
29. Ginige, A., Murugesan, S., "Web Engineering: an Introduction", IEEE Multimedia, January/March, 8(1), pp. 14-18, 2001.
30. Murugesan, S., Deshpande, Y., "Web Engineering, Managing Diversity and Complexity of Web Application Development", Lecture Notes in Computer Science 2016, Springer Verlag, Heidelberg, 2001.
31. Gellersen, H., Wicke, R., Gaedke, M., "WebComposition: an Object-Oriented Support System for the Web Engineering Lifecycle", Journal of Computer Networks and ISDN Systems, September, 29(8-13), pp. 865-1553, 1997. Also in Proceedings of the Sixth International World Wide Web Conference, pp. 429-1437, 1996.
32. Rossi, G., Schwabe, D., "Object-Oriented Web Applications Modeling", In Information Modelling in the New Millennium, IGI Publishing, USA, ISBN: 1-878289-77-2, pp. 463-484, 2001.
33. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. Comput. Netw. 33, 137-157 (2000)
34. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: UML-Based Web Engineering: An Approach Based on Standards. Chap. 7 in Web Engineering: Modelling and Implementing Web Applications, edited by L. Olsina, O. Pastor, G. Rossi and D. Schwabe, 157-191. Berlin: Springer, (2008)
35. De Troyer, O., Leune, C.: WSDM: A User-Centered Design Method for Web Sites. the 7th International World Wide Web Conference . Elsevier, 85-94 (1998)
36. Escalona, M. J., Aragon, G.: NDT: A Model Driven Approach for Web Requirements. IEEE Transactions on Software Engineering 34, no. 3, 377-390 (2008)
37. Isakowitz, T., Kamis, A., Koufaris, M., "Reconciling Top-Down and Bottom-Up Design Approaches in RMM, Journal of Data Base, 29(4), pp. 58-67, 1998.
38. Houben, G., Frasincar, F., Barna, P., Vdovjak, R., "Modelling User Input and hypermedia Dynamics in Hera", in Proceedings of the 4th International

Conference on Web Engineering (ICWE 2004), Springer LNCS 3140, Munich, pp. 60-73, 2004.
39. Garzotto, F., Mainetti, L., Paolini, P., "Hypermedia Design, Analysis, and Evaluation Issues", Communications of the ACM, 38(8), August, pp. 74-86, 1995.
40. Baresi, L., Garzotto, F., Paolini, P., "Extending UML for Modeling Web Applications", in Proceedings of the 34th Hawaii International Conference on Systems Sciences (HICSS 2001), Maui, HI, January, 2001.
41. Fraternali, P., Paolini, P., "A Conceptual Model and a Tool Environment for Developing More Scalable and Dynamic Web Applications", in Proceedings of the Conference on Extended Database Technology (EDBT'98), Valencia, Spain, March, 1998.
42. De Troyer, O., Decruyenaere, T., "Conceptual Modeling of Web Sites for End-Users", WWW Journal, 3(1), Baltzer Science Publishers, 2000 pp. 27-42.
43. Schwabe, D., Rossi, G., "An Object Oriented Approach to Web-based Application Design", Theory and Practice of Object Systems (TAPOS), 4(4), 1998.
44. Koch, N., Kraus, A., "Twoards a Common Metamodel for the Development of Web Applications", in Proceedings of the 3rd International Conference of Web Engineering (ICWE 2003), Oviedo, Spain, July, 2003.
45. Pastor, O., Pelechano, V., Fons, J., Abrahao, S., "Conceptual Modeling of Web Applications: the OOWS Approach", In: Web Engineering – Theory and Practice of Metrics and Measurement for Web Development, Mendes, E., Mosley, N., (Eds.), Springer-Verlag, 2005.
46. Gomez, J., Cachero, C., "OO-H Method: Extending UML to Model Web Interfaces", in: Information Modeling for Internet Applications, van Bommel, P., (ed.), Idea Grouop Inc., 2003.
47. Schwinger, W., Koch, N., "Modeling Web Applications", in Kappel, G., Proll, B., Reich, S., Retschitzegger, W. (eds.), Web Engineering, the Disiplince of Systematic Development of Web Applications, chapter 3, pp. 39-65, 2006.
48. Frakes, W. B., Kang, K.: Software Reuse Research: Status and Future. IEEE Transactions on Software Engineering 31, no. 7, 529-536, 2005.
49. Mili, H., Mili, F., Mili, A.: Reusing Software: Issues and Research Directions. IEEE Transactions on Software Engineering 22, no. 6, 1995.
50. McIlroy, M.D.: Mass Produced Software Components. NATO Software Engineering Conference. Garmisch: Germany, 1968.
51. Mohagheghi, P., Conradi, R.: Quality, Productivity, and Economics Benefits of Software Reuse: A Review of Industrial Studies. Empirical Software Engineering 12, 471-516, 2007.

52. Frakes, W. B., Kang, K.: Software Reuse Research: Status and Future. IEEE Transactions on Software Engineering 31, no. 7, 529-536 (2005)
53. Mili, H., Mili, F., Mili, A.: Reusing Software: Issues and Research Directions. IEEE Transactions on Software Engineering 22, no. 6 (1995)
54. Mohagheghi, P., Conradi, R.: Quality, Productivity, and Economics Benefits of Software Reuse: A Review of Industrial Studies. Empirical Software Engineering 12, 471-516 (2007)
55. Card, D., Comer, E.: Why Do So Many Reuse Programs Fail?. IEEE Software, Vol. 11, No. 5, 114-115 (1994)
56. Morisio, M., Ezran, M., Tully, C.: Success and Failure Factors in Software Reuse. IEEE Transactions on Software Engineering 28, no. 4, 340-357 (2002)
57. Catal, C.: Barriers to the Adoption of Software Product Line Engineering, ACM SIGSOFT Software Engineering Notes, Vol. 34, No. 6 (2009)
58. Sherif, K., Vinze, A.: Barriers to Adoption of Software Reuse, a Qualitative Study, Journal of Information and Management, vol. 41, no. 2, 159-175 (2003)
59. Jalender, B., Gowtham, N., Kumar, K.P., Murahari, K., Sampath, K.: Technical Impediments to Software Reuse, International Journal of Engineering Science and Technology, Vol. 2, no. 11, 6136-6139 (2010)
60. Malan, R., Wentzel, K.: Economics of Software Reuse Revisited, in Proceedings of the 3rd Irvine Software Symposium, University of California, Irvine, 109-121 (1993)
61. Lim, W.C.: Effects of Reuse on Quality, Productivity, and Economics, IEEE Software, vol. 11, no. 5, 23-30 (1994)
62. Milli, A., Fowler, S.C., Gottumkkala, R., Zhang, L.: An Integrated Cost Model for Software Reuse, In proceedings of the 22nd International Conference on Software Engineering, 157-166 (2000)
63. Postmus, D., Meijler, T.D.: Aligning the Economic Modeling of Software Reuse with Reuse Practices, journal of Information and Software Technology, vol. 50, 753-762 (2008)
64. Mens, T., Lucas, C., Steyaert, P.: Supporting Disciplined Reuse and Evolution of UML Models, Lecture Notes in Computer Science, vol. 1618, 378-392 (1999)
65. Lucas, C.: Documenting Reuse and Evolution with Reuse contracts, Ph.D. Dissertation, Vrije Universiteit Brussel (1997)
66. Steyaert, P., Lucas, C., Mens, K., D'Hondt, T.: Reuse Contracts-Managing the Evolution of Reusable Assets, in Proceedings of OOPSLA '96, SIGPLAN Notices, vol. 31, no. 10, 268-286 (1996)
67. Anguswamy, R.: A Study of Factors Affecting the Design and Use of Reusable Components, International Doctoral Symposium on Empirical Software Engineering (IDoESE'12), Sep. 21, 2012, Lund, Sweden, (2012)

68. Anguswamy, R., Frakes, W.B.: A Study of Reusability, Complexity, and Reuse Design Principles, The 6th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM'12. Lund, Sweden. (2012)
69. Lemos, O.A.L., Bajracharya, S.K., Ossher, J., Morla, R.S., Masiero, P.C., Baldi, P., Lopes, C.V.: CodeGenie: Using Test-Cases to Search and Reuse Source Code, in Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering, 525-526 (2007)
70. Hamid, A.: A Source Code Search Engine for Keyword Based Structural Relationship Search, Thesis, the University of Texas at Arlinkgton (2013)
71. Bajracharya, S., Ossher, J., Lopes, C.: Sourcerer: An Infrastructure for Large-Scale Collection and Analysis of Open-Source Code, journal of Science of Computer Programming, vol. 79, no. 1, 241-259 (2014)
72. McMillan, C., Grechanik, M., Poshyvanyk, D., Fu, C., Xie, Q.: Exemplar: A Source Code Search Engine for Finding Highly Relevant Applications, IEEE Transactions on Software Engineering, vol. 38, no. 5, 1069-1087 (2012)
73. McMillan, C., Hariri, N., Poshyvanyk, D., Cleland-Huang, J.: Recommending Source Code for Use in Rapid Software Prototypes, in Proceedings of the International Conference of Software Engineering (ICSE), 848-858 (2012)
74. Holmes, R., Walker, R.J., Murphy, G.C.: Approximate Structural Context Matching: An Approach to Recommend Relevant Examples, IEEE Transactions on Software Engineering, vol. 32, no. 12, 952-970 (2006)
75. Heinemann, L., Hummel, B., "Recommending API Methods based on Identifier Contexts", in Proceedings of the 3rd International Workshop on Search-driven Development: Users, Infrastructure, Tools and Evaluation, SUITE'11, pp. 1-4, New York, NY, USA, 2011.
76. Cheng, Z., Budgen, D.: What Do We Know About the Effectiveness of Software Design Patterns? IEEE Transactions on Software Engineering, vol. 38, no. 5, 1213-1231 (2012)
77. Jones, C.: Software Return on Investment Preliminary Analysis, Software Productivity Research, Inc.
78. Goldberg, A., Rubin, K.S., "Succeeding with Objects: Decision Frameworks for Project Management", Addison-Wesley, 1995.
79. Butler, G,. Li, L., Tjandra, I.A., "Reusable Object-Oriented Design", Department of Computer Science, Concordia University, Montreal, Quebec, 1999.
80. Sen, A.: The Role of Opportunism in the Software Design Reuse Process. IEEE Transactions on Software Engineering 23, no. 7, 418-436 (1997)

81. Barros, F.: Increasing Software Quality through Design Reuse, In proceedings of the 7th International Conference on the Quality of Information and Communications Technology, 236-241 (2010)
82. Smialek, M., Kalnins, A., Kalnina, E., Ambroziewicz, A.: Comprehensive System for Systematic Case-Driven Software Reuse. SOFSEM 2010, 697-708 (2010)
83. Karlsson, E.A.: Software Reuse- A Holistic Approach, Wiley (1995)
84. Falessi, D., Cantone, G., Kazman, R., Kruchten, P.: Decision-making Techniques for Software Architecture Design: a Comparative Survey, ACM Computing Surveys, vol. 43, no. 4, 1-28 (2011)
85. Ali, F.M., Du, W.: Toward Reuse of Object-Oriented Software Design Models. Information and Software Technology 46, no. 15, 499-517 (2004)
86. Robles, K., Fraga, A., Morato, J., Llorens, J.: Towards an Ontology-based Retrieval of UML Class Diagrams. Information and Software Technology, vol. 54, issue 1, 72-86 (2012)
87. Szyperski, C., "Component Software: Beyond Object Oriented Programming", ACM Press and Addison Wesley, New York, USA, 2002.
88. Heineman, G.T., Councill, W.T., eds.: Component-Based Software Engineering: Putting the Pieces Together, Addison-Wesley Longman Publishing Co., Inc. (2001).
89. Bachmann, F., et al., "Technical Concepts of Component-based Software Engineering", technical report, CMU/SEI-2999-TR-008, ESC-TR-2000-007, 2000.
90. Chroust, G., "Motivation in Component-Based Software Development", In: Ghaoui, C. (ed.) Chroust, G.: Motivation in component-based software development. Idea Group Reference, Hershey (2006).
91. Lau, K.K., Wang, Z., "Software Component Models", IEEE Transactions on Software Engineering, 33(10), pp. 709-724, 2007.
92. Land, R., Sundmark, D., Luders, F., Krasteva, I., Causevic, A., "Reuse with Software Components- A Survey of Industrial State of Practice", Proceedings of the 11th International Conference on Software Reuse: Formal Foundations of Reuse and Domain Engineering, ICSR '09, pp. 150-159, 2009.
93. Heinemann, L., Deissenboeck, F., Cleirscher, M., Hummel, B., Irlbeck, M., "On the Extent and Nature of Software Reuse in Open Source Java Projects", Proceedings of the 12th international conference on software reuse, ICSR'11, pp. 207-222, 2011.
94. Vitharana, P., "Risks and Challenges of Comopnent-based Software Development", Communications of the ACM, 46(8), pp. 67-72, 2003.
95. Haghpanah, N., Moaven, S., Habibi, J., Kargar, M., Yeganeh, S.H., "Aproximation Algorithms for Software Component Selection Problem",

In: Proceedings of the 14th Asia-Pacific Software Engineering Conference (ASPEC), pp. 159-166, 2007.
96. Carlson, S.E., "Genetic Algorithm Attributes for Component Selection", Research in Engineering Design 8(1), pp. 33-51, 1996.
97. Yao, H., Etzkorn, L.H., Virani, S.: Automated Classification and Retrieval of Reusable Software Component. Journal of the American Society for Information Science and Technology 59, no. 4, 613-627 (2008)
98. Butler, G., Li, L., Tjandra., L.A.: Reusable Object-Oriented Design. Department of Computer Science, Concordia University, Montreal, 19-21 (1999)
99. Gamma, E., Helm, R., Johnson, R., Vlissides, J., "Design Patterns, Elements of Reusable Object-Oriented Software", Addison Wesley, 1995.
100. Tsantalis, N., Chatzigeorgiou, A., Stephanides, G., Halkidis, S.T., "Design Pattern Detection Using Similarity Scoring", IEEE Transactions on Software Engineering 32 (11) (2006) 896-909.
101. Paydar, S., and Kahani, M.: A Semantic Web Based Approach for Design Pattern Detection from Source Code. The International Conference on Computer and Knowledge Engineering (ICCKE 2012). Mashhad, Iran (2012)
102. Kampffmeyer, H., Zschaler, S., "Finding the Pattern You Need: The Design Pattern Intent Ontology", in MoDELS, ser. Lecture Notes in Computer Science, G. Engels et al. , Eds. ,vol. 4735. Springer, 2007, pp. 211-225.
103. Dietrich, J., Elgar, C., "An Ontology Based Representation of Software Design Patterns", In Toufik Taibi (editor): Design Pattern Formalization Techniques. Idea Group Inc., 2007. ISBN: 978-1-59904-219-0.
104. Sabatucci, L., Cossentino, M., Susi, A., "Introducing Motivations in Design Pattern Representation", Proceedings of the 11th International Conference on Software Reuse: Formal Foundations of Reuse and Domain Engineering, ICSR '09, pp. 201-210, 2009.
105. Nowick, E., Eskridge, K.M., Travnicek, D.A., Chen, X., Li, J.: A Model Search Engine Based on cluster Analysis of Search Terms. Library Philosophy and Practice 7, no. 2 (2005)
106. Bozzon, A., Brambilla, M., Fraternali, P.: Searching Repositories of Web Application Models. Lecture Notes in Computer Science, vol. 6189, 1-15 (2010)
107. Kling, W., Jouault, F., Wagelaar, D., Brambilla, M., Cabot, J.: MoScript: A DSL for querying and manipulating model repositories. Lecture Notes in Computer Science, 180-200 (2011)
108. Nowick, E., Eskridge, K.M., Travnicek, D.A., Chen, X., Li, J.: A Model Search Engine Based on cluster Analysis of Search Terms. Library Philosophy and Practice 7, no. 2 (2005)

109. Zhuge, H.: A Process Matching Approach for Flexible Workflow Process Reuse. Information and Software Technology 44, no. 8, 445-450 (2002)
110. Zhuge, H.: A Process Matching Approach for Flexible Workflow Process Reuse. Information and Software Technology 44, no. 8, 445-450 (2002)
111. Robinson, W.N., Woo, H.G., "Finding Reusable UML Sequence Diagrams Automatically", IEEE Software 21(5), pp. 60-67, 2004.
112. Elias, M., Johannesson, P., "A Survey of Process Model Reuse Repositories", Proceedings of the 6th International Conference of Information Systems, Technology and Management (ICISTM 2012), Grenoble, France, 2012.
113. Markovic, I., Costa Pereira, A., "Towards a Formal Framework for Reuse in Business Process Modeling", Proceedings of the 2007 international conference on Business process management (BPM'07), pp. 484-495, 2007.
114. Bildhauer, D., Horn, T,. Ebert, J., "Similarity-driven Software Reuse", in Proceedings of the 2009 ICSE Worksop on Comparison and Versioning of Software Models, CVSM '09, pp. 31-36, Washington, DC, USA, 2009, IEEE Computer Society.
115. Smialek, M., Kalnins, A., Kalnina, E., Ambroziewicz, A., Straszak, T., Wolter, K., "Comprehensive System for Systematic Case-Driven Software Reuse", in: van Leeuwen, J., Muscholl, A., Peleg, D., Pokorny, J., Rumpe, B. (eds.) SOFSEM 2010. LNCS, vol. 5901, pp. 697-708, Springer, Heidelberg, 2010.
116. Jacobson, I., Griss, M., Jonsson, P., "Software Reuse: Architecture Process and Organization for Business Success", ACM Press, New York, 1997.
117. Jacobson, I.: Use Cases - Yesterday, Today, and Tomorrow. Software and Systems Modeling 3, 210-220 (2004)
118. Bolk, M.C., Cybulski, J.L., "Reusing UML specifications in a constrained application domain," in Proceedings of the 5th Asia Pacific Software Engineering Conference (ASPEC'98), 1998, pp. 196-202.
119. Gomes, P., Gandola, P., Cordeiro, J.: Helping Software Engineers Reusing UML Class Diagrams, in Proceedings of the 7th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development, 449-462 (2007)
120. Salami, H.O., Ahmed, M.A.: UML Artifacts Reuse: State of the Art, the International Journal of Soft Computing and Software Engineering (JSCSE), vol. 3, no. 3, 115-122 (2013)
121. Robinson, W.N., Woo, H.G.: Finding Reusable UML Sequence Diagrams Automatically, IEEE Software, vol. 21, 60-67 (2004)

122. Park, W.J., Bae, D.H.: A Two-Stage Framework for UML Specification Matching, journal of Information and Software Technology, vol. 53, 230-244 (2010)
123. Salami, H.O., Ahmed, M.A.: A Framework for Class Diagram Retrieval Using Genetic Algorithm, in proceedings of the 24th International Conference on Software Engineering and Knowledge Engineering (SEKE 2012), 737-740 (2012)
124. Bonilla-Morales, B., Crespo, S., Clunie, C.: Reuse of Use Cases Diagrams: An Approach based on Ontologies and Semantic Web Technologies, International Journal of Computer Science Issues, vol. 9, issue 1, no. 2, 24-29 (2012)
125. Gomes, P., Pereira, F.C., Paiva, P., Seco, N., Carreiro, P., Ferreira, J.L., Bento, C.: Case Retrieval of Software Designs Using WordNet, in European Conference on Artificial Intelligence (ECAI 02), 245-249 (2002)
126. Gomes, P., Pereira, F.C., Paiva, P., Seco, N., Carreiro, P., Ferreira, J.L., Bento, C.:Using Wordnet for Case-Based Retrieval of UML Models. AI Communications 17, no. 1, 13-23 (2004)
127. Bloc, M.C., Cybulski, J.L.: Reusing UML Specifications in a Constrained Application Domain, in Proceedings of the 5th Asia Pacific Software Engineering Conference (APSEC) (1998)
128. Alspaugh, T.A., Ant, A.I., Barnes, T., Mott, B.W.: An Integrated Scenario Management Strategy, in Proceedings of the 4th IEEE International Symposium on Requirements Engineering, 142-149 (1999)
129. Jacobson, I.: Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley (1992)
130. Jacobson, I.: Use Cases - Yesterday, Today, and Tomorrow. Software and Systems Modeling 3, 210-220 (2004)
131. Srisura, B., Daengdej, J.: Retrieving Use Case Diagram with Case-Based Reasoning Approach, Journal of Theoretical and Applied Information Technology, vol. 19, no. 2, 68-78 (2010)
132. Saeki, M., Reusing use case descriptions for requirements specification: towards use case patterns, in Proceedings of the 6th Asia Pacific Software Engineering Conference (APSEC), 309-316 (1999)
133. Udomchaiporn, A., Prompoon, N., Kanongchaiyos, P., Software Requirements Retrieval Using Use Case Terms and Structure Similarity Computation, in Proceedings of the 13th Asia Pacific Software Engineering Conference, APSEC, page 113-120 (2006)
134. Durao, F.A., Vanderlei, T.A., Almeida, E.S., Meira, S.R.L.: Applying a Semantic Layer in a Source Code Search Tool. the 2008 ACM Symposium on Applied Computing (SAC '08). New York, 1151-1157 (2008)
135. Clements, P., Northrop, L.M., "Software Product Lines: Practices and Patterns", 6th edn. Addison-Wesley, Reading (2007).

136. Pohl, K., Bockle, G., van der Linden, F., “Software Product Line Engineering: Foundations, Principles, and Techniques”, Springer, Berlin, 2005.
137. Kang, K., Sugumaran, V., Park, S., “Applied Software Product-Line Engineering”, Auerbach, Boca Raton, 2009.
138. Jha, M., O’Brien, L., “Identifying Issues and Concerns in Software Reuse in Software Product Lines”, In proceeding of: Formal Foundations of Reuse and Domain Engineering, 11th International Conference on Software Reuse, ICSR 2009, Falls Church, VA, USA, September 27-30, 2009.
139. Elias, M., Johannesson, P.: A Survey of Process Model Reuse Repositories, in Proceedings of ICISTM 2012, pp. 64-76 (2012)
140. Frakes, W., Terry, C.: Software Reuse: Metrics and Models, ACM Computing Surveys, vol. 28, no. 2, 415-435 (1996)
141. Koch, N., Kozuruba, S.: Requirements Models as First Class Entities in Model-Driven Web Engineering. 3rd Workhop on The Web and Requirements Engineering at ICWE 2012 (2012)
142. Monden, A., Nakae, D., Kamiya, T., Sato, S., Matsumoto, K.: Software Quality Analysis by Code Clones in Industrial Legacy Software, in Proceedings of the 8th International Symposium on Software Metrics (2002)
143. Deissenboeck, F., Hummel, B., Juergens, E., Schatz, B., Wagner, S., Girard, J.F., Teuchert, S.: Clone Detection in Automotive Model-Based Development, ICSE’08, 603-612 (2008)
144. Kelte, U., Wehren, J., Niere, J.: A Generic Difference Algorithm for UML Models, In Proceedings of the Software Engineering Conference 2005, Essen, Germany, 105–116 (2005)
145. Selonen, P.: A Review of UML Model Comparison Approaches, in Proceedings of Nordic Workshop on Model Driven Engineering, Ronneby, Sweden, August 27-29 (2007)
146. Girschick, M.: Difference Detection and Visualization in UML Class Diagrams, Technical Report TUD-CS-2006-5, (2206)
147. Bildhauer, D., Horn, T., Ebert, J.: Similarity-Driven Software Reuse, ICSE’09 Workshop (2009)
148. Li, Y., McLean, D., Bandar, Z.A., O’Shea, J.D., Crockett, K.: Sentence Similarity based on Semantic Nets and Corpus Statistics, IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 8, 1138-1150 (2006)
149. Yamamoto, T., Matsushita, M., Kamiya, T., Inoue, K.: Measuring Similarity of Large Software Systems Based on Source Code Correspondence, In: Bomarius, F., Komi-Sirvio, S. (eds.), PROFES 2005, LNCS 3547, 530-544 (2005)

150. Yamamoto, T., Matsushita, M., Kamiya, T., Inoue, K.: Similarity of Software System and Its Measurement Took SMMT, Systems and Computers in Japan, vol. 38, no. 6, 91-99 (2007)
151. Walenstein, A., El-Ramly, M., Cordy, J.R., Evans, W.S., Mahdavi, K., Pizka, M., Rama-lingam, G., von Gudenberg, J.W., Similarity in programs. In R. Koschke, E. Merlo, and A. Walenstein, editors, Duplication, Redundancy, and Similarity in Software, number 06301 in Dagstuhl Seminar Proceedings. IBFI (2007)
152. Stephan, M., Cordy, J.R.: A Survey of Methods and Applications of Model Comparison, Technical Report 2011-582 Rev. 3, School of Computing, Queen's University, Ontario, Canada (2012)
153. Woo, H.G., Robinson, W.N.: A Light-Weight Approach to the Reuse of Use-Cases Specifications, presented at Southern Association for Information Systems 2002 Conference, Savannah, GA (2002)
154. Bin, S., Liying, F., Jianzhuo, Y., Pu, W., Zhongcheng, Z.: Ontology-based Measure of Semantic Similarity between Concepts, World Congress on software Engineering (WCSE), 109-112 (2009)
155. Miller, G.: Wordnet: A Lexical Database for English. Commun. ACM 38, 39-41 (1995)
156. Trakarnviroj, A., Prompoon, N.: A Storage and Retrieval of Requirement Model and Analysis Model for Software Product Line, in Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS 2012), March 14-16, Hong Kong (2012)
157. Ilieva, M.G., Boley, H.: Representing Textual Requirements as Graphical Natural Language for UML Diagram Generation. Software Engineering and Knowledge Engineering (SEKE), 478-483 (2008)
158. Korner, S.J., Gelhausen, T.: Improving Automatic Model Creation Using Ontologies. Software Engineering and Knowledge Engineering (SEKE) (2008)
159. Samarasinghe, N., Some, S.S.: Generating a Domain Model from a Use Case Model. ISCA. (2005)
160. Yue, T., Briand, L.C., Labiche, Y.: Facilitating the Transition from Use Case Models to Analysis Models: Approach and Experiments. Transactions on Software Engineering and Methodology (TOSEM) 22(1), (2013)
161. Dobing, B., Parsons, J.: How UML is Used. Communications of the ACM 49, 109-113 (2006)
162. Klimek, R., Szwed, P.: Formal Analysis of Use Case Diagrams. Computer Science 11, 115-131 (2010)
163. Bendraou, R., Jezequel, J.M., Gervais, M.P., Blanc, X.: A Comparison of Six UML-Based Languages for Software Process Modeling. IEEE Transactions on Software Engineering 36, no. 5, 662-675 (2010)

164. Lauesen, S., Kuhail, M.A.: Task Descriptions versus Use Cases. Requirement Engineering (Springer) 17, no. 1, 3-18 (2012)
165. Anda, B., Sjoberg, D. I.K.: Investigating the Role of Use Cases in the Construction of Class Diagrams. Empirical Software Engineering 10, 285-309 (2005)
166. Liang, Y.: From Use Cases to Classes: A Way of Building Object Model with UML. Information and Software Technology 45, 83-93 (2003)
167. Yue, T., Briand, L.C., Labiche, Y.: Automatically Deriving a UML Analysis Model from a Use Case Model. Technical Report 2010-15, Simula Research Laboratory (2010)
168. Yue, T., Briand, L.C., Labiche, Y.: An Automated Approach to Transform Use Cases into Activity Diagrams. Lecture Notes in Computer Science, 337-353 (2010)
169. Gronmo, R., Moller-Pedersen, B.: From UML2 Sequence Diagrams to State Machines by Graph Transformation. Journal of Object Technology 10, 1-22 (2011)
170. Some, S.S.: An Approach for the Synthesize of State Transition Graphs from Use Cases. International Conference on Software Engineering Research and Practice (SERP'03), 456-462 (2003)
171. Yue, T., Briand, L.C., Labiche, Y.: A Systematic Review of Transformation Approaches Between User Requirements and Analysis models. Requirements Engineering 16, no. 2, 75-99 (2011)
172. Mellor, S. J., Clark, A. N., Futagami, T.: Model-Driven Development. IEEE Software 20, no. 5, 14-18 (2003)
173. Selic, B.: The Pragmatics of Model-Driven Development. IEEE Software 20, no. 5, 19-25 (2003)
174. Koch, N., Knapp, A., Kozuruba, S.: Assessment of Effort Reduction due to Model-to-Model Transformations in the Web Domain. Vol. 7387, in Web Engineering, 215-222 (2012)
175. Vilain, P., Schwabe, D., Sieckenius, C.: Use Cases and Scenarios in the Conceptual Design of Web Application. Technical Report MCC 12/00, Departamento de Informatica, PUC-Rio, Rio de Janeiro, Brasil, (2000)
176. Kang, S., Kim, H., Baik, J., Choi, H., Keum, C.: Transformation Rules for Synthesis of UML Activity Diagram from Scenario-based Specification. 34th Annual IEEE Computer Software and Applications Conference. 431-436 (2010)
177. Antoniou, G., Harmelen, F.V.: Web Ontology Language: OWL. In Handbook of Ontologies, edited by S. Staab and R. Studer, 91-110 (2009)
178. Calero, C., Ruiz, F., Piattini, M.: Ontologies for Software Engineering and Software Technology. (2006)
179. Bauer, B., Roser, S.: Semantic-Enabled Software Engineering and Development. GI Jahrestagung (2), 293-296 (2006)

180. Bucci, G., Sandrucci, V., Vicario, E., "Ontology-Driven Enterprise Application Integration", Proceedings of the 22nd International Conference on Software Engineering & Knowledge Engineering (SEKE'2010), Redwood City, San Francisco Bay, CA, USA, 2010.
181. Izza, S., Vincent, L., Burlat, P., "A Unified Framework for Application Integration – an Ontology-Driven Service-Oriented Approach", In proceeding of: ICEIS 2005, Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005.
182. Nasser, V.H., Du, W., MacIsaac, D., "An Ontology-based Software Test Generation Framework", Proceedings of the 22nd International Conference on Software Engineering & Knowledge Engineering (SEKE'2010), Redwood City, San Francisco Bay, CA, USA, 2010.
183. Nasser, V.H., Du, W., MaccIsaac, D., "Knowledge based Software Test Generation", Proceedings of the 21st International Conference on Software Engineering & Knowledge Engineering (SEKE'2009), 2009.
184. Moser, T., Durr, G., Biffl, S., "Ontology-Based Test Case Generation For Simulating Complex Production Automation Systems", Proceedings of the 22nd International Conference on Software Engineering & Knowledge Engineering (SEKE'2010), Redwood City, San Francisco Bay, CA, USA, 2010.
185. Nguyen, C.D., Perini, A., Tonella, P., "Ontology-based Test Generation for Multi Agent Systems", 7th International joint conference on Autonomous agents and multiagent systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1315-1320, 2008.
186. Chang, S., Colace, F., Santo, M.D., Zegarra, E., Qie, Y., "An Approach for Software Component Reusing based on Ontological Mapping", Proceedings of the 24nd International Conference on Software Engineering & Knowledge Engineering (SEKE'2012), Redwood City, San Francisco Bay, CA, USA, 2012.
187. Iqbal, A., Ureche, O., Hausenblas, M., Tummarello, G.: LD2SD: Linked Data Driven Software Development. the 21th International Conference on Software Engineering and Knowledge Engineering (SEKE). Boston, Massachusetts (2009)
188. Schügerl, P., Rilling, J., Charland, P.: Enriching SE ontologies with bug report quality. In: Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering (SWESE '08). (2008).
189. Tappolet, J., "Semantics-aware Software Project Repositories", ESWC 2008 Ph.D. Symposium, June 2008.
190. Wang, H.H., Damljanovic, D., Sun, J., "Enhanced Semantic Access to Formal Software Models", in Proceedings of the 12th Internatinoal

Conference on Formal Engineering Methods and Software Engineering, pp. 237-252, 2010.
191. Keivanloo, I., Forbes, C., Rilling, J., Charland, P., "Towards Sharing Source Code Facts Using Linked Data", In Proceedings of the 3rd International Workshop on Search-Driven Development: Users, Infrastructure, Tools and Evaluation (SUITE '11), pp. 25-28, 2011.
192. Berger, O., "Linked Data Descriptions of Debian Source Packages Using ADMS.SW", In Proceedings of the 8th International Workshop on Semantic Web Enabled Software Engineering (SWESE 2012), Nara, Japan, 2012.
193. Shahri, H. H., Hendler, J. A., Porter, A.A.: Software Configuration Management Using Ontologies. the 3rd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2007). Innsubruk, Austria, (2007)
194. Damljanovic, D., Bontcheva, K., "Enhanced Semantic Access to Software Artefacts", 4th International Workshop on Semantic Web Enabled Software Engineering (SWESE'08), in collaboration with ISWC 2008, Karlsruhe, Germany, October, 2008.
195. Dietrich, J., Elgar, C., "A Formal Description of Design Patterns Using OWL", Proceedings of the 16th Australian Conference on Software Engineering (ASWEC2005), IEEE Computer Society, 2005.
196. Ren, Y., Gröner, G., Lemcke, J., Rahmani, T., Friesen, A., Zhao, Y., Pan, J. Z. & Staab, S. (2009). Validating Process Refinement with Ontologies.. In B. C. Grau, I. Horrocks, B. Motik & U. Sattler (eds.), Description Logics, : CEUR-WS.org.
197. Siegemund, K., Thomas, E.J., Zhao, Y., Pan, J., Assmann, U., "Towards Ontology-Driven Requirements Engineering", in Proceedings of the 7th International Workshop on Semantic Web Enabled Software Engineering, SWESE 2011.
198. Kaviani, N., Mohabbati, B., Gasevic, D., Finke, M., "Semantic Annotations of Feature Models for Dynamic Product Configuration in Ubiquitous Environments", In Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering at 7th International Semantic Web Conference, Karlsruhe, Germany, 2008.
199. Settas, D., Stamelos, I., "Using Ontologies to Represent Software Project Management Antipatterns", n proceeding of: Proceedings of the Nineteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2007), Boston, Massachusetts, USA, July 9-11, 2007.
200. Moroi, T., Yoshiura, N., Suzuki, S., "Conversion of Software Specifications in Natural Languages into Ontologies for Reasoning", 8th

International Workshop on Semantic Web Enabled Software Engineering (SWESE2012), 2012.
201. Siegemund, K., Zhao, Y., Pan, J.Z., Abmann, U., “Measure Software Requirement Specifications by Ontology Reasoning”, In proceeding of: Proceedings of the 8th International Workshop on Semantic Web Enabled Software Engineering (SWESE 2012), 2012.
202. Moser, T., Winkler, D., Heindl, M., Biffl, S., “Automating the Detection of Complex Semantic Conflicts between Software Requirements”, Proceedings of the 23nd International Conference on Software Engineering & Knowledge Engineering (SEKE'2011), 2011.
203. Maleshkova, M., Pedrinaci, C., Domingue, J., Alvaro, G., Martines, I., “Using Semantics for Automating the Authentication of Web APIs”, In Proceedings of the 9th international semantic web conference on The semantic web - Volume Part I (ISWC'10), Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, and Lei Zhang (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 534-549.
204. Berners-Lee, T.: Linked Data. Design Issues for the World Wide Web, http://www.w3.org/DesignIssues/LinkedData.html (2006)
205. LOD Cloud, http://lod-cloud.net
206. Dietrich, J., Jones, N., Wright, J., “Using social networking and semantic web technology in software engineering- use cases, patterns, and a case study”, Journal of Systems and Software, vol. 81, pp. 2183-2193, (2008)
207. Dietrich, J., Elgar, C., “Towards a Web of Patterns”, Journal of Web Semantics, Volume 5 , Issue 2 (June 2007), pp 108-116.
208. Korner, S.J., Brumm, T.: Improving Natural Language Specifications with Ontologies. Software Engineering and Knowledge Engineering (SEKE) (2009)
209. Robinson, W.N., Pawlowski, S.D., “managing requirements inconsistency with development goal monitors”, IEEE Transactions on Software Engineering, vol. 25, no. 6, pp. 816-835, (1999)
210. Dietze, S., Liu, D., Yu, H.Q., Pedrinaci, C., “Semantic Web-driven Development of Service-oriented Systems – Exploiting Linked Data for Services Annotation and Discovery”, 7th International Workshop on Semantic Web Enabled Software Engineering, Bonn, Germany, 2011.
211. OMG Unified Modeling Language (OMG UML) Superstructure, version 2.4. (2011)
212. Miller, G.: Wordnet: A Lexical Database for English. Commun. ACM 38, 39-41 (1995)
213. Wu, Z., Palmer, M.: Verb Semantics and Lexical Selection. Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics. pp. 133–138, (1994)

214. Lin, D.: An Information-Theoretic Definition of Similarity. Proceedings of the 15th International Conference on Machine Learning, vol. 1, 296-304 (1998)
215. Levenshtein, V. I.: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, Soviet Physics Doklady, Vol. 10, No. 8, pp. 707-710, (1966)
216. Conallen, J., "Building Web Applications with UML", Addison Wesley, 2002.
217. Hamilton, K., Miles, R., "Learning UML 2.0", O'Reilly, 2006.
218. Mellor, S.J., Balcer, M.J., "Executable UML: A Foundation for Model-Driven Architecture" Addison Wesley, 2002.
219. Alchimowicz, B., Jurkiewicz, J., Ochodek, M., Nawrocki, J., Building Benchmarks for Use Cases, Computing and Informatics, vol. 29, no. 1, 27-44 (2010)
220. Frakes, W.B., Baeza-Yates, R., Information Retrieval: Data Structures and Algorithms, Prentice-Hall, 1992.

واژه‌نامه

Adaptation	تطبیق
Annotation	حاشیه‌نویسی
Business Process Model	مدل فرآیند کسب و کار
Case Based Reasoning (CBR)	استدلال موردی
Conflict Resolution	حل تعارض
Difference computation	محاسبه تفاضل
Domain Engineering	مهندسی دامنه
Domain Knowledge	دانش دامنه
Domain Specific Language (DSL)	زبان خاص دامنه
Feature model	مدل ویژگی
Functional Requirements	نیازمندی‌های عملیاتی
Graph Matching	انطباق گراف
Incorporation	الحاق
Metamodel	فرامدل
Methodology	روشگان
Model Driven Architecture	معماری مبتنی بر مدل
Model Reuse	استفاده مجدد از مدل
Model Search Engine	موتور جستجوی مدل
Model Transformation	تبدیل مدل
Model-Driven Development (MDD)	توسعه مبتنی بر مدل
Navigation	ناوش
Ontology	هستان‌شناسی
Parser	مترجم
Part-Of-Speech (POS) Tagger	برچسب زننده نقش کلمات
Prototype	نمونه اولیه
Quantifiable	قابل کمی‌سازی
Query Expansion	توسعه پرس و جو
Requirements models	مدل نیازمندی‌ها
Salt	چاشنی
Semantic Web Enabled Software Engineering	مهندسی نرم‌افزار مبتنی بر وب معنایی
Software clone	کپی نرم‌افزاری
Software Components	مؤلفه‌های نرم‌افزاری
Source Code Search Engine	موتور جستجوی کد منبع
Systematic	سامانمند
Triple Store	مخزن سه‌گانه

Web engineering	مهندسی وب

Abstract

Web engineering has emerged as a asoftware discipline which specifically addresses systematic development of high-quality web applications. The methodologies proposed for web engineering mostly follow the Model Driven Development (MDD) style in which models are the main driver of the development process. A challenge with these methodologies is that developing each new web application requires creating a probably large set of models. Providing a model reuse approach can be considered as a main solution to this challenge. In the current thesis, a new reuse approach is proposed for specification of functional requirements models in the earliest steps of the development lifecycle. This approach takes the brief description of the functional requirements of a new web application in terms of UML use case diagram, and semi-automatically generates the draft of the detailed description in terms of UML activity diagrams. The proposed approach includes two main steps. In the first step a semantic model repository is prepared, and the second step provides a reuse process. New algorithms are employed in the first step for activity diagram annotation and behavior/concept detection. The second step introduces a new metric for measuring similarity of use cases and also a new algorithm for activity diagram adaptation. In addition, the semantic web technologies are used in the first step to provide a flexible representation layer, and the semantic web sources are used in the second step as a source for addresseing information needs of the proposed approach. Experimental evaluations demonstrate that the proposed approach has good precision and recall, and the use of the semantic web has improved the quality. Consequently, the meain underlying idea of the proposed approach, i.e. using UML use case diagrams as the starting point of the reuse approach, is proved to be sound and prommissing. However, the proposed algorithms have still some weaknesses and their improvement needs more research and development.