علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۱، شماره ۱ (الف)، ۱۳۹۲
صفحات ۳۰-۴۳
مقاله عادی

مروری بر روش‌ها و فرصت‌های ایجاد خط تولید نرم‌افزار برای سیستم‌های قدیمی با رویکرد مهندسی مجدد

الهام درمنکی فراهانی[۱] عباس حیدرنوری[۲] جعفر حبیبی[۲]

[۱]دانشگاه صنعتی شریف، پردیس بین‌المللی کیش، کیش، ایران
[۲]دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران

چکیده

از مفهوم خط تولید نرم‌افزار[۱] جهت ایجاد خانواده‌ای از محصولات نرم‌افزاری مشابه استفاده می‌شود. برای این منظور، در اکثر شرکت‌های نرم‌افزاری از طریق مهندسی مجدد سیستم‌های قدیمی اقدام به ایجاد خط تولید نرم‌افزار می‌گردد. دلیل این امر آن است که از آنجا که سیستم‌های قدیمی اغلب دارای سابقه‌ای طولانی و دارای نسخه‌های مختلف با نقاط اشتراک و افتراق فراوان می‌باشند، پتانسیل زیادی برای پیکربندی مجدد و تبدیل شدن به خط تولید نرم‌افزار را دارند. لذا مبحث مهندسی مجدد[۲] سیستم‌های قدیمی از جمله فرآیندهای مهم در مبحث ایجاد خط تولید نرم‌افزار به شمار می‌رود. با وجود اینکه در این زمینه کارهای صنعتی فراوانی صورت گرفته، ولیکن تاکنون یک نگاه کلی که سیر تحول مهندسی مجدد سیستم‌های قدیمی جهت ایجاد خطوط تولید را نشان دهد، انجام نپذیرفته است. بدین منظور، در این مقاله قصد آن داریم تا یک مرور کلی بر روی پژوهش‌هایی که در زمینه فرآیندهای مختلف مهندسی مجدد و نیز روش‌های مهندسی معکوس استخراج مدل و کدهای سیستم‌های قدیمی با رویکرد ایجاد خط تولید نرم‌افزار صورت گرفته، داشته باشیم تا بتوانیم روش‌های موجود و چالش‌های پیش روی تحقیقات آینده در این زمینه را شناسایی نماییم و آنها را به صورت منظمی دسته‌بندی نماییم.

کلمات کلیدی: خط تولید نرم‌افزار، مهندسی مجدد، سیستم‌های قدیمی، متدولوژی، بازآرایی[۳] جنبه‌گرا، بازآرایی ویژگی‌گرا، بازآرایی شیءگرا.

۱- مقدمه

خط تولید نرم‌افزار در صنعت، جهت تولید منظم خانواده‌ای از سیستم‌های نرم‌افزاری استفاده می‌شود [۱]. خط تولید نرم‌افزار ترکیبی می‌باشد از توسعه منظم و استفاده مجدد از مؤلفه‌هایی که قسمت‌های مشترک و متغیر خط تولید نرم‌افزار را شامل می‌شوند [۲، ۳].

بسیاری از سیستم‌های قدیمی با قابلیت پیکربندی بسیار بالا و سابقه‌ای طولانی، دارای نسخه‌ها و متغیرهای بسیار، مستعد مهندسی مجدد و تبدیل شدن به خط تولید نرم‌افزار می‌باشند. تکنیک‌های بسیار زیادی جهت تبدیل سیستم‌های قدیمی به خط تولید وجود دارد. تعدادی از آنها به روش‌های عمومی مهندسی مجدد شباهت فراوانی دارند[۱۰، ۱۱، ۱۲]، برخی دیگر به تکنیک‌های تبدیل کننده مدل[۴] [۳۴، ۳۵]، و برخی دیگر به تکنیک‌های تبدیل کدهای برنامه[۵] [۴۹، ۵۰]. پیشنهادات آکادمیک و تجربیات صنعتی متنوعی این مباحث را مورد توجه قرار داده‌اند. اما یک مرور کلی که کل چرخه تکامل فرآیند مهندسی مجدد سیستم‌های قدیمی و ایجاد خط تولید نرم‌افزار را در برگیرد وجود ندارد. دراین مقاله انواع روش‌های تبدیل سیستم‌های قدیمی به خط تولید نرم‌افزار را مورد بررسی قرار خواهیم داد. در این مقاله سعی شده است تا مستندات علمی مابین سال‌های ۱۹۹۸ تا ۲۰۱۲ مورد بررسی قرار گیرند.

ادامه مقاله به این صورت سازماندهی شده است. در بخش دوم مفهوم مهندسی مجدد مرور خواهد گردید. سپس در بخش سوم کارهای انجام شده در خصوص فرآیند مهندسی مجدد سیستم‌های قدیمی مورد بررسی قرار خواهند گرفت. در

ادامه در بخش‌های چهارم و پنجم به ترتیب پژوهش‌های انجام شده در خصوص مهندسی مجدد مدل‌ها و کدهای سیستم‌های قدیمی ارائه خواهند شد. در بخش ششم، انواع روش‌های مهندسی مجدد را مقایسه و ارزیابی خواهیم نمود. در بخش هفتم خلاصه‌ای از چالش‌های باقیمانده و فرصت‌های مطالعاتی در رابطه با این مبحث مورد بررسی قرار خواهند گرفت و در نهایت، در بخش هشتم نتیجه‌گیری مطالب آمده است.

۲- مفهوم مهندسی مجدد

مهندسی مجدد نرم‌افزار یک مفهوم قدیمی با تاریخچه‌ای طولانی می‌باشد. مهندسی مجدد نرم‌افزار ابتدا به صورت "بررسی و تغییر یک سیستم جهت بازسازی آن به شکلی جدید" تعریف شده است [۹۱]. هدف اصلی مهندسی مجدد، رسیدن به سطح جدیدی از کارآرایی موجودیت‌های موجود بدون نیاز به تکرار دوباره چرخه توسعه سیستم می‌باشد. در حقیقت، معمولاً برای رفع اشکالات و بالا بردن سطح کیفی یک سیستم ممکن است یک یا چند بار چرخه توسعه سیستم تکرار شود ولی در مهندسی مجدد رسیدن به کیفیت بالاتر نرم‌افزار بدون تکرار این چرخه مدنظر می‌باشد.

به طور معمول، در مهندسی مجدد، یک سیستم قدیمی دوباره ساختاردهی می‌شود تا به یک سیستم جدید با ویژگی‌های عملکردی بهتر تبدیل شود و با محدودیت‌های جدید نرم‌افزاری و سخت‌افزاری سازگاری بیشتری پیدا کند. از مفهوم مهندسی مجدد به خوبی جهت پیدا کردن روندی برای ایجاد خط تولید نرم‌افزار از روی سیستم‌های قدیمی می‌توان بهره برد.

به غیر از اصطلاح مهندسی مجدد، اصطلاحاتی مانند مهاجرت[۶] و تکامل[۷] نیز مرتبط با این حوزه می‌باشند. از اصطلاح مهاجرت بیشتر در زمان تغییر زبان‌ها، پایگاه‌های داده یا سیستم عامل استفاده می‌شود و اغلب برای مبدل‌های خودکار تغییر فرم‌دهنده سیستم‌های قدیمی استفاده می‌شود.

جدول ۱- خلاصه فعالیت‌های صورت گرفته در زمینه فرآیند مهندسی مجدد سیستم‌های قدیمی و ایجاد خط تولید

سال	نام مولف	روش ابداعی یا بررسی موردی	مؤسسه	روش اعتبارسنجی
۱۹۹۸	DeBaud [۱۰]	مهندسی مجدد به عنوان نقطه شروع ایجاد خط تولید	SEI	
۱۹۹۸	Weiderman [۱۱]	رهنمودهایی برای سازمان‌ها	SEI	
۱۹۹۸	Scherlis [۱۲]	تکنیک‌های مدیریت و دستی طراحی	SEI	
۱۹۹۹	Bayer [۵]	روش RE-PLACE	IESE	نمونه صنعتی
۲۰۰۱	Stoermer [۱۳]	روش MAP	SEI	بررسی موردی
۲۰۰۲	Smith [۷۰]	بررسی موردی روش OAR	SEI	بررسی موردی
۲۰۰۲	O'Brien [۱۴]	روش OAR	SEI	
۲۰۰۲	Raghavan [۱۸]	واسط کاربری مشترک برای مؤلفه‌ها (روش ADORE)	IESE	
۲۰۰۴	Hansen [۲۳]	کارگاه عملی برای مبحث مهندسی مجدد		بررسی موردی
۲۰۰۴	Pinzger [۹]	فرآیند محور بر اساس روش PLUSE	IESE	نمونه صنعتی
۲۰۰۵	Knodel [۱۶]	بازیابی دارایی‌ها	IESE	
۲۰۰۵	Knodel [۱۷]	تجزیه و تحلیل کفایت دارایی‌ها	IESE	نمونه صنعتی
۲۰۰۵	Kang [۲۰]	روبات سرویس کنترل خانه و خط تولید سیستم HSR		بررسی موردی
۲۰۰۸	Breivold [۱۹]	بر مبنای تحلیل وابستگی‌ها، ارزیابی کیفی صفات (تحلیل پذیری، قابلیت گسترش و. .) در خط تولید شرکت ABB		نمونه صنعتی
۲۰۰۸	Hubaux [۲۶]	بررسی مکانیزم تغییرپذیری خط تولید PloneMeeting		بررسی موردی
۲۰۰۸	Hubaux [۲۷]	جداسازی دغدغه‌ها[۸] در خط تولید PloneMeeting		بررسی موردی
۲۰۰۹	Lee [۲۲]	راه‌اندازی جعبه ساخت خط تولید برای شرکت		نمونه صنعتی
۲۰۱۰	Jun [۲۱]	خط تولید سیستم مدیریت کسب و کار دادگاه		نمونه صنعتی
۲۰۱۰	Ghanam [۲۵]	فرآیند توسعه چابک در خط تولید Buddi		بررسی موردی
۲۰۱۱	Bosch [۲۴]	فرآیند توسعه چابک در خط تولید Quickbooks		نمونه صنعتی
۲۰۱۱	Bartholdt [۲۸]	مهندسی مجدد خط تولید سیستم Siemens healthcare imaging		نمونه صنعتی
۲۰۱۱	Wu [۲۹]	فرآیند کشف کلونی‌های نرم‌افزاری در خط تولید DirectBank		نمونه صنعتی
۲۰۱۱	Zhang [۳۰]	فرآیند چابک و تدریجی: خط تولید Alcatel-Lucent		نمونه صنعتی
۲۰۱۲	Ramos [۳۲]	مهندسی مجدد سیستمی از سیستم‌ها (SOS)		بررسی موردی
۲۰۱۲	Almorsy [۳۱]	ایجاد چارچوب SMURF در سیستم‌های سرویس‌گرا		بررسی موردی

همچنین اصطلاحات کامل به "فرآیند تغییر ویژگی‌های موجودیت‌های در حالت حول یا عناصر تشکیل دهنده آن" اشاره دارد [۹۲]. معمولاً اصطلاحات کامل به تغییرات نرم‌افزار در طول زمان اشاره دارد و زمانی اتفاق می‌افتد که نیازمندی‌های اولیه تغییر یافته باشند [۹۳].

با توجه به مفاهیم مختلف مهاجرت و تکامل، در این مقاله قصد داریم تا بر روی اصطلاح "مهندسی مجدد" به مفهوم تغییر در سیستم‌های قدیمی به منظور ایجاد خط تولید نرم‌افزار تمرکز نماییم و یک مرور کلی بر روی مباحث مرتبط با این اصطلاح داشته باشیم.

۳- فعالیت‌های انجام شده در خصوص فرآیند مهندسی مجدد

در جدول ۱ روش‌ها، متدلوژی‌ها و فرآیندهایی که تاکنون در خصوص مهندسی مجدد سیستم‌های قدیمی و ایجاد خط تولید نرم‌افزار ارائه شده‌اند، به صورت خلاصه آمده است. همچنین در ستون آخر این جدول می‌توان مشاهده نمود که هر یک از این روش‌ها در چه محیطی مورد آزمایش و به کار برده شده‌اند.

در این جدول، فعالیت‌هایی که در این زمینه توسط موسسات SEI و IESE صورت پذیرفته است نیز گردآوری شده است. این دو مؤسسه از پیشگامان فرهنگ خط تولید نرم‌افزار به شمار می‌روند و زیرساخت‌های لازم در استفاده مجدد از مؤلفه‌های سیستم‌ها را به وجودآورده‌اند.

مستندات مربوط به مؤسسه SEI، شامل مقالاتی از De Baud [۱۰]، Weiderman [۱۱] و Scherlis [۱۲] می‌باشند که مسایلی را در خصوص فرآیند عمومی مهندسی مجدد و نه مختص به خط تولید نرم‌افزار را مورد بررسی قرار داده‌اند. از جمله روش‌های ارائه شده توسط این مؤسسه می‌توان به روش MAP[۹] [۱۳] و نیز روش OAR[۱۰] [۱۴] اشاره نمود. همچنین در مقالات [۹] و [۱۵] از این روش‌ها به ترتیب در نمونه‌هایی عملی برای استخراج و دوباره‌سازی معماری‌ها استفاده شده است.

مؤسسه IESE نیز روش RE-PLACE[۱۱] [۵] را ارائه نموده است که روشی برای مهندسی مجدد سیستم‌های قدیمی و استخراج معماری خط تولید نرم‌افزاربه شمارمی‌رود. ارائه‌کنندگان این روش، همچنین در یک نمونه صنعتی روش‌های پیشنهادی خود را مورد استفاده قرار داده‌اند. Knodel [۱۶] مستندات موجود در این زمینه را در کنار هم قرار داده و با یکدیگر مقایسه نموده و در نهایت روش ADORE[۱۲] را به عنوان یک روش مهندسی مجدد تقاضا محور[۱۳] پیشنهاد نموده است [۱۷].

باقی مستندات موجود در این زمینه، اساساً شامل مطالعات موردی و یا تجربیات صنعتی انجام شده توسط روش‌های ارائه شده قبلی و آموخته‌های یافت شده از این تجربیات می‌باشد. برای نمونه Raghavan [۱۸] در شرکت نوکیاو Pinzger [۹] در شرکت نوکیا و فیلیپس با همکاری مؤسسهIESE و همچنین Breivold [۱۹] در شرکت ABB نمونه‌هایی از این دست هستند که تجربیات عملی صنعتی در این زمینه را ارائه نموده‌اند.

علاوه بر این‌ها، Kang [۲۰]، Jun [۲۱] و Lee [۲۲] پروژه‌هایی صنعتی در زمینه مهندسی مجدد را با موفقیت به پایان رسانیده‌اند. در نهایت Hansen [۲۳] انواع چالش‌هایی که ممکن است در زمینه مهندسی مجدد سیستم‌های قدیمی در یک سازمان در ارتباط با ذینفعان به وجود آیند را مورد توجه قرار داده است. اینگونه چالش‌ها ممکن است فرآیند مهندسی مجدد را در یک سازمان با مشکلات زیادی روبرو سازد و مسایل را پیچیده‌تر نمایند. در هر حال، در طی چند سال اخیر، موارد دیگری نیز در ارتباط با فرآیند مهندسی مجدد مورد توجه قرار گرفته‌اند که به شرح ذیل می‌باشند:

۱) فرآیند مهندسی مجدد به صورت چابک توسط Bosch [۲۴] و Ghanam [۲۵].
۲) بررسی چالش‌های مرتبط با مهندسی مجدد توسط Hubaux [۲۶، ۲۷].
۳) خط تولید نرم‌افزار سلسله مراتبی توسط Bartholdt [۲۸].
۴) فرآیند پایین به بالاجهت کشف مشترکات توسط Wu [۲۹].
۵) فرآیند چابک و تدریجی مهندسی مجدد در شرکت Alcatel-Lucent توسط Zhang [۳۰].

اخیراً در سال ۲۰۱۲ نیز توسط Ramos [۳۲] در زمینه مهندسی مجدد سیستمی از سیستم‌ها فعالیت‌هایی صورت گرفته است. در این مقاله به این اشاره شده است که ممکن است مجموعه‌ای از سیستم‌ها وجود داشته باشد که اشتراک‌های زیادی نیز در ابتدا با یکدیگر ندارند ولی پس از انجام مهندسی مجدد مشترکات زیادی پیدا کرده و می‌توان به عنوان یک خط تولید نرم‌افزار آنها را در نظر گرفت.

همچنین در همین سال Almorsy [۳۱] چارچوب SMURF را جهت مهندسی مجدد سیستم‌های سرویس‌گرا معرفی نموده است و ذکر نموده است که مهندسی مجدد اینگونه سیستم‌ها که به صورت سرویس عمل می‌نمایند بایستی با جزئیات بیشتری صورت پذیرد تا قابلیت سرویس‌دهی اینگونه سیستم‌ها به دیگر سیستم‌ها پس از مهندسی مجدد دچار اختلال نشود.

از تمامی این مقالات و مستندات علمی می‌توان نتیجه گرفت که گرایش، بیشتر به سمت متدلوژی توسعه چابک است و اکثراً از متدولوژی توسعه سنتی فاصله گرفته‌اند. در کل، مستندات فعلی در خصوص فرآیند و متدولوژی مهندسی مجدد سیستم‌های قدیمی کامل و نهایی نیستند و به جزئیات پرداخته نشده است و به عنوان یک مبحث باز می‌تواند هنوز مطرح باشد.

۴- فعالیت‌های انجام شده در خصوص استخراج مدل سیستم‌های قدیمی

در جدول ۲ خلاصه‌ای از مستندات مرتبط با مهندسی مجدد مدل‌های سیستم‌های قدیمی موجود جهت ایجادیک خط تولید نرم‌افزار جدید ارائه شده است. این مقالات به لحاظ روش استفاده شده در مهندسی مجدد به دو گروه اصلی تقسیم شده‌اند:

۱) در روش اول از مستندات قدیمی برای ساخت مستندات و مدل‌های خط تولید نرم‌افزار استفاده شده است و معمولاً اینکار به صورت دستی صورت می‌پذیرد.
۲) در روش دوم از کد برنامه‌های قدیمی و ابزار برایبه دست آوردنمدل‌ها و مستندات خط تولید نرم‌افزار استفاده می‌شود.

در رابطه با گروه اول، John [۳۴، ۳۵] و Knodel [۳۵] تلاش نموده‌اند تا از مستندات سیستم‌های قدیمی و موجود در ایجادمدل‌های مورد نیاز در خط تولید استفاده نمایند. برای این منظور آنها از تکنیک‌های موجود در مهندسی مجدد استفاده نموده و آنها را با مفاهیم خط تولید نرم‌افزار تطبیق داده‌اند. علاوه بر اینها Pashov [۳۶] و Kim [۳۷] از مفهوم مدل ویژگی‌ها برای ایجاد معماری مؤلفه‌گرای سیستم‌های قدیمی استفاده نموده‌اند.

همچنین در مقالاتی که توسط Ajila [۳۸] و Conrjero [۳۹] منتشر شده‌اند، از دسته‌بندی‌های مرزی[۱۴] [۳۶] و ماتریس‌های ردیابی[۱۵] [۳۷] برای کشف دغدغه‌های سراسری[۱۶] استفاده شده است. البته فرآیند در تمامی آنها هنوز به صورت دستی انجام شده است.

جدول ۲- خلاصه فعالیت‌های صورت گرفته در زمینه استخراج مدل خط تولید از مستندات و کدهای قدیمی

سال	مولف	فعالیت صورت گرفته	ابزار استفاده شده	روش اعتبارسنجی
۲۰۰۲	John [۳۴]	استفاده از مستندات قدیمی		نمونه صنعتی
۲۰۰۴	Pashov [۳۶]	بازیابی معماری و مدل‌های ویژگی در سیستم پردازش تصویر		نمونه صنعتی
۲۰۰۵	Knodel [۳۵]	بازیابی دارایی‌ها در خطوط تولید		
۲۰۰۵	Ajila [۳۸]	استخراج ویژگی		
۲۰۰۵	Chen [۴۳]	تکنیک‌های تحلیل ویژگی‌ها با استفاده از ابزار مهندسی مجدد	RECON	بررسی موردی
۲۰۰۶	John [۳۵]	استفاده از مستندات قدیمی		نمونه صنعتی
۲۰۰۶	Graaf [۳۶]	مهاجرت توسط تغییر فرم مدل‌ها	ATL	بررسی موردی
۲۰۰۷	Kim [۳۷]	بازیابی معماری و مدل‌های ویژگی در خط تولید صدا و تصویر دیجیتال		نمونه صنعتی
۲۰۰۷	Kastner [۴۸]	تحلیل ویژگی‌گرای کدهای قدیمی	CIDE	
۲۰۰۹	Conejero [۳۹]	تحلیل دغدغه‌های سراسری		
۲۰۱۰	Stoiber [۴۰]	استفاده از روش Feature Unweaving، بازآرایی خودکار مدل	ADORA	

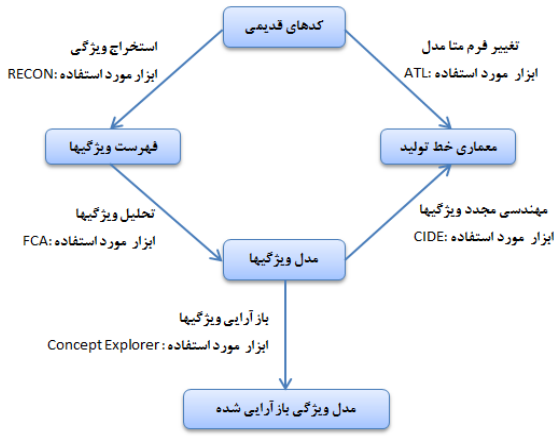
جدول ۳- خلاصه‌ای از ابزارهای استفاده شده جهت استخراج مدل خط تولید

ابزار	کارکرداصلی	روش دسترسی	قابلیت‌ها
ATL [۷۱]	تغییر شکل مدل	متن باز	مدل‌های محاوره‌ای مبتنی بر MDE
ADORA [۷۲]	بازآرایی مدل	متن باز	ابزار مدیریت نیازمندی‌های غیر استاندارد
RECON [۷۳]	مهندسی مجدد جهت کشف ویژگی‌ها	رایگان	کشف ویژگی‌ها در کد
FEAT [۷۴]	گراف دغدغه‌ها	رایگان Eclipse Plug-in	جستجوی وابستگی‌های ساختاری برنامه
CIDE [۷۵]	بازآرایی کد	رایگان Eclipse Plug-in	مشخص نمودن ویژگی‌ها با رنگ‌های مختلف
Dali [۷۶]	مهندسی مجدد	ابزار یکپارچه SEI	استخراج معماری، پشتیبانی از روش MAP

در مقاله‌ای که توسط Stoiber [۴۰] نیز منتشر شده است از روش Feature Unweaving و با استفاده از یک ابزار گرافیکی بر مبنای زبان ADORA [۴۱] جهت استخراج ویژگی‌های متغیر استفاده نموده است. ابزار مورد استفاده آنها [۴۲] به صورت اتوماتیک یک مدل ADORA را به مدلی تبدیل می‌نماید که در آن عناصر وابسته به هر ویژگی به صورت یک جنبه[۱۷] ظاهر شده‌اند.

Chen [۴۳] از تحلیل‌های ایستا و پویای ویژگی‌ها برای جا دادن ویژگی‌ها در توابع نهایی پیاده‌سازی شده استفاده نموده است. تمامی این روش‌ها توسط ابزارهای تجاری و رایگان مانندRECON[۱۸] [۴۴] یاFEAT[۱۹] [۴۵] پشتیبانی می‌شوند. Graaf [۴۶] از یک تبدیل کننده خودکار مدل برایبه دست آوردن معماری خط تولید نرم‌افزار از روی معماری سیستم‌های قدیمی استفاده نموده است. آنها برای این امر مجموعه‌ای از قوانین تبدیل را به زبانATL[۲۰] [۴۷] ایجاد نمودند. البته روش پیشنهادی آنها تنها زمانی امکان‌پذیر است که بتوان تغییرپذیری را توسط یک فرا مدل[۲۱] تعریف و نمایش داد.

در نهایت Kastner [۴۸] نیز ابزاری به نام Colored IDE (CIDE) را جهت خودکارسازی استخراج ویژگی‌ها[۲۲] از کد سیستم‌های قدیمی ابداع نموده است. در این روش، هر یک از بخش‌های کد برنامه که مربوط به یک ویژگی خاص می‌باشد با رنگی متفاوت علامت‌گذاری می‌شود. البته این روش نیز همچنان دستی باقیمانده است و وابسته به تجربیات کاربر نیز می‌باشد. با بررسی تلاش‌های انجام گرفته می‌توان نتیجه گرفت که تکنیک‌های مهندسی مجدد می‌تواند به صورت سودمندی برای استخراج ویژگی‌ها مورد استفاده قرار گیرند. ابزارهایی مانند FEAT و RECON نیز می‌توانند در این رابطه سودمند واقع شوند. می‌توان با استفاده از تکنیک‌های مهندسی مجدد در یک فرآیند تعریف شده روشی را جهت استخراج مدل ویژگی (و نه فقط شناسایی ویژگی‌ها به تنهایی) ابداع نمود و این موضوع هنوز به صورت یک چالش باقیمانده است. در صورتیکه قسمت‌های مختلف کد برنامه بتواند به صورت خودکار به هر ویژگی نسبت داده شود، می‌توان از ابزار CIDE در این زمینه استفاده نمود. در جدول ۳ خلاصه‌ای از ابزارهای نامبرده شده در این قسمت ارائه شده است. همچنین شکل ۱ نیز تصویری کلی از توالی فعالیت‌های مربوط به استخراج مدل و مهندسی مجدد را نشان می‌دهد.



شکل ۱- توالی فعالیت‌های مربوط به استخراج مدل و مهندسی مجدد

۵- فعالیت‌های صورت گرفته در زمینه مهندسی مجدد کدهای قدیمی

جدول ۴ شامل مقالات و مستنداتی است که مربوط به مهندسی مجدد کد برنامه‌های قدیمی و تبدیل آنها به رویکردهای شیءگرا (OOP)[۲۳]، ویژگی‌گرا (FOP)[۲۴]، یا جنبه‌گرا (AOP)[۲۵]می‌باشد. این دستاوردها در نهایت معماری اولیه خط تولید نرم‌افزار را تشکیل می‌دهند. در زمینه برنامه‌نویسی شیءگرا در مقایسه با سایر روش‌ها مطالب علمی بیشتری یافت می‌شود.

همچنین برنامه‌نویسی ویژگی‌گرا نیز می‌تواند در دسته برنامه‌نویسی شیءگرا قرار گیرد زیرا تقریباً نوع پیشرفته برنامه‌نویسی شیءگرا می‌باشد. ولیکن، برنامه‌نویسی ویژگی‌گرا کمتر از جنبه‌گرا مورد توجه قرار گرفته شده است. لذا در ادامه، مطالب در دو قسمت برنامه‌نویسی شیءگرا (به همراه ویژگی‌گرا) و برنامه‌نویسی جنبه‌گرا ارائه خواهند شد.

۵-۱- مهندسی مجدد کدهای قدیمی به روش برنامه‌نویسی شیءگرا و ویژگی‌گرا

Zhang [۴۹]، Simon [۵۰]، Akers [۵۱] و Olszak [۵۲] ابزارهای متفاوتی را جهت تحلیل و تبدیل سیستم‌های قدیمی خط تولید نرم‌افزار پیشنهاد نموده‌اند و در تمامی آنها از تکنیک‌ها و ابزار مهندسی مجدد و زبان برنامه‌نویسی متداول شیءگرا بهره گرفته شده است. در [۴۹] از ابزاری به نام XVCL برای ساخت خط تولید تلفن همراه استفاده شده است و تطبیق‌پذیری مابین بسترهای نرم‌افزاری J2SE و J2ME بررسی شده است. Simon از یک ابزار به نام XFIG [۵۰] و از یک تکنیک مناسب مهندسی مجدد بر مبنای تحلیل مفهومی صوری به نام FCA جهت تحلیل ویژگی‌ها استفاده نموده‌اند.

Akers [۵۱] ابتدا یک سیستم طراحی نگهداری (DMS) که یک ابزار تجاری به شمار می‌رود را تشریح نموده و سپس بحث نموده استک هچگونه این سیستم را در یک سیستم کنترل ارتباطات هوایی که به زبان C++ نوشته شده است به کار برند. شرکت Boeing قبلاً خط تولیدی از این سیستم کنترل ارتباطات هوایی را ایجاد نموده است. این سیستم به عنوان یک سیستم توکار[۲۶] به شمار می‌رود. DMS یک ابزار کامل مهندسی مجدد است که عملیات تحلیل را انجام داده و بر روی نمایش درخت واره یک برنامه اجرا می‌شود.

Chaing [۵۳] نیز یک معماری توزیع شده را برای استفاده مجدد از سیستم‌های قدیمی داده‌گرا در صنعت ارائه نموده است. Olszak [۵۲] نیز یک متد و دو بررسی موردی را برای پیمانه‌ای کردن مجدد ویژگی‌گرا مختص برنامه‌های نوشته شده به زبان Java ارائه نموده است. آنها برنامه را پیمایش می‌نمایند تا ویژگی‌ها را در سیستم‌های نوشته شده به زبان Java برنامه مشخص نموده و پیمانه‌های ویژگی را به صورت کاملاً صریح مشخص نمایند.

جدول ۴- خلاصه فعالیت‌های صورت گرفته در زمینه مهندسی مجدد کدهای قدیمی

سال	مولف	فعالیت صورت گرفته	ابزار استفاده شده	روش اعتبارسنجی
۲۰۰۲	Zhang [۴۹]	مهندسی مجدد و ایجاد خط تولید دستگاه تلفن همراه (J2SE به J2ME)	XVCL	بررسی موردی
۲۰۰۲	Simon [۵۰]	تحلیل کدهای قدیمی با استفاده از FCA	XFIG	
۲۰۰۴	Chiang[۵۳]	ایجاد یک معماری توزیع شده از طریق معماری Wrapping		نمونه صنعتی
۲۰۰۵	Alves [۶۵]	ایجاد خط تولید بازی‌های موبایل (AspectJ)	FLiP	بررسی موردی
۲۰۰۶	Liu [۵۵]	تئوری بازآرایی ویژگی‌گرا و کاربرد آن در جاوا	FOR	توصیف صوری
۲۰۰۶	Trujillo [۵۶]	بازآرایی کدها و مستندات در یک خط تولید (جاوا)		بررسی موردی
۲۰۰۷	Akers [۵۱]	استفاده از ابزار مهندسی مجدد جهت ساخت خط تولید سیستم کنترل خطوط هوایی Avionic (تغییر کدهای C++)	DMS	نمونه صنعتی
۲۰۰۷	Kästner [۶۴]	بازآرایی Berkeley DB و کشف ۳۸ ویژگی		بررسی موردی
۲۰۰۷	Kästner [۶۳]	بازآرایی ویژگی‌گرا (AspectJ)	CIDE	
۲۰۰۷	Calheiros[۱۹]	ایجاد ابزار بازآرایی تغییرپذیری خط تولید (AspectJ)	FLiP	بررسی موردی، ایجاد ابزار
۲۰۰۸	Alves [۶۶]	استخراج کدهای خط تولید از طریق برنامه‌نویسی جنبه‌گرا (AspectJ)	FLiP	ایجاد ابزار
۲۰۰۹	Olszak [۵۲]	پیدا کردن ویژگی‌ها در ماژولهای جاوا		بررسی موردی
۲۰۰۹	Kästner [۷۷]	جداسازی فیزیکی و مجازی دغدغه‌های سراسری (جاوا)		توصیف صوری، بررسی موردی
۲۰۱۰	Ghanam [۵۹]	روش Test-Driven برای بازآرایی (جاوا)	Eclipse plugin	بررسی موردی
۲۰۱۰	Dabholkar [۶۰]	استفاده از متد FORM برای چارچوب میان افزار	MPC	بررسی موردی
۲۰۱۰	Tizzei [۷۸]	نگاه جنبه‌گرا به مهندسی مجدد ویژگی‌گرا		بررسی موردی
۲۰۱۱	Lopez-Herrejon [۵۷]	بازآرایی برای استخراج خطوط تولید با استفاده از برنامه‌نویسی ویژگی‌گرا (جاوا)		بررسی موردی
۲۰۱۱	Xue [۶۱]	تحلیل تغییرپذیری برای کشف محل کلون و ویژگی‌ها (جاوا)		
۲۰۱۱	Lozano [۶۲]	مروری بر روش‌های کشف تغییرات در کد برنامه		

به هر حال در یک برنامه نمی‌توان کاملاً کدهای مربوط به هر ویژگی را از یکدیگر جدا ساخت و حتماً کدهایی هستند که در چند ویژگی مشترک می‌باشند. چندین مقاله نیز پیشنهاد نمودند تا از الگوی FOP [۵۴] برای مدیریت ویژگی‌ها در سطح کد استفاده شود و برای این کار نسخه اصلاح شده Java و ATS[۲۷] را پیشنهاد نموده‌اند.

Liu [۵۵] یک ابزار بازآرایی ویژگی‌گرا (FOR[۲۸]) را جهت تجزیه یک برنامه به ویژگی‌ها (معکوس ترکیب ویژگی‌ها) ایجاد نموده‌اند. در این ابزار هر عضو یک کلاس به یک ویژگی اختصاص داده می‌شود و ابزار بازآرایی آنها بر مبنای Eclipse ایجاد شده است. ولی مشکل اینجاست که این ابزار به صورت عمومی در اختیار نمی‌باشد.

Trujilo [۵۶] یک بررسی موردی در زمینه بازآرایی دستی ویژگی‌های داخلی نرم‌افزار ATS را ارائه نموده‌است. بدین صورت که هرویژگی (شامل کد، فایل‌های مرجع، مستندات html و آزمون رگرسیون خاص آن ویژگی) تنها در یکی از زیر مجموعه ابزارهای این نرم‌افزار کپسوله می‌شود. Lopez-Herrejon [۵۷] کاتالوگی از روش‌ها برای بازآرایی و استخراج ویژگی‌های متغیر از سیستم‌های قدیمی ارائه نموده است.

این کاتالوگ شامل روش‌های بازآرایی مانند اضافه کردن کد در ابتدای پیمانه، اضافه کردن کد در انتهای پیمانه، اضافه کردن کد در هر جای پیمانه که نیاز است، و بازنویسی پیمانه می‌باشد. هدف، سازماندهی مجدد کد به صورت ویژگی‌گرا (FOP) می‌باشد. (از Feature IDE [۵۸] برای نوشتن متغیرها استفاده می‌شود).

روش‌های مشخص دیگری نیز در این زمینه معرفی شده‌اند. برای مثال Ghanam [۵۹] یک روش توسعه چابک برای بازآرایی کدهای موجود با استفاده از آزمون‌هایی برای کشف تغییرپذیری‌ها معرفی نموده است. آنها یک Eclipse Plug-in به نام Product Line Designer را ایجاد نمودند. محدودیت این ابزار ایجاد شده عدم پشتیبانی از وابستگی نقاط تغییرپذیری و متغیرها می‌باشد.

Dabholkar [۶۰] روش FORM[۲۹] که یک روش مهندسی معکوس ویژگی‌گرا مختص میان افزارهاست را معرفی نموده است. این روش بر مبنای تحلیل کد برنامه است و در سه سطح پیمانه‌ای می‌شوند: نگاشت ویژگی‌ها، کشف مجموعه بسته‌ها، و ترکیب میان‌افزار[۳۰]. در این روش، از یک کامپایلر به نام MPC[۳۱] نیز استفاده گردیده است.

Xue [۶۱] نیز استفاده از ابزارهای موجود را برای تشخیص کلونی‌های نرم‌افزاری[۳۲] و مکان ویژگی‌ها جهت خودکارسازی فرآیندهایی که تاکنون به صورت دستی ابداع و ارائه شده‌اند پیشنهاد نموده است. در نهایت، Lozano [۶۲] یک مرور کلی بر روی کشف مفهوم تغییرپذیری در کد برنامه داشته و مفاهیم مرتبط را سازماندهی نموده است.

۵-۲- مهندسی مجدد کدهای قدیمی به روش برنامه‌نویسی جنبه‌گرا

در طی سالیان اخیر، گرایش زیادی به استفاده از تکنیک‌های برنامه‌نویسی جنبه‌گرا (AOP) برای پیاده‌سازی تغییرپذیری خط تولید نرم‌افزار ایجاد شده است. Kastner [۶۳] از ابزار CIDE برای علامت‌گذاری نمودن کدهای مربوط به هر ویژگی در یک سیستم قدیمی نوشته شده به زبان Java استفاده می‌کند. برای پیاده‌سازی این روش، از بازآرایی AspectJ استفاده می‌شود.

در مقالات مرتبط دیگری نیز Kastner [۶۴] استفاده از AspectJ را توسط یک بررسی موردی مورد ارزیابی قرار داده است و یک سیستم بانک اطلاعاتی توکار به نام Berkeley DB را بازآرایی نموده و ۳۸ ویژگی را علامت‌گذاری نموده است. متاسفانه نتایج نهایی این تجربیات همگی منفی بوده است و دلیل آن مشکلات متعددی است که با خود زبان AspectJ داشته‌اند و به همین دلیل نویسندگان راه حلهای جایگزینی را که نزدیک به روش‌های مرسوم شیءگرا می‌باشند را استفاده نموده‌اند. در یکی از تجربیات [۶۵]،از یک مدل که هم جداسازی فیزکی و هم مجازی (CIDE) دغدغه‌ها را پشتیبانی می‌کند، استفاده نمودند و به صورت صوری بازآرایی را در هر دو جهت تعریف نمودند تا هم ارزی آنها را نشان بدهد.

Alves [۶۵، ۶۶] و Calheiros [۱۹] یک خط تولید را توسط تکنیک‌های بازآرایی در حوزه بازی‌های تلفن همراه ایجاد نمودند. در روش آنها تغییرپذیری توسط ساختارهای جنبه‌گرا به کار گرفته شده‌اند. آنها یک گراف از دغدغه‌های تغییرپذیر را با استفاده از FEAT [۶۸] ایجاد نمودند. در [۶۵] و [۶۷] با استفاده از FLiP (یک ابزار پیاده‌سازی شده در Eclipse) دغدغه‌ها را بازآرایی نمودند. یکی از مؤلفه‌های این ابزار به نام FLiPEx، یک ابزار بازآرایی است که اجازه می‌دهد تا تغییرپذیری از کلاس‌های Java استخراج شده و با استفاده از یک ویزارد به جنبه‌ها تبدیل شوند.

FLiPEx قادر است تا به ابزار تجاری خط تولید pure::variants برای مدیریت ویژگی‌ها متصل شود. طراحی خط تولید نرم‌افزار بر مبنای برنامه‌نویسی جنبه‌گرا (به تنهایی یا در کنار دیگر روش‌ها مثل ویژگی‌گرا) فواید زیادی را نسبت به روش سنتی شیءگرا در بر دارد.

در هر حال استفاده از AspectJ به تنهایی دارای مشکلات متعددی می‌باشد [۶۴] که می‌توانند توسط ترکیب جنبه‌ها با دیگر مفاهیم (مؤلفه‌ها یا لایه‌های mixin) برطرف شوند. ابزارهایی مانند FLiPEx می‌توانند به بازآرایی جنبه‌گرای سیستم‌های قدیمی خط تولید نرم‌افزار کمک نمایند. روش جداسازی مجازی ویژگی‌ها همانند CIDE و پیاده‌سازی استاندارد شیءگرا سودمندی خود را در تجربیات صنعتی نشان داده است.

جداسازی فیزیکی ویژگی‌ها به معنای مشخص نمودن کاملاً شفاف تغییرپذیری‌ها می‌باشد. جداسازی دقیق ویژگی‌های متغیر با استفاده از AOP(AspectJ) یا FOP(Feature IDE) یا دیگر ابزارها بایستی بررسی شوند. البته شواهد نشان دهنده این است که روش FOP بهتر و مؤثرتر واقع شده است [۵۷]. یک فهرست از روش‌های بازآرایی AOP وجود دارد ولی بایستی در رابطه با روش‌ها و سیستم‌های پیچیده‌تر نیز روش‌هایی پیش‌بینی شوند [۵۷]. یک ارزیابی و مقایسه عمیق میان روش‌های AOP و FOP در [۶۹] ارائه شده است. این حوزه دانش هنوز یک حوزه باز و پویا می‌باشد و مطمئناً در آینده در این رابطه کارهای پیشرفته‌تری انجام خواهد شد.

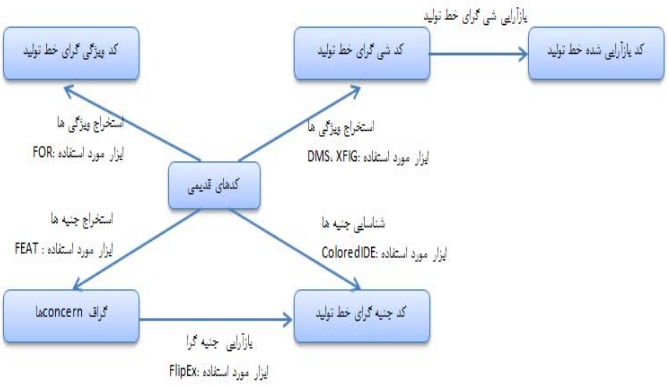
۵-۳- خلاصه‌ای از ابزارهای مهندسی مجدد کدهای قدیمی

در جدول ۵ و شکل ۲ ابزارهایی که جهت استخراج یک خط تولید از کدهای قدیمی می‌توانند مورد استفاده قرار گیرند، نشان داده شده‌اند. برخی از آنها جنبه‌گرا و برخی دیگر ویژگی‌گرایا شیءگرا می‌باشند. ابزارهای اولیه ایجاد شده اغلب در کشف ویژگی‌ها کاربرد دارند. ولی ابزارهای جدیدتر بیشتر بر روی بازآرایی خط تولید ایجاد شده تمرکز دارند و اغلب آنها متن باز یا Eclipse Plug-in می‌باشند.

برای مثال FLiPEx بخشی از یک ابزار برای بازآرایی کدها به ویژگی‌های فیزیکی است و CIDE نیز ابزاری برای علامت‌گذاری دستی کد جهت جداسازی مجازی ویژگی‌ها از یکدیگر است.

جدول ۵- خلاصه‌ای از ابزار های استفاده شده جهت مهندسی مجدد کدهای قدیمی

ابزار	کارکرداصلی	روش دسترسی	قابلیت ها
DMS [۷۹]	سیستم طراحی قابلیت نگهداری (مهندسی مجدد)	تجاری	ابزار خط تولید
Concept Explorer [۸۰]	تحلیل مفهومی صوری	رایگان	ابزار عمومی، نیاز به آماده سازی داده‌ها
CIDE [۸۱]	بازآرایی کد	متن باز	ویژگی‌ها با رنگ‌های مختلف نشان داده می‌شوند
XFIG [۸۲]	مهندسی مجدد، کشف ویژگی‌ها	متن باز	تحلیل مفهومی صوری
FLipEX [۸۳]	استخراج جنبه‌ها	EclipsePlug-in	تبدیل کد Java به AspectJ
XVCL [۸۴]	ترکیب خط تولید	متن باز	جاوا با قابلیت اجرای دستورات XVCL
FOR [۸۵]	استخراج ویژگی‌ها	دسترسی عمومی ندارد	ویژگی‌گرا



شکل۲- ارتباط مابین کدهای قدیمی و کدهای بازسازی شده شیءگرا، ویژگی‌گرا و جنبه‌گرا

۶- مقایسه و ارزیابی انواع روش‌های مهندسی مجدد

با توجه به مطالب ارائه شده در بالا، مشخص گردید که روش‌ها و ابزارهای متعددی جهت مهندسی مجدد سیستم‌های قدیمی در جهت ایجاد خط تولید نرم‌افزار ابداع و ارائه گردیده‌اند.

در خصوص ارزیابی انواع فرآیندهای پیشنهادی مهندسی مجدد می‌توان به این نکته اشاره نمود که هر کدام از این روش‌ها در حقیقت تکمیل کننده روش ارائه شده ماقبل خود بوده و به صورت جداگانه پاسخگوی بخشی از نیازهای این حوزه تحقیقاتی می‌باشند. لذا مقایسه آنها با یکدیگر کاری صحیح نمی‌باشد. برای نمونه در روش MAP که مؤسسه SEI ارائه نموده است بیشتر تمرکز بر روی مهندسی مجدد معماری سیستم‌های قدیمی در جهت تشکیل معماری خط تولید می‌باشد و این رویکرد در سایر روش‌ها مشاهده نمی‌گردد. همچنین، معمولاً مهندسی مجدد سیستم‌های قدیمی و ایجاد یک خط تولید نرم‌افزار در یک سازمان، باعث بهبود کلی می‌شود و این یک حقیقت انکارناپذیر است و نمی‌توان به جهت بهبود حاصل از تشکیل خط تولید در یک سازمان، روشی را کامل و بدون نقص اعلام نمود. از سویی دیگر، برای مستندسازی میزان افزایش کیفیت یا بهره‌وری یک شرکت پس از ایجاد خط تولید، بایستی قبل و بعد از مهندسی مجدد شاخص‌های اندازه‌گیری مشخص شده باشند (همانند سطح چهارم مدل CMMI مؤسسه SEI) و متأسفانه اینکار در اکثر شرکت‌ها صورت نمی‌پذیرد و خیلی معمول نمی‌باشد.

در رابطه با مقایسه روش‌ها و ابزارهای مهندسی مجدد مدل‌ها و کدهای سیستم‌های قدیمی، بایستی میزان بهبود ناشی از اجرای هر یک را بر روی یک نمونه مورد مطالعه محاسبه نمود. در برخی از مقالات شاخص‌هایی جهت ارزیابی میزان بهبود ایجاد شده ناشی از فعالیت مهندسی مجدد در یک سیستم ارائه شده است که خلاصه‌ای از آنها در جدول ۶ آمده است. برای مثال Kolb [۶] یک مطالعه موردی صنعتی را ارائه نموده است که در آن از روش PuLSE که مؤسسه IESE ارائه نموده است، جهت مهندسی مجدد مؤلفه‌های یک نرم‌افزار قدیمی استفاده شده است. در این مقاله، تحلیل مؤلفه‌های C++ موجود با استفاده از شاخص‌ها انجام و جنبه‌های مختلف کیفی مانند قابلیت انعطاف، قابلیت استفاده مجدد، و قابلیت نگهداری مورد بررسی قرار گرفته است. از طریق اندازه‌گیری شاخص‌های مربوط به کد برنامه، میزان بهبود حاصل از اعمال مهندسی مجدد به صورت خودکار یا دستی (تغییر نام فایل‌ها یا توابع، تقسیم فایل‌های طولانی) محاسبه و مورد مقایسه قرار گرفته‌اند. در کل کارهای فراوانی در رابطه با شاخص‌های ارزیابی مهندسی مجدد وجود دارند که از جمله آنها می‌توان به موارد ذیل اشاره نمود:

۱) نرمال سازی و تعیین سطح سازمانی شاخص‌ها.

۲) مقایسه میان خطوط تولید نرم‌افزار ایجاد شده بر مبنای زبان‌های کلاسیک شیءگرا و زبان‌های ویژگی‌گرا و جنبه‌گرا با استفاده از این شاخص‌ها.

۷- چالش‌های باقیمانده و فرصت‌های مطالعاتی

در جدول ۷ خلاصه‌ای از فرصت‌های مطالعاتی که در زمینه توسعه مهندسی مجدد سیستم‌های قدیمی موجود می‌باشد، قابل مشاهده است. در ستون اول این جدول زمینه‌های کاری مختلف در این حوزه لیست شده‌اند. در ستون دوم مهمترین فعالیت‌هایی که تاکنون در هر زمینه صورت گرفته به صورت خلاصه آورده شده است. در ستون سوم مباحث باز و فرصت‌های مطالعاتی در هر زمینه ذکر شده است. در ادامه راجع به هر یک از فرصت‌های مطالعاتی در این حوزه توضیحاتی ارائه خواهد شد:

- **متدولوژی و فرآیند مهندسی مجدد.** روش‌ها، ابزار و شاخص‌های ارزیابی مهندسی مجدد، نقطه شروع خوبی برای ارزیابی کدهای قدیمی و جستجوی ویژگی‌ها در میان بخش‌های مختلف سیستم‌های قدیمی به شمار می‌روند. فعالیت در زمینه مفاهیم اولیه خط تولید و فرآیند تکامل سیستم‌ها یکی از مباحثی است که فعالیت‌های بسیاری بر روی آن صورت گرفته است. لکن اخیراً گرایش بیشتر به سمت توسعه سریع وجود دارد. تأثیر فرآیند توسعه سریع بر فرآیند مهندسی مجدد سیستم‌های قدیمی می‌تواند یکی از مباحثی باشد که در آینده بیشتر می‌تواند مورد توجه قرار گیرد و هنوز جزئیات فراوانی راجع به این موضوع وجود دارد که به آنها پرداخته نشده است و به عنوان یک مبحث باز همچنان وجود دارد. تاثیر کلی ایجاد خط تولید و تکامل

سیستم‌های قدیمی بر روی سازمان‌ها از جمله تأثیر بر روی بهره‌وری، زمان رسیدن به بازار، کیفیت نهایی محصول، دقیقاً توسط افراد درگیر در این حوزه اندازه‌گیری نشده‌اند و هنوز به عنوان یک چالش در این حوزه وجود دارند.

- **استخراج مدل ویژگی‌ها**. با بررسی‌های صورت گرفته در این حوزه مشخص گردید، تکنیک‌های زیادی برای تشخیص ویژگی‌ها در سیستم‌های قدیمی وجود دارند. برای برخی از این ویژگی‌ها، ابزارهای متفاوتی نیز ایجاد گردیده‌اند. اما هیچیک از این تکنیک‌ها و ابزارها بر روی استخراج مدل ویژگی و تحلیل تغییرپذیری کار ننموده‌اند. یکی از مسایلی که وجود دارد این است که نه تنها ویژگی‌ها را شناسایی کنیم، بلکه وابستگی میان آنها و سلسله مراتبشان در کد اصلی برنامه را نیز شناسایی نماییم. کامل نمودن تکنیک‌ها و ابزارهای موجود جهت ایجاد قابلیت استخراج مدل ویژگی و بهینه‌سازی آن، یکی از مباحث باز در این حوزه می‌باشد.
- **استخراج ویژگی‌ها از کدهای قدیمی**. شناسایی و استخراج مؤلفه‌های مناسب جهت تبدیل شدن به موجودیت‌های خط تولید تقریباً توسط افراد مختلف به صورت کامل مورد بررسی قرار گرفته شده است. کارهای فراوانی برای اندازه‌گیری میزان مناسب بودن مؤلفه‌ها برای استفاده مجدد یا مهندسی مجدد و تبدیل شدن به موجودیت‌های خط تولید صورت گرفته است. لکن انتخاب بهترین زبان برنامه‌نویسی هدف برای مهندسی مجدد کدها یا بازآرایی هنوز به عنوان یک مبحث باز مطرح می‌باشد. بعضی از تحقیقات، ادعا می‌نمایند که جنبه‌گرایی جهت جداسازی دغدغه‌ها، بهترین روش برای پیاده‌سازی ویژگی‌ها می‌باشد. برخی دیگر، مشکلاتی را در استفاده از زبان‌های جنبه‌گرا کشف نموده‌اند و پیشنهاد کرده‌اند که از تلفیق روش‌های مختلف استفاده شود.

جدول ۶- خلاصه‌ای از شاخص‌های استفاده شده جهت ارزیابی کیفیت و مورد فایده بودن مهندسی مجدد

شاخص	تعریف شاخص
COC [۸۶]	میزان اتصال کلاس‌ها و اشیا
MPC [۸۶]	اتصال از طریق انتقال پیام
DAC [۸۶]	اتصال داده‌ها
NMPUB [۸۷]	تعداد متدهای عمومی پیاده‌سازی شده توسط یک کلاس
WMC [۸۷]	میزان پیچیدگی در یک کلاس
NOC [۸۷]	تعداد فرزندان و نوه‌های یک کلاس
DIT [۸۷]	سطح یک کلاس در یک ارث‌بری سلسله مراتبی
OCAEC [۸۷]	تعداد دفعاتی که یک کلاس به عنوان یک صفت در یک کلاس دیگر استفاده شده است
CALLS_IN [۸۷]	تعداد دفعاتی که متدهای یک کلاس توسط دیگر کلاس‌ها فراخوانی شده‌اند
LCOM [۸۸]	میزان چسبندگی- تعداد مجموعه متدهایی که به یک صفت خاص دسترسی دارند
LOC [۸۸]	تعداد خطوط کد پیمانه یا تابع
LOMC [۸۸]	تعداد خطوط قابل نگهداری کد پیمانه یا تابع (بدون در نظر گرفت خطوط خالی یا توضیحات)
FAN-IN [۸۸]	تعداد توابعی که که یک تابع را فراخوانی می‌نمایند
FAN-OUT [۸۸]	تعداد توابعی که درون یک تابع فراخوانی می‌شوند
CumulatedData [۸۸]	تعداد کل اعضا در یک ساختار متراکم
Associations [۸۸]	میزان وابستگی با دیگر ساختارها
Max Nesting [۸۸]	حداکثر سطح تو در تویی ساختارهای کنترلی
SOC [۸۹]	میزان مشترک بودن
IOC [۸۹]	تأثیر مشترک بودن
PrR [۸۹]	محصولات مرتبط با قابلیت استفاده مجدد
IPrR [۸۹]	اثر محصولات با قابلیت استفاده مجدد
RB [۸۹]	فایده دوباره استفاده نمودن
SD [۹۰]	درجه پراکندگی (از شاخص‌های دغدغه‌های سراسری)
TG [۹۰]	درجه درهم‌تنیدگی[۳۳] (از شاخص‌های دغدغه‌های سراسری)
StartMethod [۹۰]	شاخص‌های محلی‌سازی
EndMethod [۹۰]	شاخص‌های محلی‌سازی
BeforeReturn [۹۰]	شاخص‌های محلی‌سازی
NestedStatement [۹۰]	شاخص‌های محلی‌سازی

جدول ۷- خلاصه‌ای از فعالیت‌های انجام شده و فرصت‌های مطالعاتی باقیمانده در زمینه‌مهندسی مجدد

حوزه	فعالیت صورت گرفته	چالش‌ها و فرصت‌های مطالعاتی باقیمانده
فرآیند و متدولوژی	• ابداع روش‌های کلاسیک توسط موسسات SEI و IESE • تجربیات صنعتی فراوان • ابداع متدهای OAR، MAP، RE-PLACE • ابداع توسعه چابک فرآیند مهندسی مجدد	• ارزیابی سیستماتیک میزان بهبود صورت گرفته در فرآیند و محصول نهایی • مقایسه و ارزیابی میان فرآیند توسعه کلاسیک و چابک
استخراج مدل	• استفاده از ابزار مهندسی مجدد جهت شناخت و استخراج ویژگی‌ها از کدهای قدیمی • ایجاد ابزار RECON، FEAT، CIDE	• کامل نمودن ابزار با رویکرد خط تولید نرم‌افزار (استخراج مدل ویژگی‌ها علاوه بر استخراج ویژگی‌ها) • انتخاب بهترین زبان هدف برای مهندسی مجدد کدها
استخراج ویژگی‌ها از کدهای قدیمی با رویکرد برنامه‌نویسی شیءگرای	• استفاده از تکنیک‌ها و ابزار مهندسی مجدد توسط زبان‌های شیءگرایی مانند JAVA و C++ • ابزاری مانند DMS، XVCL و XFIG	• ارزیابی و مقایسه روش‌های جداسازی فیزیکی و مجازی ویژگی‌ها • ارزیابی و مقایسه روش‌های جنبه‌گرا (AspectJ) با روش‌های ویژگی‌گرا (Java و C++)
استخراج ویژگی‌ها از کدهای قدیمی با رویکرد برنامه‌نویسی ویژگی‌گرا	• جداسازی کدهای Java (جداسازی فیزیکی ویژگی‌ها) • بازآرایی ویژگی‌گرا • ابداع ابزار رایگان Ahead و FeatureIDE	
استخراج ویژگی‌ها از کدهای قدیمی با رویکرد برنامه‌نویسی جنبه‌گرا	• بازآرایی جنبه‌گرا • ابداع ابزاری مانند FLiPEx plug-in و CIDE	
ارزیابی و مقایسه انواع روش‌های مهندسی مجدد	• ابداع شاخص‌های ارزیابی و کاربردی در زمینه برنامه نویس شیءگرا و جنبه‌گرا	• نرمال سازی و دسته‌بندی شاخص‌ها • نیاز به داده‌های جمع‌آوری شده بیشتر (قبل و بعد از مهندسی مجدد) برای ارزیابی

از طرف دیگر، در صورتی که از زبان‌های برنامه‌نویسی مثل Java یا C++ استفاده شود، باعث می‌شود تا خط تولید ایجاد شده مورد استفاده شرکت‌های بیشتری قرار گیرد (به دلیل وسعت استفاده از این دو زبان برنامه‌نویسی). پس مشاهده می‌نماییم که هنوز یک چالش بزرگ در این زمینه وجود دارد که می‌تواند مورد بررسی قرار گیرد.

- **ارزیابی و تحلیل تجربی**. تا به امروز مطالعات زیادی در زمینه جمع‌آوری داده‌های تجربی جهت ارزیابی میزان کیفی موجودیت‌های خروجی ناشی از مهندسی مجدد سیستم‌های قدیمی صورت پذیرفته است. در اغلب آنها شاخص‌های مختلفی برای ارزیابی میزان مناسب بودن مؤلفه‌های موجود برای تبدیل شدن به موجودیت‌های خط تولید معرفی شده‌اند. مجموعه‌ای از ابزارها نیز در این زمینه موجود هستند که هیچیک از آنها با ابزار خاصی از خط تولید که جهت شناسایی مؤلفه‌ها به کار برده می‌شود، یکپارچه و مرتبط نیستند. این مبحث نیز به عنوان یک فرصت مطالعاتی باز و چالش باقیمانده می‌تواند مطرح شود. همچنین در زمینه استانداردسازی این شاخص‌ها نیز هنوز فعالیتی صورت نگرفته است. در خصوص ارزیابی روش‌های ابداع شده مهندسی مجدد و مقایسه توانمندی‌ها و کاستی‌های هر یک نسبت به سایر روش‌ها نیز تابه حال پژوهشی صورت نپذیرفته است و به عنوان یک فرصت مطالعاتی باقیمانده است.

۸- نتیجه‌گیری

در این مقاله مروری بر روی فعالیت‌های انجام شده در زمینه مهندسی مجدد سیستم‌های قدیمی با رویکرد ایجاد خط تولید نرم‌افزار صورت پذیرفته است. در اینجا فعالیت‌های انجام شده به سه دسته فرآیندی، استخراج مدل و استخراج کد دسته‌بندی گردیده‌اند. در رابطه با هر یک از این سه حوزه نیز روش‌ها و ابزارهای ابداع شده دسته‌بندی و گردآوری شده است. در این مقاله هدف اصلی ما در ابتدا محدود کردن حوزه تحقیقاتی این زمینه و در نهایت مشخص نمودن روش‌هایی است که در زمینه مهندسی مجدد سیستم‌های قدیمی تا به امروز ابداع و معرفی شده‌اند. همچنین سعی شده است، روش‌هایی که به خصوص در صنعت کاربرد عملی داشته‌اند بیشتر مورد توجه قرار گرفته و ابزارهای خاصی که در این زمینه توسط مقالات مختلف معرفی شده‌اند، گردآوری شوند.

به علاوه در این مقاله سعی شده است تا چالش‌ها و فرصت‌های مطالعاتی باقیمانده در این زمینه را نیز بیان نماییم. برطبق بررسی‌های صورت گرفته مشخص گردید که تعداد مقالاتی که از روش‌های صوری جهت اثبات روش خود بهره برده‌اند بسیار اندک بوده و استفاده از این روش می‌تواند به صورت یک فرصت مطالعاتی باز مطرح باشد. همچنین ارزیابی هر یک از روش‌ها و ابزارهای معرفی شده در این مقاله نیز همچنان می‌تواند در آینده مورد توجه قرار گیرد. در نهایت با توجه به مطالب گردآوری شده مشخص گردید که رویکرد مهندسی مجدد سیستم‌های قدیمی جهت ایجاد خط تولید نرم‌افزار، رویکردی پویا و رو به رشد بوده و دارای مسایل و چالش‌های فراوانی می‌باشد که می‌تواند مورد توجه محققین این حوزه قرار گیرند.

مراجع

[1] F. van der Linden, K. Schmid, and E. Rommes, "Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering," *Springer*, pp. 333, 2003.

[2] SEI, "A Framework for Software Product Line Practice, Version 5. 0," http://www.sei.cmu.edu/productlines/frame_report/frame_bib.htm, November 2011.

[3] J. Bosch, *Design and Use of Software Architectures. Adopting and Evolving a Product-Line Approach*, Addison-Wesley, 2000.

[4] B. A. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," *Proc, Int'l Conf. Software Engineering*, pp. 273-281, 2004.

[5] J. Bayer, J. -F. Girard, M. Würthner, J. M. DeBaud, and M. Apel, "Transitioning Legacy Assets to a Product Line Architecture," *Journal of Software Engineering*, vol. 16, no. 87, pp. 446-463, 1997.

[6] R. Kolb, D. Muthig, T. Patzke, and K. Yamauchi, "Refactoring a legacy component for reuse in a software product line: a case study," *Journal of Software and Maintainance*, vol. 18, no. 2, pp. 109-132, 2006.

[7] M. Pinzger, "Architecture recovery for product families," *Proc, Int'l Conf. Software Product-Family Engineering*, pp. 332-351, 2004.

[8] I. John, "Integrating Legacy Documentation Assets into a Product Line," *Proc, Int'l Workshop on Software Product-Family Engineering*, pp. 113-124, 2002.

[9] M. Pinzger, "Architecture recovery for product families," *Proc, Int'l Conf. Software Product-Family Engineering*, pp. 332-351, 2004.

[10] J. M. De Baud, and J. F. Girard, "The Relation between the Product Line Development Entry Points and Reengineering," *Proc, Int'l Workshop on Development and Evolution of Software Architectures for Product Families*, pp. 132-139, 1998.

[11] N. Weiderman, J. Bergey, D. Smith, and S. Tilley, "Can Legacy Systems Beget Product Lines," *Proc, Int'l Workshop of ESPRIT ARES*, pp. 22-24, 1998.

[12] W. L. Scherlis, "Structural Views, Structural Evolution, and Product Families," *Proc, Int'l Conf. Development and Evolution of Software Architectures for Product Families*, pp. 235-240, 1998.

[13] C. Stoermer, and L. O'Brien, "MAP- Mining Architectures for Product Line Evaluations," *Proc, IEEE/IFIP Int'l Conf. Software Architecture*, pp. 35-44, 2011.

[14] L. O'Brien, F. Hansen, R. Seacord, and D. Smith, "Mining and managing software assets," *Proc, Int'l Conf. Software Technology and Engineering Practice*, pp. 82-90, 2002.

[15] R. Kazman, and S. J. Carrière, "Playing detective: reconstructing software architecture from available evidence," *Journal of Automated Software Engineering*, pp. 107-138, 1999.

[16] J. Knodel, and et al., "Asset Recovery and Incorporation into Product Lines," *Proc, Int'l Conf. Reverse Engineering*, pp. 120, 2005.

[17] J. Knodel, and D. Muthig, "Analyzing the Product Line Adequacy of Existing Components," *Proc, Int'l Workshop on Reengineering Towards Product Lines*, pp. 67-69, 2005.

[18] G. Raghavan, "Improving software quality in product families through systematic reengineering," *Proc, European Int'l Conf. Software Quality*, pp. 90-99, 2002.

[19] H. P. Breivold, S. Larsson, and R. Land, "Migrating Industrial Systems towards Software Product Lines: Experiences and Observationsthrough Case Studies," *Proc, Euromicro Int'l Conf. Software Engineering and Advanced Applications*, pp. 232-239, 2008.

[20] K. C. Kang, M. Kim, J. Lee, and B. Kim, "Feature-oriented re-engineering of legacy systems into product line assets-a case study," *Proc, Int'l Conf. Software Product Lines*, pp. 45-56, 2005.

[21] G. Jun, D. Eryu, and L. Bin, "Feature-oriented re-engineering using product line approach," *Proc, Int'l Conf. Information Science and Engineering*, pp. 255-260, 2010.

[22] H. Lee, H. Choi, K. Kang, D. Kim, and Z. Lee, "Experience Report on Using a Domain Model-Based Extractive Approach to Software Product Line Asset Development," *Journal of Formal Foundations of Reuse and Domain Engineering*, vol. 57, no. 91, pp. 137-149, 2009.

[23] C. H. B. Hansen, "Component reengineering workshops: A low-cost approach for assessing specific reengineering costs across product line," *Proc, European Int'l Conf. Software Maintenance and Reengineering*, vol. 8, no. 2, pp. 154-162, 2004.

[24] J. Bosch, and P. M. Bosch-Sijtsema, "Introducing agile customer-centered development in a legacy software product line," *Journal of Software, Practice and Experience*, vol. 57, no. 3, pp. 871-882, 2011.

[25] Y. Ghanam, and F. Maurer, "Extreme Product Line Engineering-Refactoring for Variability: A Test-Driven Approach," *Proc, Int'l Conf. Agile Processes in Software Engineering and Extreme Programming*, pp. 43-57, 2010.

[26] A. Hubaux, P. Heymans, and D. Benavides, "Variability modelling challenges from the trenches of an open source product line reengineering project," *Proc, Int'l Conf. Software Product Line*, pp. 55-64, 2008.

[27] A. Hubaux, P. Heymans, and H. Unphon, "Separating variability concerns in a product line re-engineering project," *Proc, ACM Int'l workshop on AOSD*, pp. 1-8, 2008.

[28] J. Bartholdt, and D. Becker, "Re-engineering of a hierarchical product line," *Proc, Int'l Conf. Software Product Line*, pp. 232-240, 2011.

[29] Y. Wu, Y. Yang, X. Peng, C. Qiu, and W. Zhao, "Recovering object-oriented framework for software product line reengineering," *Proc, Int'l Conf. Top Productivity through Software Reuse*, pp. 119-134, 2011.

[30] G. Zhang, L. Shen, X. Peng, Z. Xing, and W. Zhao, "Incremental and Iterative Reengineering towards Software Product Line: An Industrial Case Study," *Proc, Int'l Conf. Software Maintenance*, pp. 418-427, 2011.

finding and describing concerns using structural program dependencies," *Proc, Int'l Conf. Software Engineering*, pp. 406-416, 2002.

[46] B. Graaf, S. Weber, and A. V. Deursen, "Migrating Supervisory Control Architectures Using Model Transformations," *Proc, Int'l Conf. Software Maintenance and Reengineering*, pp. 151-160, 2006.

[47] "AMMA, ATL Development Tools," http://www. eclipse.org/m2m/atl/doc/, November 2012.

[48] C. Kustner, M. Kuhlemann, and D. S. Batory, "Automatic Feature Oriented refactoring of legacy applications," *Proc, Int'l Workshop on Refactoring Tools*, pp. 62-63, 2007.

[49] W. Zhang, S. Jarzabek, N. Loughran, and A. Rashid, "Reengineering a PC-based System into the Mobile Device Product Line," *Proc, Int'l Workshop on Principles of Software Evolution*, pp. 149-160, 2002.

[50] D. Simon, and T. Eisenbarth, "Evolutionary Introduction of Software Product Lines," *Proc, Int'l Conf. Software Product Lines*, vol. 23, no. 79, pp. 1-14, 2002.

[51] R. L. Akers, I. D. Baxter, M. Mehlich, B. J. Ellis, and K. R. Luecke, "Case Study: Re-Engineering C++ Component Models via Automatic Program Transformation," *Journal of Information and Software Technology*, vol. 49, no. 3, pp. 275-291, 2007.

[52] I. Olszak, and B. N. Jensen, "Remodularizing Java Programs for Comprehension of Features," *Proc, Int'l Workshop on Feature-Oriented Software Development*, pp. 9, 2009.

[53] C. Chiang, and R. Y. Lee, "Developing Tools for Reverse Engineering in a Software Product-Line Architecture," *Proc, IEEE Int'l Conf. Information Reuse and Integration*, pp. 42-47, 2004.

[54] D. Batory, J. N. Sarvela, and A. Rauschmayer, "Scaling Stepwise Refinement," *IEEE Trans. Software Engineering*, pp. 355-371, 2004.

[55] J. Liu, D. Batory, and C. Lengauer, "Feature oriented refactoring of legacy applications," *Proc, Int'l Conf. Software Engineering*, pp. 112, 2006.

[56] Trujillo, D. Batory, and O. Diaz, "Feature Refactoring a Multi-Representation Program into a Product Line," *Proc, Int'l Conf. Generative Programming and Component Engineering*, pp. 191-200, 2006.

[57] R. E. Lopez-Herrejon, L. Montalvillo-Mendizabal, and A. Egyed, "FromRequirements to Features: An Exploratory Study of Feature Oriented Refactoring," *Proc, Int'l Conf. Software Product Line*, pp. 181-190, 2011.

[58] Database-Workgroup, "FeatureIde: Eclipse-Plugin for Feature-Oriented Software Development," http: //www. fosd. defide, November 2012.

[31] M. Almorsy, "SMURF: Supporting Multi-tenancy Using Re-aspects Framework," *Proc, Int'l Conf. Engineering of Complex Computer Systems*, pp. 361-370, 2012.

[32] M. A. Ramos, "Reengineering legacy systems towards system of systems development," *Proc, Int'l Conf. Information Reuse and Integration*, pp. 624-630, 2012.

[33] I. John, "Capturing Product Line Information from Legacy User Documentation," *Proc, Int'l Conf. Software Product Lines, Springer*, pp. 127-159, 2006.

[34] I. John, "Integrating Legacy Documentation Assets into a Product Line," *Proc, Int'l Workshop on Software Product-Family Engineering*, pp. 113-124, 2002.

[35] J. Knodel, and et al., "Asset Recovery and Incorporation into Product Lines," *Proc, Int'l Conf. Reverse Engineering*, pp. 120, 2005.

[36] I. Pashov, and M. Riebisch, "Using feature modeling for program comprehension and software architecture recovery," *Proc, IEEE Int'l Conf. Engineering of Computer-Based Systems*, pp. 406-417, 2004.

[37] K. Kim, H. Kim, and W. Kim, "Building Software Product Line from the Legacy Systems, Experience in the Digital Audio and Video Domain," *Proc, Int'l Conf. Software Product Line*, pp. 171-180, 2007.

[38] S. A. Ajila, "Reusing base product features to develop product line architecture," *Proc, IEEE Int'l Conf. Information Reuse and Integration*, pp. 288-293, 2005.

[39] J. M. Conejero, J. Herndez, and E. Jurado, "Early Analysis of Modularity in Software Product Lines," *Proc, Int'l Conf. Software Engineering and Knowledge Engineering*, pp. 721-736, 2009.

[40] R. Stoiber, S. Fricker, M. Jehle, and M. Glinz, "Feature Unweaving: Refactoring Software Requirements Specifications into Software Product Lines," *Proc, IEEE Int'l Conf. Requirements Engineering*, pp. 403-404, 2010.

[41] M. Glinz, "Object-oriented modeling with ADORA," *Journal of Information Systems*, vol. 27, no. 3, pp. 425-444, 2002.

[42] M. Glinz, "Supporting stepwise, incremental product derivation in product line requirements engineering," *Proc, Int'l Conf. VaMoS*, pp. 77–84, 2010.

[43] F. Chen, S. Li, H. Yang, C. H. Wang, and W. Cheng-Chung Chu, "Feature analysis for service-oriented reengineering," *Proc, Int'l Conf. Asia-Pacific Software Engineering*, pp. 201-208, 2005.

[44] N. Wilde, M. Buckellew, H. Page, V. Rajlich, and L. Pounds, "A comparison of methods for locating features in legacy software," *Journal of Systems and Software*, vol. 65, no. 5, pp. 105-114, 2003.

[45] M. R. Robillard, and G. C. Murphy, "Concern graphs:

[59] Y. Ghanam, and F. Maurer, "Extreme Product Line Engineering–Refactoring for Variability: A Test-Driven Approach," *Proc, Int'l Conf. Agile Processesin Software Engineering and Extreme Programming*, pp. 43-57, 2010.

[60] A. Dabholkar, and A. Gokhale, "Middleware Specialization for Product-Lines Using Feature-Oriented Reverse Engineering," *Proc, Int'l Conf. Information Technology*, pp. 696-701, 2010.

[61] Y. Xue, "Reengineering legacy software products into software product line based on automatic variability analysis," *Proc, Int'l Conf. Software Engineering*, pp. 1114-1117, 2011.

[62] A. Lozano, "AnOverview of Techniques for Detecting Software Variability Concepts in Source Code," *Proc, Int'l Conf. Advances in Conceptual Modeling*, pp. 141-150, 2011.

[63] C. Kastner, M. Kuhlemann, and D. S. Batory, "Automatic Feature Oriented refactoring of legacy applications," *Proc, Int'l Workshop on Refactoring Tools*, pp. 62-63, 2007.

[64] C. Kastner, S. Apel, and D. Batory, "A Case Study Implementing Features Using AspectJ," *Proc, Int'l Conf. Software Product Lines*, pp. 223-232, 2007.

[65] V. Alves, F. Calheiros, V. Nepomuceno, A. Menezes, S. Soares, and P. Borba, "FLiP: Managing Software Product Line Extraction and Reaction with Aspects," *Proc, Int'l Conf. Software Product Lines*, pp. 354-354, 2008.

[66] V. Alves, P. M. Jr, L. Cole, P. Borba, and G. Ramalho, "Extracting and Evolving Mobile Games Product Lines," *Proc, Int'l Conf. Software Product lines, Springer Verlag*, pp. 34-44, 2005.

[67] F. Calheiros, P. Borba, S. Soares, and V. Nepomuceno, "Product Line Variability Refactoring Tool," *Proc, Int'l Workshop on Refactoring Tools*, pp. 33-34, 2007.

[68] M. R. Robillard, and G. C. Murphy, "Concern Graphs: Finding and Describing Concerns Using Structural Program Dependencies," *Proc, Int'l Conf. Software Engineering*, pp. 406-416, 2002.

[69] R. E. Lopez-Herrejon, L. Montalvillo-Mendizabal, and A. Egyed, "From Requirements to Features: An Exploratory Study of Feature-Oriented Refactoring," *Proc, Int'l Conf. Software Product Line*, pp. 181-190, 2011.

[70] D. B. Smith, L. O'Brien, and J. Bergey, "Using the Options Analysis for Reengineering (OAR) Method for Mining Components for a Product Line," *Proc, Int'l Conf. Software Product Lines*, pp. 316-327, 2002.

[71] "AMMA, ATL Development Tools," http://www.eclipse.org/m2m/atl/doc/, November 2012.

[72] M. Glinz, "Object-oriented modeling with ADORA," *Journal of Information Systems*, vol. 27, no. 2, pp. 425-444, 2002.

[73] N. Wilde, M. Buckellew, H. Page, V. Rajlich, and L. Pounds, "A comparison of methods for locating features in legacy software," *Journal of Systems and Software*, vol. 65, no. 5, pp. 105-114, 2003.

[74] M. R. Robillard, and G. C. Murphy, "Concern graphs: finding and describing concerns using structural program dependencies," *Proc, Int'l Conf. Software Engineering*, pp. 406-416, 2002.

[75] C. Kustner, M. Kuhlemann, and D. S. Batory, "Automatic Feature Oriented refactoring of legacy applications," *Proc, Int'l* Workshop on Refactoring Tools, pp. 62-63, 2007.

[76] R. Kazman, and S. J. Carrière, "Playing detective: reconstructing software architecture from available evidence," *Journal of Automated Software Engineering*, pp. 107-138, 1999.

[77] C. Kastner, S. Apel, and M. Kuhlemann, "A Model of Reactoring Physically and Virtually Separated Features," *Proc, ACM Int'l Conf. Generative Programming and Component Engineering*, pp. 157-166, 2009.

[78] L. P. Tizzei, and J. Lee, "An Aspect-Oriented View to Support Feature-oriented Reengineering," *Proc, Int'l Workshop on Aspect-Oriented Modeling*, pp. 10-12, 2010.

[79] R. L. Akers, I. D. Baxter, M. Mehlich, B. J. Ellis, and K. R. Luecke, "Case Study: Re-engineering C++ Component Models via Automatic Program Transformation," *Journal of Information and Software Technology*, vol. 49, no. 3, pp. 275-291, 2007.

[80] B. Graaf, S. Weber, and A. V. Deursen, "Migrating Supervisory Control Architectures Using Model Transformations," *Proc, European Conf. Software Maintenance and Reengineering*, pp. 151-160, 2006.

[81] C. Kastner, M. Kuhlemann, and D. S. Batory, "Automatic Feature Oriented Refactoring of Legacy Applications," *Proc, Int'l Workshop on Refactoring Tools*, pp. 62-63, 2007.

[82] D. Simon, and T. Eisenbarth, "Evolutionary Introduction of Software Product Lines," *Journal of Software Product Lines*, vol. 23, no. 79, pp. 1-14, 2002.

[83] V. Alves, and F. Calheiros, V. Nepomuceno, A. Menezes, S. Soares, and P. Borba, "FLiP: Managing Software Product Line Extraction and Reaction with Aspects," *Proc, Int'l Conf. Software Product Line*, pp. 354-354, 2008.

[84] W. Zhang, S. Jarzabek, N. Loughran, and A. Rashid, "Reengineering a PC-based System into the Mobile Device Product Line," *Proc, Int'l Workshop on Principles of Software Evolution*, pp. 149-160, 2003.

[85] J. Liu, D. Batory, and C. Lengauer, "Feature-Oriented Refactoring of Legacy Applications," *Proc, Int'l Conf. Software Engineering*, p. 112, 2006.

[86] J. M. Conejero, J. Hernandez, and E. Jurado, "Early Analysis of Modularity in Software Product Lines," *Proc, Int'l Conf. Software Engineering and Knowledge Engineering*, pp. 721-736, 2009.

[87] D. Ganesan, and J. Knodel, "Identifying Domain-Specific Reusable Components from Existing OO Systems to Support Product line Migration," *Proc, Int'l Workshop on Reengineering towards Product Lines*, pp. 12-14, 2005.

[88] R. Kolb, D. Muthig, T. Patzke, and K. Yamauchi, "A Case Study in Refactoring a Legacy Component for Reuse in a Product Line," *Proc, IEEE Int'l Conf. Software Maintenance*, pp. 369-378, 2005.

[89] C. Berger, H. Rendel, and B. Rumpe, "Measuring the Ability to Form a Product Line from Existing Products," *Proc, Int'l Conf. VAMOS*, pp. 151-154, 2010.

[90] M. V. Couto, M. T. Valente, and E. Figueiredo, "Extracting Software Product Lines: A Case Study Using Conditional Compilation," *Proc, European Conf. Software Maintenance and Reengineering*, pp. 191-200, 2011.

[91] E. Chikofsky, and J. Cross, "Reverse Engineering and Design Recovery: A Taxonomy," *IEEE Trans. Software*, vol.7, no. 1, pp. 13-18, 1990.

[92] N. H. Madhavji, J. Fernandez-Ramil, and D. Perry, *Software Evolution and Feedback: Theory and Practice*, John Wiley & Sons, 2006.

[93] C. L. Nehaniv, and P. Wernick, "Introduction to Software Evolvability," *Proc, Int'l Workshop on the Principles of Software Evolution*, pp. 47-49, 2007.

الهام درمنکی فراهانی مدارک کارشناسی و کارشناسی‌ارشد مهندسی نرم‌افزار خود را به ترتیب در سال‌های ۱۳۸۱ و ۱۳۸۳ از دانشگاه آزاد اسلامی واحد تهران مرکزی و تهران جنوب دریافت نموده است. در حال حاضر ایشان مشغول تحصیل در رشته مهندسی نرم‌افزار در مقطع دکتری در دانشگاه صنعتی شریف می‌باشد. از ایشان تاکنون مقالات متعددی در کنفرانس‌های داخلی و خارجی به چاپ رسیده است. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از مهندسی نرم‌افزار، خط تولید نرم‌افزار، باز مهندسی سیستم‌های نرم‌افزاری و معماری سازمانی.
آدرس پست‌الکترونیکی ایشان عبارت است از :

efarahani@ce.sharif.edu

عباس حیدرنوری مدرک دکترای خود در علوم کامپیوتر را در سال ۱۳۸۸ از دانشگاه واترلو کانادا و مدارک کارشناسی و کارشناسی‌ارشد در مهندسی نرم‌افزار خود را به ترتیب در سال‌های ۱۳۷۸ و ۱۳۸۰ از دانشگاه صنعتی شریف دریافت نموده است. بعد از اتمام دوره دکترا، ایشان به مدت یک سال به عنوان پژوهشگر پسا دکترا در دانشگاه لوگانو سوییس مشغول به انجام پژوهش بوده است. ایشان سپس به مدت یک سال به عنوان مهندس ارشد نرم‌افزار در زمینه تولید نرم‌افزارهای هوشمند همراه در شرکت Xtreme Labs Inc. در تورنتو کانادا مشغول به کار بوده است. دکتر حیدرنوری در حال حاضر به عنوان عضو هیأت علمی در دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف مشغول به فعالیت می‌باشد. زمینه‌های پژوهشی ایشان عبارتند از مهندسی نرم‌افزار، مهندسی معکوس و باز مهندسی سیستم‌های نرم‌افزاری، و تولید سیستم‌های نرم‌افزاری به روش مؤلفه‌گرا.
آدرس پست‌الکترونیکی ایشان عبارت است از :

heydarnoori@sharif.edu

جعفر حبیبی کارشناسی مهندسی کامپیوتر و کارشناسی‌ارشد مهندسی صنایع خود را به ترتیب در سال‌های ۱۳۵۹ و ۱۳۶۷ از مدرسه عالی کامپیوتر و دانشگاه تربیت مدرس و دکتری در علوم کامپیوتر را از دانشگاه منچستر انگلستان در سال ۱۳۷۸ دریافت نمود. وی هم‌اکنون عضو هیأت علمی و رئیس دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف و رئیس انجمن کامپیوتر ایران می‌باشد. زمینه‌های پژوهشی ایشان در مهندسی کامپیوتر عبارتند از: ارزیابی کارایی سیستم‌های کامپیوتری، مهندسی نرم‌افزار، شبکه‌های نظیر به نظیر، شبکه‌های اجتماعی و داده‌کاوی .
آدرس پست‌الکترونیکی ایشان عبارت است از :

jhabibi@sharif.edu

اطلاعات بررسی مقاله:

تاریخ ارسال: ۹۲/۱/۷
تاریخ اصلاح: ۹۲/۳/۱۸
تاریخ قبول شدن: ۹۲/۵/۳
نویسنده مرتبط: الهام درمنکی فراهانی، دانشگاه صنعتی شریف پردیس بین‌المللی کیش، کیش، ایران.

---

[1]Software Product Line (SPL)
[2]Reengineering
[3]Refactoring
[4]Model Transformation Techniques
[5]Code Transformation Techniques
[6]Migration
[7]Evolution
[8]Separation of Concerns
[9]Mining Architecture for Product Line
[10]Option Analysis for Reengineering
[11]Reengineering Enabled Product Line Architecture Creation and Evolution
[12]Architecture and Domain Oriented Reengineering
[13]Request Driven
[14]Boundary Classification
[15]Traceability Matrices
[16]Crosscutting Concern
[17]Aspect
[18]Software Reconnaissance
[19]Feature Exploration and Analysis Tool
[20]Atlas Transformation Language
[21]Meta Model
[22]Feature Model Extraction
[23]Object Oriented Programming
[24]Feature Oriented Programming
[25]Aspect Oriented Programming
[26]Embebeded
[27]Ahead Tool Suite
[28]Feature Oriented Refactoring
[29]Feature Oriented Reuse Engineering

[30]Middleware
[31]Make Project Creator
[32]Software Clones
[33]Tangling