

آزمون خودکار نرم افزارهای شی گرا با استفاده از موردهای کاربری و ضوابط OCL

سعید جلیلی^۱ آرش کربلایی^۲

۱- استادیار گروه کامپیوتر، دانشکده فنی و مهندسی - دانشگاه تربیت مدرس - تهران - ایران

SJalili@modares.ac.ir

۲- دانش آموخته کارشناسی ارشد گروه کامپیوتر، دانشکده فنی و مهندسی - دانشگاه تربیت مدرس - تهران - ایران

Arash.Karbalaei@gmail.com

چکیده: آزمون خودکار با استفاده از ضوابط نرم افزار از مهمترین زمینه های تحقیقاتی آزمون نرم افزار می باشد. اهمیت آزمون خودکار در کاهش هزینه های تولید نرم افزار و بالا بردن ضریب اطمینان به نرم افزار است. روش های موجود برای آزمون خودکار با تعریف معیارهای مورد نیاز که وابسته به کد و یا ضوابط نرم افزار می باشد، به دنبال تولید داده آزمون به منظور پوشش معیارها، اجرای آزمون روی برنامه و بررسی نتایج حاصل با اراکل مورد نظر هستند. در این مقاله روشی به منظور آزمون خودکار نرم افزارهای شی گرا بر پایه مورد کاربری نرم افزار ارائه شده است. معماری این روش از سه بخش تولید خودکار داده آزمون، آزمون خودکار برنامه و تحلیل نتایج تشکیل می شود. با تعریف سناریوی کاربری مورد نظر در قالب دیاگرام ترتیب مرحله طراحی و دریافت دیاگرام های کلاس، و فعالیت مورد نیاز برنامه در مرحله طراحی و کد برنامه، سیستم به تولید داده آزمون، اجرای آن و تحلیل نتایج آزمون می پردازد. از نتایج مهم این روش امکان تولید الگوی آزمون، تولید داده آزمون بر اساس سناریوهای کاربری، امکان انجام آزمون های مختلف و استفاده از ساختار یکپارچه برای مدل سازی می باشد.

کلمات کلیدی: آزمون خودکار نرم افزار، تولید داده آزمون، آزمون نرم افزار شی گرا، مورد کاربری، ضوابط OCL

تاریخ دریافت مقاله : ۱۳۸۴/۱۲/ ۲۹

تاریخ پذیرش مقاله : ۱۳۸۷/۶/۳۰

نام نویسنده ی مسئول : سعید جلیلی

نشانی نویسنده ی مسئول : تهران - تقاطع بزرگراه چمران و جلال آل احمد - دانشگاه تربیت مدرس - دانشکده ی فنی و مهندسی - گروه کامپیوتر

دوباره کامپایل شده و کد مجهز شده^۷ برنامه را ایجاد می کند. سپس داده های آزمون تولید شده از روی دستگاه معادلات بر روی این کد برنامه اجرا می شود و در نتیجه اطلاعات مربوط به وضعیت های فوق الذکر به تفکیک هر سناریوی کاربری در یک فایل کنترلی ذخیره می شوند. در نهایت بخش تحلیل آزمون با بررسی اطلاعات ذخیره شده، به تحلیل درستی اجرای آزمون می پردازد. در رویکرد پیشنهادی اراکل آزمون به عبارت دیگر همان تلفیق پیش شرطها و پس شرطهای متدهای همکاری کننده در سناریوی کاربری و نامتغیرهای کلاس های همکاری کننده در سناریو و روند فراخوانی متدها در سناریوی کاربری است.

در ادامه در بخش دوم، سابقه تحقیقات انجام شده در آزمون خودکار نرم افزار را مورد بررسی قرار می دهیم. در بخش سوم، روش پیشنهادی معرفی شده و اجزای آن به تفصیل تشریح می شوند. در بخش چهارم، روش پیشنهادی مورد ارزیابی قرار می گیرد. در بخش پنجم، تحقیقات آینده در این زمینه و همچنین نقاط قابل توسعه روش پیشنهادی معرفی می گردد. در بخش آخر نیز نتیجه گیری ارائه خواهد شد. پیوست شماره یک حاوی شرح یک نرم افزار نمونه می باشد که به منظور تشریح نحوه عمل روش خودکار آزمون، ارائه شده است. شرح کلی این نرم افزار نمونه به همراه دیاگرام های ورودی آن در این پیوست قرار دارد و نتایج اجرای روش بر روی نرم افزار نمونه در بخش های مختلف مقاله ارائه شده است. در این نرم افزار نمونه به دلیل کاهش صفحات مقاله از دیاگرام همکاری به جای دیاگرام ترتیب (که معادل یکدیگر هستند) استفاده شده است.

۲- روش های آزمون خودکار نرم افزارهای شی گرا

آزمون نرم افزارهای شی گرا دارای چهار سطح می باشد که این سطوح از پایین ترین واحد شی گرایی آغاز شده و به بالاترین واحد که سیستم می باشد ختم می شوند؛ این سطوح عبارتند از سطح الگوریتمی^۸ یا سطح متد^۹، سطح کلاس^{۱۰}، سطح خوشه^{۱۱} و سطح سیستم^{۱۲} [۳]. در سطح الگوریتمی متدهای کلاس به صورت جداگانه مورد آزمون قرار می گیرند. آزمون در سطح کلاس به کنترل درستی کوچکترین واحد شی گرایی یعنی کلاسها می پردازد. در این سطح کلیه کلاسها از لحاظ درستی کارکرد و برآوردن نیازهایی که طبق آنها تعریف شده اند بررسی می شوند. در سطح خوشه کلاسهای همکار در قالب مفهومی به نام خوشه گروه بندی شده، و روابط بین اشیای موجود در خوشه بررسی می شود. در سطح سیستم کل نرم افزار از لحاظ کارکرد مورد آزمون قرار می گیرد. برای این منظور روابط بین زیرسیستمها (خوشهها) در جهت انجام کارآیی مورد نیاز بررسی می گردد. آزمون سطح سیستم با ورودیها و خروجیهای قابل درک برای کاربران نهایی سیستم انجام می شود [۳].

از قدیمی ترین تحقیقات انجام شده در خودکار سازی آزمون نرم افزارهای شی گرا روش ASTOOT می باشد که در سال ۱۹۹۴

بطور کلی آزمون نرم افزار به دلیل صرف وقت زیاد و هزینه مالی بالایی که به روند تولید نرم افزار تحمیل می کند، معمولاً در فرآیند تولید نرم افزار مورد توجه کافی قرار نمی گیرد و همین امر از کیفیت نرم افزارهای تولیدی تا حد زیادی می کاهد. بر اساس تحقیقات آماری انجام شده، اگر آزمون نرم افزار به صورت کامل انجام شود در حدود ۵۰٪ هزینه تولید نرم افزار را به خود اختصاص خواهد داد [۱]. به همین دلیل تولیدکنندگان نرم افزار و پژوهشگران در زمینه فرآیندهای تولید نرم افزار، به دنبال آرایه راه کارهایی به منظور خودکار سازی آزمون نرم افزار می باشند. با وجود تحقیقات فراوانی که در این زمینه انجام شده، تاکنون هیچ روشی که بتواند تمام مراحل آزمون نرم افزارهای شی گرا و یا ساخت یافته را به صورت خودکار انجام دهد ارائه نشده است [۲]. علاوه بر نرم افزارهای تحقیقاتی، تعدادی نرم افزار تجاری نیز به این منظور به وجود آمده اند که موفق ترین آنها تنها به انجام آزمون کلاس در سطح کد می پردازند.

آنچه در این مقاله به آن پرداخته شده، آرایه روشی در زمینه آزمون خودکار نرم افزارهای شی گرا می باشد. این روش قادر به آزمون نرم افزار در سطوح کلاس، خوشه و سیستم بوده و برای دریافت ضوابط نرم افزار از زبان مدل سازی UML به همراه زبان قید شی^۱ استفاده می کند. آزمون سطح الگوریتمی (به عبارت دیگر آزمون واحد یا آزمون سطح متد^۲) در این مقاله مد نظر نمی باشد. روش آرایه شده، بر پایه موردهای کاربری^۳ و ضوابط UML بوده در نتیجه به کاربری های کاربران (یا سایر سیستمها) از نرم افزار اهمیت می دهد. برای هر مورد کاربری نرم افزار یک دیاگرام ترتیب^۴ در مرحله طراحی و برای هر متد مشارکت کننده در دیاگرام ترتیب نیز یک دیاگرام فعالیت^۵ دریافت می شود. یادآوری می گردد در دیاگرام فعالیت، قیود و شرطهای مورد نظر توسط زبان قید شی (OCL) مشخص شده اند. با دریافت این دیاگرامها و دیاگرام کلاس مربوط به هر مورد کاربری با استفاده از روش های آرایه شده در این مقاله، از تلفیق دیاگرام های ترتیب و فعالیت، خط مسیر اجرای سناریوی مورد نظر بر اساس متدهای همکاری کننده به دست می آید. (لازم به یادآوری است که هر مورد کاربری از چند سناریوی کاربری تشکیل شده که هر کدام دارای دیاگرام ترتیب تشریح کننده آن می باشد). پس از آن با استفاده از شرطهای موجود در خط مسیر اجرای هر سناریوی کاربری، دستگاه معادلاتی به دست می آید که با حل هر دستگاه می توان داده های آزمون مربوط به خط مسیر اجرای سناریوی کاربری را تولید کرد. به منظور واری سحت عملکرد نرم افزار در حین اجرای داده های آزمون در سوره برنامه نیز خطوطی به منظور ثبت وضعیت پیش شرطها، وضعیت پس شرطها، وضعیت نامتغیرها^۶ و روند اجرای متدها گنجانده می شود تا وضعیت های اشاره شده در فایل های کنترلی ذخیره شوند (این پیش شرطها، پس شرطها و نامتغیرها توسط طراح نرم افزار تهیه شده و به صورت ضوابط OCL می باشند)، این سوره برنامه

میلادی توسط Doong و همکارش [۴] ارایه شد. در این روش که برای آزمون سطح کلاس به کار می‌رود، هر مورد آزمون شامل یک جفت ترتیب از پیغام‌ها با پرچسب‌هایی می‌باشد که مشخص می‌کنند آیا ترتیب‌ها باید اشیای کلاس تحت آزمون را در حالات یکسان قرار دهند. آزمون به این صورت انجام می‌شود که به ترتیب پیغام‌ها به اشیا کلاس تحت آزمون فرستاده می‌شود و سپس مقدار بازگشتی از اشیا با مقداری که از طرف کاربر اعلام شده است (اراکل)، مقایسه می‌شود. در این روش با استفاده از ضوابط جبری^{۱۳} کلاس مورد آزمون و تعدادی قوانین بازنویسی عبارات^{۱۴} می‌توان به صورت خودکار داده آزمون را تولید کرد. همچنین به منظور خودکارسازی اجرای آزمون نیز ابزاری معرفی شده است که با تولید داده آزمون آن را بر روی کلاس مورد آزمون اجرا می‌کند.

متدولوژی TACCLE برای آزمون سطح کلاس و سطح خوشه نرم‌افزار توسط Chen و همکارانش [۳] ارایه شده است. این متدولوژی از ایده‌های [۵و۴] برای آزمون استفاده می‌کند. در [۵] یک متدولوژی به منظور تلفیق روش‌های جعبه سیاه و جعبه سفید برای آزمون نرم‌افزارهای شی‌گرا معرفی شده است. در این متدولوژی از روش جعبه سیاه برای انتخاب مورد‌های آزمون و از روش جعبه سفید به منظور تشخیص معادل بودن دو شی که از اجرای یک مورد آزمون حاصل شده‌اند، به کار می‌رود. در TACCLE از روش ضوابط جبری برای بیان کلاس‌ها و از Contract [۶] برای بیان تبادل پیغام در بین خوشه‌ها استفاده می‌شود. ضابطه جبری یک کلاس دارای دو قسمت تعریف نحوی و توصیف معنایی می‌باشد. تعریف نحوی، فهرست عملیات را بیان می‌کند و توصیف معنایی با استفاده از گزاره‌های تساوی^{۱۵}، خصوصیات رفتاری عملیات را بیان می‌کنند.

Chen و همکارانش [۷] با گسترش ایده [۵] که روشی برای آزمون سطح کلاس نرم‌افزارهای شی‌گرا بر پایه ضوابط جبری می‌باشد، به ارایه ابزاری برای خودکارسازی آزمون سطح کلاس نرم‌افزارهای شی‌گرا پرداخته‌اند. در این ابزار از روش ارایه شده در [۵] برای تولید داده آزمون استفاده می‌شود با این تفاوت که با ارایه راه‌کارهایی از کمبودهای موجود در روش ارایه شده، کاسته شده و همچنین دامنه انتخاب مورد‌های آزمون با حفظ پوشش مورد نظر، کاهش داده شده است. این منظور با استفاده از یک گراف به نام گراف اعضا داده مرتبط^{۱۶} (DRG) به دست می‌آید. این گراف تجریدی از پیاده‌سازی ضوابط مورد نظر است. Murray و همکارانش [۸] نیز ابزاری برای آزمون سطح کلاس بر اساس ضوابط برنامه ارایه کرده‌اند.

Kim و همکارانش [۹] از دیگرام‌های حالت UML برای آزمون سطح کلاس استفاده کرده‌اند. مجموعه‌ای از محدوده‌های پوشش براساس کنترل‌ها و گردش داده موجود در این نمودارها تعیین شده‌اند. در ابتدا دیگرام‌های حالت به ماشین‌های حالت متناهی توسعه یافته^{۱۷} (EFSM) تبدیل شده و سپس با تبدیل آن‌ها به گراف جریان از روش‌های جریان داده استفاده می‌شود.

Offutt و همکارانش ابتدا از ضوابط UML [۱۰] و سپس از دیگرام‌های همکاری UML [۱۱] برای آزمون سطح سیستم نرم‌افزارهای شی‌گرا استفاده کرده‌اند. روش ارایه شده تنها در پوشش تولید آزمون‌های ترتیب کامل رویدادها که نیازمند دانش مهندس آزمون می‌باشد، موفق عمل نمی‌کند. در این روش، زوج‌های تعریف-استفاده داده‌ها، تمام زوج‌های تعریف-استفاده داده‌های سراسری^{۱۸}، ایجاد-استفاده اشیا^{۱۹}، استفاده-نابودی اشیا^{۲۰} و ایجاد-نابودی اشیا^{۲۱} بررسی می‌شوند. در تحقیقی که توسط Offutt و همکاران [۱۲] ارایه شده است، ساختار کلی روش‌های برپایه حالت مورد بررسی قرار گرفته است. بر این اساس چند معیار کلی برای آزمون نرم‌افزارهای برپایه حالت ارایه شده است.

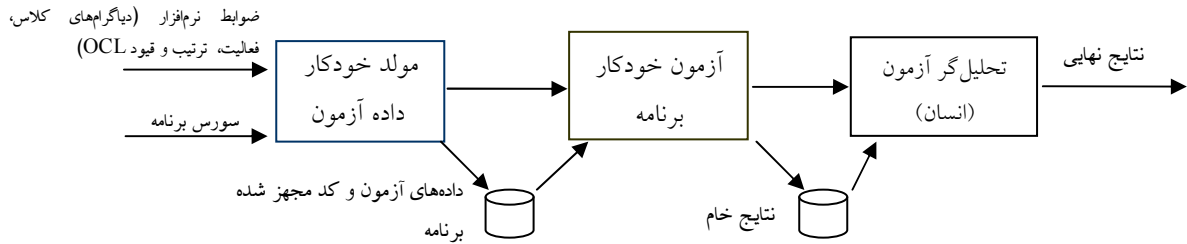
Frakin و همکارش [۱۳] ابزاری را به منظور تولید داده آزمون برای نرم‌افزارهای شی‌گرا در سطح الگوریتمی برپایه Java با استفاده از دیگرام‌های ترتیب UML معرفی کرده‌اند. در این روش از مقادیر ورودی در دیگرام‌های ترتیب برای تولید داده آزمون استفاده می‌شود. برای آزمون برنامه‌های Java، Marinov و همکارش چارچوبی ارایه کرده‌اند [۱۴] و Boyapati و همکارانش نیز ابزاری به نام Korat [۱۵] [۱۵] برای آزمون برنامه‌های Java ارایه کرده‌اند.

در [۱۶] روی طریقه آزمون خودکار نرم‌افزار از روی ضوابط جبری تحقیقی صورت گرفته است. در این مقاله بر روی ویژگی‌های ADT^{۲۲} برای آزمون نرم‌افزار تکیه شده‌است. در روش ارایه شده با استفاده از قوانین بازنویسی، ضوابط نرم‌افزار اجرا شده و به عنوان اراکل با نتایج اجرای کد مقایسه می‌گردند.

Chan و همکارانش [۱۷] یک مرور اجمالی از روش‌های آزمون یکپارچگی برای برنامه‌های شی‌گرا ارایه کرده‌اند. ما نیز [۱۸] روشی برای تولید داده آزمون با استفاده از دیگرام‌های UML و ضوابط OCL ارایه کرده‌ایم که در معماری پیشنهادی (در این مقاله) برای خودکارسازی فرآیند آزمون از نتایج [۱۸] به عنوان بخشی از این معماری استفاده می‌کنیم.

۳- سیستم آزمون خودکار

ساختار سیستم آزمون پیشنهادی در شکل شماره ۱ ارایه شده که به طور کلی از سه جزء مولد خودکار داده آزمون، آزمون خودکار برنامه و تحلیل‌گر آزمون تشکیل شده است. مولد خودکار داده آزمون، با دریافت ورودی‌های مورد نیاز که همان‌گونه که در شکل شماره ۱ نشان داده شده عبارتند از ضوابط نرم‌افزار و سورس برنامه، داده آزمون مورد نیاز را تولید می‌کند. پس از تولید داده آزمون، جزء آزمون خودکار برنامه، داده‌های تولید شده را بر روی کد برنامه، اجرا کرده و نتایج را در تعدادی فایل کنترلی ذخیره می‌کند. در نهایت جزء تحلیل‌گر آزمون با استفاده از نتایج به دست آمده در جزء قبل به تحلیل آزمون انجام شده می‌پردازد.



شکل (۱): ساختار سیستم آزمون نرم افزارهای شی گرا (پیشنهادی)

۳-۱-۱- مولد درخت فراخوانی متد

درخت فراخوانی متد، درختی است که گره‌های آن فراخوانی متد و یال‌های آن شرط فراخوانی متد می‌باشند. نمونه‌ای از درخت فراخوانی متد در شکل شماره ۳ نمایش داده شده است. با استفاده از این درخت می‌توان خط اجرای سناریو را ترسیم کرده و شرط‌های لازم برای اجرای سناریو را به دست آورد. این درخت فراخوانی برای اساس سناریوی کاربری ارایه شده در نرم‌افزار نمونه (پیوست شماره یک) تولید شده است. دیاگرام مورد کاربری در شکل شماره ۱-۱ و دیاگرام همکاری سناریوی مورد نظر در شکل شماره ۱-۳ در پیوست شماره یک نمایش داده شده‌اند.

به منظور ایجاد درخت فراخوانی متد، می‌بایست مسیر اجرایی سناریوی مورد کاربری به دست آید. روش به دست آوردن مسیر، استفاده از دیاگرام ترتیب می‌باشد که در آن سیر فراخوانی متدها بیان شده است. پس از تعیین متدهای همکاری کننده و تشخیص ورودی‌ها و خروجی‌های آن‌ها از روی دیاگرام کلاس، دیاگرام فعالیت مربوط به هر متد به منظور تعیین مسیرهای مختلف سناریوی شرح داده شده به همراه قیود اجرای آن و به دست آوردن درخت فراخوانی متد مورد استفاده قرار می‌گیرد. الگوریتم تولید درخت فراخوانی متد به منظور تولید این درخت ابتدا مسیرهای موجود در دیاگرام فعالیت را تعیین می‌کند. در هر متد مسیری که مربوط به سناریوی مورد نظر می‌باشد جدا می‌شود. این مسیر به ترتیب فراخوانی متدها که در دیاگرام ترتیب مشخص شده، پیمایش می‌شود و با رسیدن به هر نماد در دیاگرام فعالیت اگر نماد آغاز، پایان، حالت و یا فعالیت باشد، چهار گره برای متدهای موجود در رخدادهای آن (entry, do, exit, event) ایجاد می‌کند. گره‌های entry, do, exit به ترتیب به هم متصل می‌شود. از هر کدام از گره‌ها با شرط موجود در on event یک یال به گره مربوط به on event وصل می‌شود. اگر نماد انتقال باشد یک گره برای event آن ایجاد شده و از گره قبلی یک یال با شرط موجود در gurad_condition به آن متصل می‌شود. این عمل تا پایان مسیر ادامه می‌یابد. به این صورت از مسیر، یک درخت فراخوانی متد به دست می‌آید [۱۸].

طراحی تفصیلی اجزای ساختار روش پیشنهادی در ادامه شرح داده می‌شود. لازم به ذکر است که تمامی دیاگرام‌های دریافتی در این روش، دیاگرام‌های مرحله طراحی نرم افزار می‌باشند.

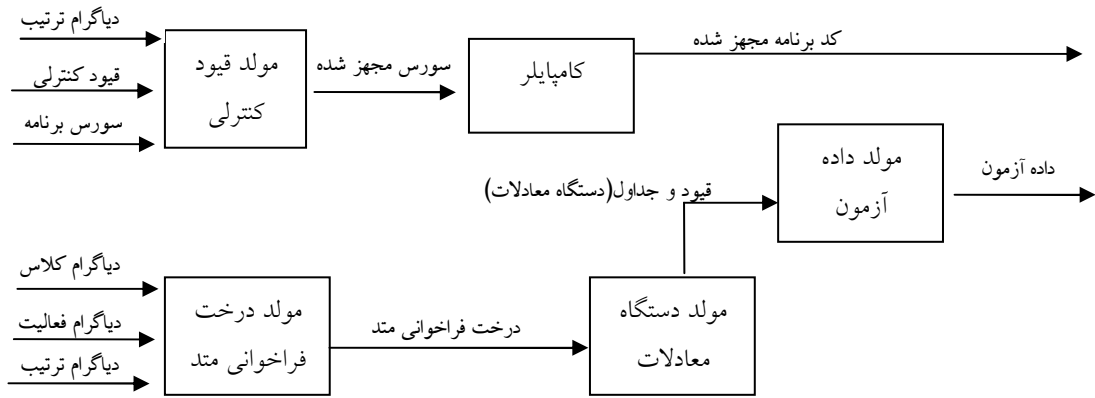
۳-۱-۲- مولد خودکار داده آزمون

جزء مولد خودکار داده آزمون خود جزء مهم و پیچیده‌ای است که از زیر بخش‌های مولد قیود کنترلی، مولد درخت فراخوانی متد، مولد دستگاه معادلات و مولد داده آزمون تشکیل شده است. زیربخش‌های فوق‌الذکر به همراه ارتباط آن‌ها در شکل شماره ۲ ارایه شده است. زیربخش مولد قیود کنترلی با دریافت دیاگرام ترتیب، قیود کنترلی (OCL) و سورس برنامه، تعدادی خط برنامه به منظور ثبت مقادیر قیود کنترلی و روند اجرای متدها در حین اجرای برنامه، به سورس برنامه اضافه می‌کند که این عمل مجهزسازی برنامه نامیده می‌شود. خروجی این زیربخش، قیود دریافت شده و سورس مجهز شده برنامه می‌باشد. این سورس به کامپایلر داده شده و خروجی آن (کد مجهز شده برنامه) به عنوان ورودی به جزء آزمون خودکار برنامه داده می‌شود.

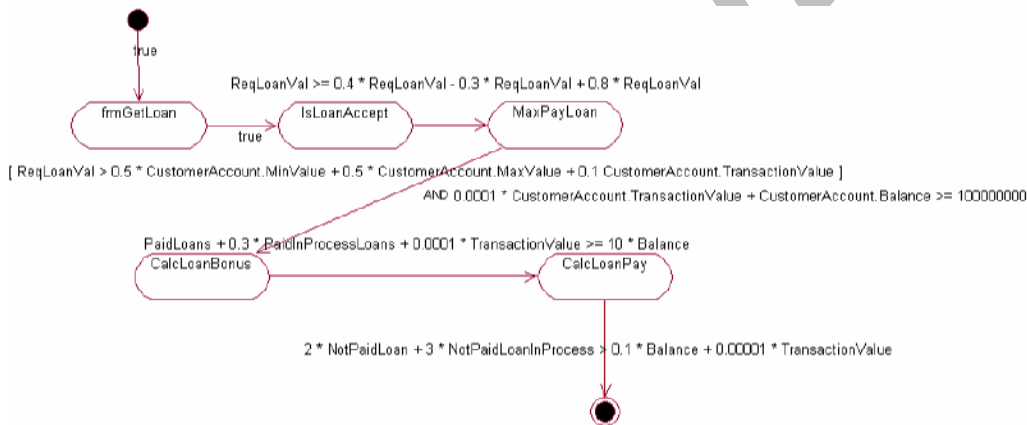
زیربخش مولد درخت فراخوانی متد با دریافت دیاگرام ترتیب مربوط به سناریوی کاربری مورد نظر، دیاگرام کلاس و دیاگرام‌های فعالیت مربوط به متدهای همکاری کننده بر اساس الگوریتم‌های ویژه‌ای [۱۸] درخت فراخوانی متد را تولید می‌کند.

زیربخش مولد دستگاه معادلات با دریافت درخت فراخوانی متد، دستگاه معادلات برای اجرای این سناریو را تولید می‌کند. این دستگاه معادلات شرط‌هایی می‌باشند که ارضا شدن آن‌ها باعث اجرای مسیر مورد نظر در درخت فراخوانی متد و یا به عبارتی اجرای سناریوی کاربری مورد نظر می‌شود.

دستگاه معادلات همراه با قیود کنترلی به زیربخش مولد داده آزمون داده شده و از روی آن‌ها داده آزمون برای سناریوی کاربری مورد نظر تولید می‌شود. این داده‌ها همراه با کد برنامه مجهز شده به جزء آزمون خودکار برنامه داده می‌شود.



شکل (۲): زیربخش ها و ارتباطات آنها در جزء مولد خودکار داده آزمون



شکل (۳): نمونه ای از درخت فراخوانی متد در نرم افزار نمونه (پیوست شماره یک)

۳-۱-۲- مولد دستگاه معادلات

متدهای همکاری کننده در آن مشخص شده است، پارامترهای ورودی متدهای همکاری کننده بر اساس ورودی‌های متد آغازکننده سناریو و همچنین ویژگی‌های کلاس‌ها بازنویسی می‌شوند. به این ترتیب دستگاه معادلات نهایی بر اساس ورودی‌های متد آغازکننده و ویژگی‌های کلاس‌های همکاری کننده به دست می‌آید و برای حل و تولید داده آزمون به زیربخش مولد داده آزمون فرستاده می‌شود. نمونه‌ای از دستگاه معادلات برای سناریوی کاربری انتخابی در مثال در شکل شماره ۴ ارائه شده است.

پس از تولید درخت فراخوانی متد، برای تولید دستگاه معادلات کافی است که مجموعه شرط‌هایی را که در خط اجرای سناریو در درخت متد وجود دارد را در قالب یک دستگاه معادلات قرار دهیم. با حل این دستگاه می‌توان مقدار ورودی‌های مورد نیاز برای پیمایش مسیر مورد نظر و در نتیجه داده آزمون را به دست آورد. به منظور تشکیل این دستگاه ابتدا شرط‌های موجود بر روی یال‌هایی که در درخت فراخوانی متد قرار دارند، انتخاب می‌شوند. این شرط‌ها دارای پارامترهایی از نوع ورودی‌ها و خروجی‌های متدهای همکاری کننده و همچنین ویژگی‌های کلاس‌های مربوط می‌باشند. لازم به یادآوری است که داده آزمون متشکل از مقادیر ورودی متد آغاز کننده سناریو و ویژگی‌های کلاس‌های همکاری کننده می‌باشد. به همین دلیل با استفاده از دیاگرام فعالیت متد آغاز کننده که نحوه فراخوانی

$ReqLoanVal \geq 0.4 * ReqLoanVal - 0.3 * ReqLoanVal + 0.8 * ReqLoanVal$,
 $ReqLoanVal > 0.5 * CustomerAccount.MinValue + 0.5 * CustomerAccount.MaxValue + 0.0001 * CustomerAccount.TransactionValue$,
 $CustomerAccount.PaidLoans + 0.3 * CustomerAccount.PaidInProcessLoans + 0.0001 * CustomerAccount.TransactionValue \geq 10 * CustomerAccount.Balance$,
 $2 * CustomerAccount.NotPaidLoans + 3 * CustomerAccount.NotPaidInProcessLoans > 0.0001 * 0.00001 * CustomerAccount.TransactionValue + 0.1 * CustomerAccount.Balance$

شکل (۴): نمونه‌ای از دستگاه معادلات مربوط به مسیر انتخابی در شکل شماره ۳

۳-۱-۳- مولد قیود کنترلی

در بخش ۳-۲-۱ ملاحظه خواهید کرد که روش پیشنهادی از پیش‌شرطها و پس‌شرطهای متدهای همکاری کننده و نامتغیرها به عنوان اراکل استفاده می‌کند. به این منظور باید بتوان مقادیر آن‌ها را در حین اجرا ثبت نمود تا درستی اجرای برنامه از روی آن واریسی شود. مولد قیود کنترلی ابتدا، دیاگرام ترتیب و سوریس برنامه مورد نظر را دریافت می‌کند. از روی دیاگرام ترتیب، ترتیب اجرای متدها و اشیای همکاری کننده و کلاس آن‌ها به دست می‌آید. سپس آن دسته از قیود کنترلی (نامتغیر^{۲۴}) مربوط به هر کلاسی که شی‌ای از آن در دیاگرام ترتیب مرحله طراحی سناریوی کاربری مورد نظر استفاده شده است، دریافت می‌شود. این قیود را طراح برنامه به صورت دستی و به صورت ضوابط OCL در جزء مولد خودکار داده آزمون وارد می‌کند. نامتغیرها شرطهایی هستند که در هر حالت در کلاس و یا اشیای آن برقرار می‌باشند. آن‌گاه ابتدا به کلیه متدهای موجود در برنامه خطوطی اضافه می‌شود که ورود به متد و اجرای آن‌ها در یک فایل کنترلی ثبت می‌کند. سپس خطوطی به ابتدا و انتهای متد اضافه می‌شوند که درست بودن پیش‌شرط^{۲۵} را در هنگام ورود به متد و پس‌شرط^{۲۶} را در هنگام خروج از متد در یک فایل کنترلی ثبت می‌کنند. مقادیر پیش‌شرطها و پس‌شرطها از ضوابط نرم‌افزار به دست می‌آید. (این شرطها از نوع خطی می‌باشند) پس از آن به منظور بررسی درستی قیود موجود بر روی کلاس (نامتغیر) در ابتدای متد و بعد از خطوط مربوط به ثبت اجرای متد و پیش‌شرط، خطوطی اضافه می‌شود که درستی قیود موجود بر روی کلاسی که متد در آن قرار دارد و همچنین، کلاس‌هایی که اشیای آن‌ها به عنوان ورودی به متد داده می‌شوند، را در یک فایل کنترلی دیگر ثبت می‌کند. سپس این خطوط (همان سوریس برنامه به همراه کدهای اضافه شده، به نام سوریس برنامه مجهز شده) به کامپایلر داده شده، کد برنامه به دست آمده به عنوان ورودی به جزء آزمون خودکار برنامه داده می‌شود. از طرف دیگر قیود کنترلی مربوط به کلاس‌ها به عنوان شرط به زیربخش مولد داده آزمون منتقل می‌شود تا در تولید داده استفاده شود.

۳-۱-۴- مولد داده آزمون

به منظور تولید داده آزمون می‌بایست معادلات به دست آمده از بخش ۳-۱-۲ را حل نمود. پیش از حل این دستگاه معادلات باید این نکته را مد نظر داشت که پس از ساده‌سازی دستگاه (حذف معادلات یکسان)

به منظور حل آسان‌تر آن، نامعادلات را با اضافه کردن یک مجهول به معادله تبدیل می‌کنیم. برای این کار ابتدا تمام مجهول‌ها و ثابتهای موجود در معادله را به یک طرف برده، تمام نامعادلات را به صورت $ax + by + \dots \geq 0$ یا $ax + by + \dots > 0$ تبدیل می‌کنیم آن‌گاه متغیر c را به صورت زیر به آن اضافه می‌کنیم.

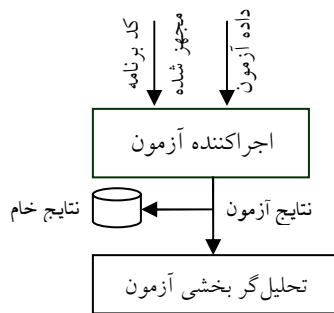
الف: اگر نامعادله به صورت $ax + by + \dots \geq 0$ باشد آن را به صورت $ax + by + \dots + c = 0$ و $c \geq 0$ تبدیل می‌کنیم.

ب: اگر نامعادله به صورت $ax + by + \dots > 0$ باشد آن را به صورت $ax + by + \dots + c = 0$ و $c > 0$ تبدیل می‌کنیم.

با مجهول اضافه شده به مانند یک ثابت در هنگام حل معادلات رفتار می‌کنیم. با استفاده از این روش دستگاه نامعادلات به دستگاه معادلات تبدیل می‌شود.

پس از تبدیل دستگاه نامعادلات به دستگاه معادلات برای حل ساده دستگاه ابتدا تعداد معادلات و تعداد مجهولات را با هم برابر می‌کنیم. برای انجام این کار، ابتدا پارامترهای آزاد (پارامترهایی که ورودی تابع آغازکننده مورد کاربری نیستند) را به سمت راست معادلات می‌بریم، اگر بازم تعداد مجهول‌های موجود در سمت چپ از تعداد معادلات بیشتر بود با انتخاب مجهول‌های دارای تعداد تکرار بیشتر در معادلات و سمت راست بردن آن‌ها، تعداد مجهول‌های موجود در سمت چپ را با تعداد معادلات برابر می‌کنیم. در این حالت با حل نمادی دستگاه معادلات و با استفاده از روش کرامر و یا روش حذفی می‌توان به الگوی آزمون دست یافت. اگر دستگاه معادلات ناسازگار باشد دارای جواب نمی‌باشد. در غیر این صورت دستگاه دارای یک یا چند جواب است. در حالت خاص اگر دستگاه همگن و سازگار باشد (صفر بودن ثابت‌ها) و اگر تعداد مجهولها از معادلات بیشتر باشد دستگاه بی‌نهایت جواب دارد و در صورت مساوی و یا کمتر بودن تعداد مجهول‌ها از معادلات دستگاه یک یا چند جواب دارد [۱۹].

سمت راست معادله را به صورت یک تابع در نظر گرفته و با استفاده از دترمینان معکوس و ضرب آن در این تابع، جواب معادله برای تمامی متغیرها به دست می‌آید. در صورت صفر بودن دترمینان از روش حذفی استفاده می‌کنیم. جواب به دست آمده الگوی آزمون برای متغیرها می‌باشد. با توجه به پارامترهای C_i اضافه شده برای هر متغیر یک دسته الگو پیدا می‌شود که با مقداردهی C_i می‌توان الگوها را به دست آورد.



شکل (۷): زیربخش‌ها و ارتباطات آنها در جزء آزمون خودکار برنامه

۳-۲-۱- اجرا کننده آزمون

زیربخش اجراکننده آزمون وظیفه دریافت کد برنامه مجهز شده و اجرای آن بر اساس داده آزمون را برعهده دارد. برای انجام این کار کد برنامه باید دارای دو پیش شرط زیر باشد:

الف- کلیه قطعاتی^{۲۷} که کلاس‌های آغاز کننده سناریو در آنها قرار دارند دارای واسط برنامه‌نویسی باشند.

ب- متد آغازکننده سناریو در دیاگرام ترتیب مشخص شده باشد. این مساله توسط ایجاد یک عامل با نام Tester که این متد را فراخوانی می‌کند، صورت می‌گیرد.

بر این اساس پس از مشخص کردن سناریوی مورد آزمون، دیاگرام ترتیب متناظر آن سناریو از درون ضوابط UML خوانده شده، پس از تعیین متد آغازین، آن متد با داده‌های مورد نظر اجرا می‌شود. بدیهی است که سایر متدهای روی مسیر نیز اجرا می‌شوند. در حین اجرای آزمون، نتایج حاصل توسط خطوطی که در برنامه تعبیه شده در فایل‌های کنترلی نوشته می‌شود تا در مرحله بعدی (زیربخش تحلیل‌گر بخشی آزمون) با بررسی این فایل‌ها روند فراخوانی متدها و نقض نامتغیرها را بررسی کرده، نتایج آزمون را اعلام کند.

یکی از مهمترین وظایف این بخش تولید اشیایی می‌باشد که به عنوان ورودی به متد آغازکننده داده می‌شوند. همان‌گونه که در بخش قبل اشاره شد، داده‌های آزمون برای ویژگی‌های اشیاء تولید می‌شود و این بخش می‌بایست بر این اساس یک شی از کلاس ورودی با مقدار ویژگی‌های تولید شده ایجاد کند. به ویژگی‌هایی که مقدار آنها در هنگام تولید داده آزمون، مشخص نشده یک مقدار تصادفی داده می‌شود.

۳-۲-۲- تحلیل‌گر بخشی آزمون

در روش پیشنهادی به روش مرسوم از اراکل برای بررسی درستی اجرای برنامه با داده آزمون استفاده نمی‌شود. بلکه اراکل مورد نظر در

با استفاده از الگوی تولید داده آزمون می‌توان تعداد بیشماری داده آزمون تولید کرد. کارشناس آزمون تعداد مورد نیاز داده آزمون را برای آزمون سناریوی مورد نظر توسط جزء آزمون خودکار برنامه را وارد کرده، سپس جزء آزمون خودکار برنامه به تعداد مورد نظر با روش فوق‌الذکر به تولید داده آزمون می‌پردازد.

الگوی تولید داده آزمون و یک نمونه از داده آزمون تولید شده برای نرم‌افزار نمونه (پیوست الف) متناظر با معادله شکل شماره ۴ در شکل‌های شماره ۵ و ۶ ارایه شده است.

$$0.1 * ReqLoanVal \geq 0$$

$$ReqLoanVal > 0.5 * CustomerAccount.MinValue + 0.5 * CustomerAccount.MaxValue + 0.1 * CustomerAccount.TransactionValue$$

$$0.0001 * CustomerAccount.TransactionValue + CustomerAccount.Balance \geq 100000000$$

$$CustomerAccount.PaidLoans + 0.3 * CustomerAccount.PaidInProgressLoans + 0.0001 * CustomerAccount.TransactionValue \geq 10 * CustomerAccount.Balance$$

$$2 * CustomerAccount.NotPaidLoans + 3 * CustomerAccount.NotPaidInProgressLoans > 0.1 * CustomerAccount.Balance + 0.00001 * CustomerAccount.TransactionValue$$

شکل (۵): الگوی تولید داده آزمون برای مسیر انتخابی در شکل

شماره ۳

$$ReqLoanVal = 24398578485.54$$

$$CustomerAccount.MinValue = 346283.74$$

$$CustomerAccount.MaxValue = 7883034.64$$

$$CustomerAccount.TransactionValue = 3453037345.59$$

$$CustomerAccount.Balance = 4313464.95$$

$$CustomerAccount.PaidLoans = 45314847.34$$

$$CustomerAccount.PaidInProgressLoans = 3493.12$$

$$CustomerAccount.NotPaidLoans = 457304.39$$

$$CustomerAccount.NotPaidInProgressLoans = 998694031.23$$

شکل (۶): داده آزمون تولیدی برای مسیر انتخابی در شکل

شماره ۳

۳-۲-۳- آزمون خودکار برنامه

همان‌گونه که در شکل شماره ۷ نشان داده شده است، جزء آزمون خودکار برنامه شامل زیربخش اجرا کننده آزمون و زیربخش تحلیل‌گر بخشی آزمون است. زیربخش اجراکننده آزمون، کد برنامه مجهز شده را با داده‌های آزمون اجرا می‌کند. نتایج این عمل در فایل‌های کنترلی ذخیره می‌شود. سپس تحلیل‌گر بخشی آزمون، نتایج آزمون را از این فایل‌ها دریافت کرده و آن‌ها را تفسیر می‌کند. نتایج این تفسیر در قالب فایل نتایج ذخیره می‌شوند.

این روش، مجموعه پیش‌شرطها و پس‌شرطهای متدها و نامتغیرهای نرم‌افزار برقرار باشند. پس می‌توان این مجموعه (پیش‌شرطها و پس‌شرطهای متدهای همکاری کننده و نامتغیرهای کلاس‌های همکاری کننده و همچنین روند اجرای متدها در سناریوی همکاری) را به عنوان اراکل آزمون در نظر گرفت. در نتیجه پس از اجرای آزمون، تحلیل‌گر بخشی آزمون به بررسی درستی روند اجرای سناریو و روش ما یک فرد متخصص است، وظیفه تحلیل نهایی نتایج آزمون را برعهده دارد. نتایج بیشتر آزمون در [۲۰] ارایه شده است که در این جا به جهت کمبود جا به آن نمی پردازیم.

۴- پیاده سازی و ارزیابی روش پیشنهادی

در این بخش ابتدا به بررسی نحوه پیاده سازی روش پیشنهادی و سپس بررسی تطبیقی روش پیشنهادی با دیگر روش‌های موجود می‌پردازیم. برپایه مطالب عنوان شده در بخش‌های پیشین، می‌توان پارامترهایی را به منظور بررسی و مقایسه روش‌ها تعریف کرد. این پارامترها عبارتند از نحوه تعیین سناریوی آزمون، ساختار مورد استفاده برای مدل سازی نرم‌افزار، مولد تولید داده آزمون، میزان خودکارسازی آزمون.

۴-۱- پیاده سازی روش پیشنهادی

روش ارایه شده در این مقاله با استفاده از نرم افزار VisualStudio.net در چارچوب .net و زبان C# پیاده سازی شده است. ضوابط نرم افزار ورودی برپایه UML مانند پیوست شماره ۱ می باشند.

کلیه ضوابط UML مورد نیاز از طریق فایل مدل Rational Rose دریافت شده و اطلاعات مربوط به ضوابط OCL نیز به نرم‌افزار آزمون خودکار وارد می‌شود. نتایج حاصل از اجرای نرم افزار آزمون خودکار برای سیستم مدیریت ریسک در بخش ۳ ارایه شد.

نرم افزار آزمون پیاده سازی شده به منظور آزمون سه سیستم: مدیریت ریسک وام، حسابداری انبار و دفترداری و همچنین بسته نرم افزاری گردش کار به کار گرفته شده است. نمونه ساده شده سیستم مدیریت ریسک که نرم افزار در آزمون آن به کار گرفته شده در پیوست شماره ۱ ارایه شده است.

۴-۲- ارزیابی روش پیشنهادی

منظور از سناریوی آزمون، تعیین مفهومی است که بر اساس آن بتوان موردی آزمون را به دست آورد. هر سناریوی آزمون دارای یک دسته مورد آزمون می‌باشد که معمولا می‌توان آن‌ها را در قالب یک الگوی آزمون ارایه کرد.

موجود در کلاس‌های سناریوی کاربری می‌باشند که به منظور تضمین درستی اجرای برنامه باید در طول اجرای برنامه درستی آن‌ها برقرار باشد. به طور کلی برای هر متد چنانچه پیش‌شرط داشته باشیم دارای یک محافظ^{۲۸} هستیم که در صورت برقرار نبودن وارد متد نشویم. از طرف دیگر با وجود پس‌شرط می‌توان ناهنجاری در اجرای متد را بررسی کرد و در صورت بروز ناهنجاری تصمیم مناسب گرفت. نامتغیرها هم با توجه به ماهیت کلاس‌ها می‌بایست در طول اجرای همچنین بررسی درستی پیش‌شرطها و پس‌شرطهای متدها و نامتغیرهای موجود بر روی کلاس‌ها می‌پردازد. این نتایج در شکل‌های شماره ۸ و ۹ ارایه شده‌اند. پیش‌شرطها و پس‌شرطهای متدها و نامتغیرهای موجود بر روی کلاس‌های نرم‌افزار نمونه در شکل شماره ۹-۱ در پیوست شماره یک نمایش داده شده است.

```
InquiryLoan : frmGetLoan.ShowLoanReq
LoanManager.IsLoanAccept
LoanControler.MaxPayLoan
Customer.CalcLoanBonus
Customer.CalcLoanPay
```

شکل (۸): مسیر اجرایی متدها در نمونه شکل شماره ۳

```
collaborationTrace : true
CustomerAccount, Invariant: Inv1, true
CustomerAccount, Invariant: Inv2, true
CustomerAccount, Invariant: Inv3, true
CustomerAccount, Invariant: Inv4, true
CustomerAccount, Invariant: Inv5, true
CustomerAccount, Invariant: Inv6, true
CustomerAccount, Invariant: Inv7, true
CustomerAccount, Invariant: Inv8, true
```

شکل (۹): تحلیل اجرای آزمون برای نمونه شکل شماره ۳

۴-۳- تحلیل‌گر آزمون

همان‌گونه که در بخش ۳-۲-۲ اشاره شد، در این مقاله اراکل را به صورت مقادیر روند فراخوانی متدهای همکاری کننده، پیش‌شرط و پس‌شرط موجود در متدها و همچنین مقادیر نامتغیرها جهت درستی مسیر اجرای آزمون در نظر گرفتیم. با توجه به این که ممکن است طراح تمام پیش‌شرطها، پس‌شرطها و نامتغیرها را به صورت کامل وارد نکرده باشد و از طرفی تعداد مورد نیاز داده آزمون توسط یک عامل انسانی مشخص می‌شود، پس از اجرای آزمون به یک عامل انسانی به منظور تصمیم‌گیری در مورد درستی نتایج حاصل از آزمون و کامل بودن آزمون با توجه به تعداد داده‌های آزمون، نیاز است. تحلیل‌گر انسانی علاوه بر کنترل درستی نتایج حاصل در هنگام اجرای آزمون برای تمام سناریوهای کاربری می‌تواند سناریوهایی را که هیچ‌گاه اجرا نمی‌شوند را نیز شناسایی کند. این مسیرها می‌توانند به دو دلیل ایجاد شده باشند که عبارتند از: (۱) مورد کاربری که از قلم افتاده و در نتیجه پیاده‌سازی نشده است (۲) مسیر غیرقابل گذر (منطقی و یا غیر منطقی) که این مورد معمولا ناشی از اشتباه طراح می‌باشد. پس از تولید نتایج آزمون و تحلیل بخشی آن جزء تحلیل‌گر آزمون که در

جدول (۱): مقایسه روشهای آزمون نرم افزارهای شی گرا با روش پیشنهادی

چگونگی تعیین سناریوی آزمون	ساختار یکپارچه مدل سازی	استفاده از روشهای متداول	مولد تولید داده آزمون	میزان خودکارسازی آزمون
[۳] دید طراحی	×	×	نحوه تبادل پیغام ها	متوسط
[۵] دید طراحی		×	نحوه تبادل پیغام ها	کم
[۷] دید طراحی	✓	×	نحوه تبادل پیغام ها	کم
[۸] دید طراحی	×	×	نحوه تبادل پیغام ها	
[۹] دید طراحی	✓	✓	شرطهای دیاگرام همکاری	کم
[۱۰] دید طراحی	✓	✓	×	کم
[۱۱] دید طراحی	✓	✓	×	کم
روش پیشنهادی دید کاربر نهایی	✓	✓	الگوی تولید داده	متوسط

جدول ۱ اشاره شده، تنها روش پیشنهادی است که از این رویکرد استفاده می کند.

از دید معیار میزان خودکارسازی آزمون، روش پیشنهادی دارای خودکارسازی با درجه متوسط می باشد که درجه بالایی از خودکارسازی است. در بین روش های موجود تنها روش [۳] دارای این درجه از خودکار سازی است. یادآوری می گردد، با ارایه کامل پیش شرطها و پس شرطهای متدها و نامتغیرهای موجود بر روی کلاسها به جزء آزمون خودکار برنامه می توان تا حد بسیار زیادی نیمه خودکار روش پیشنهادی را کاهش داده، روند تحلیل را تقریباً خودکار نمود.

از دید معیار ساختار یکپارچه مدل سازی، در روش پیشنهادی از زبان مدل سازی UML که استاندارد غیررسمی مدل سازی نرم افزارهای شی گرا است به منظور مدل سازی نرم افزار استفاده شده که به همین دلیل یکپارچگی در مدل سازی نیز وجود دارد. OCL نیز زبان ایجاد قید بر روی اشیا است که به همراه زبان مدل سازی UML و به منظور ارایه امکاناتی برای توصیف بهتر مدل های شی گرا، ارایه شده است. با توجه به گسترش استفاده از UML، پژوهشها بر روی روش های آزمون با استفاده از آن در حال گسترش است. روش های [۹]، [۱۰] و [۱۱] نیز به منظور مدلسازی از UML و دیگر روشها از مدل های ضوابط جبری استفاده کرده اند. روش [۷] با وجود استفاده از ضوابط جبری دارای ساختار یکپارچه می باشد

تقریباً تمامی روشهایی که نیازمند به دست آوری مقادیر آزمون در حین اجرا و یا واریسی حین اجرای نرم افزار می باشند از روش مجهزسازی کد استفاده می کنند. روش پیشنهادی نیز، به منظور استخراج مقادیر مورد نیاز (در زمان اجرای نرم افزار) از مجهزسازی کد برنامه استفاده کرده است. در صورتی که کد برنامه موجود نباشد با استفاده از این روش تنها می توان به تولید داده آزمون پرداخت و بر روی دیاگرامهای فعالیت محل اضافه کردن کدهای ثبت کننده مقادیر پیش شرطها، پس شرطها و نامتغیرها را مشخص کرد.

ساختار مورد استفاده در مدل سازی برنامه که در روش های مبتنی بر ضوابط بررسی می شود خود می تواند به دو بخش: (۱) یکپارچگی در مدل سازی و (۲) میزان استفاده از روش های معمول مهندسی نرم افزار در مدل سازی (مانند زبان UML) تقسیم شود.

میزان خودکارسازی را می توان در سه دسته: کم، متوسط و کامل در نظر گرفت. روشها با خودکارسازی کم، آنهایی هستند که تنها مرحله تولید داده آزمون در آنها خودکار شده است. روشها با خودکارسازی متوسط، علاوه بر تولید داده آزمون، بخشی از تحلیل و روشها با خودکارسازی کامل، تمام فرآیند آزمون را خودکارسازی کرده اند.

مقایسه روش پیشنهادی با روش های مطرح شده در بخش ۲ مقاله در جدول ۱ ارایه شده است. در این جدول سطرها نمایانگر روشها و ستونها عوامل مقایسه می باشند.

همانگونه که در جدول ۱ مشاهده می شود، از دید معیار چگونگی تعیین سناریوی آزمون، روش پیشنهادی بر خلاف روشهای دیگر بر دید کاربر از سیستم نهایی متکی است و نه کد و یا طراحی سیستم. در روش پیشنهادی از سناریوهای مورد کاربری (به عبارت دیگر نیازها در سطح کاربران) که روش استفاده کاربر نهایی از سیستم می باشد به عنوان سناریوی آزمون استفاده شده است.

از دید معیار مولد تولید داده آزمون، الگوی تولید داده آزمون از دستاوردها و قابلیت های مهم روش پیشنهادی می باشد که با پالایش شرط اجرای مسیر به دست می آید. الگوی تولید داده آزمون را هم می توان به منظور تولید خودکار داده آزمون مورد استفاده قرار داد و هم این الگو می تواند به عنوان یک مرجع مقایسه ای برای درستی طراحی توسط طراح مورد استفاده قرار گیرد. تنها روش پیشنهادی در بین روش های موجود است که به استخراج الگوی آزمون می پردازد. با توجه به این که الگوی تولید داده آزمون در روش پیشنهادی از تلفیق مدل های ایستا و پویای نرم افزار به دست می آید می توان به برتری آن در برابر روشهایی که از تبادل پیغام و یا دیاگرام همکاری (که تنها مدل های پویا می باشند) بهره می برند، اشاره کرد. همانگونه که در

۵- نتیجه گیری

تولید داده آزمون و آزمون خودکار نرم افزار به منظور تعیین درستی نرم افزار از گام های مهم در آزمون نرم افزار می باشد. در صورتی که گام تولید داده آزمون، داده های مورد نیاز را به صورت صحیح و به تعداد مناسب (با توجه به نامحدود بودن داده آزمون) تولید کند، دیگر مراحل آزمون نرم افزار با سرعت و دقت بیشتری انجام خواهد شد. به همین منظور تحقیقاتی در زمینه خودکارسازی این روند صورت گرفته است. تحقیقات انجام شده در این زمینه عمدتاً در زمینه تولید داده آزمون به منظور ایجاد پوشش در یک زمینه خاص مانند پوشش شاخه می باشد. در زمینه شی گزایی نیز تحقیقات عمدتاً بر روی تولید خودکار داده آزمون برای کلاس ها تکیه دارند. در صورتیکه با تکیه بر آزمون نرم افزار بر اساس سناریوهای کاربری می توان بسیاری از خطاهای سطح بالای نرم افزار را شناسایی کرد که این مساله منجر به شناسایی خطاهای سطح پایین تر خواهد شد.

در این مقاله روشی به منظور تولید خودکار داده آزمون و آزمون خودکار، برای نرم افزارهای شی گرا ارائه شده است. این روش بر اساس تولید داده برای سناریوهای کاربری طراحی شده است. در روش پیشنهادی، سناریوی کاربری در مرحله طراحی بر اساس دیگرام های ترتیب مرحله طراحی دریافت شده و ضوابط هر متد همکاری کننده در سناریو از طریق OCL و دیگرام فعالیت مشخص می گردد. بر اساس دیگرام فعالیت و مسیری که برای اجرای سناریو مورد نظر، نرم افزار از آن می گذرد، قیود کنترلی برای ارضای سناریو مشخص می شود. از حل این قیود می توان مقادیر داده مورد نظر برای آزمون را به دست آورد.

مزیت این روش بر روش های دیگر: (۱) در تکیه آن بر سناریوهای کاربری سیستم می باشد در حالی که روش های دیگر بر سناریوهای موجود در کد برنامه یا طراحی سیستم تکیه کرده اند، (۲) از دید معیار مولد تولید داده آزمون، الگوی تولید داده آزمون از دستاوردها و قابلیت های مهم روش پیشنهادی می باشد، (۳) از دید معیار میزان خودکارسازی آزمون، روش پیشنهادی دارای خودکارسازی با درجه متوسط می باشد که درجه بالایی از خودکارسازی است و در بین روش های موجود تنها روش [۳] دارای این درجه خودکارسازی است، و در نهایت (۴) استفاده از قیود کنترلی و ضوابط OCL و تلفیق دیگرام های ترتیب و فعالیت به منظور دستیابی به مسیر دقیق اجرای نرم افزار بدون استفاده از کد برنامه از نقاط متمایز کننده روش پیشنهادی از روش های موجود است.

در زمینه واریاسی طراحی با توجه به این نکته که امکان تولید شرط های مورد نیاز برای اجرای یک مسیر در روش ذکر شده وجود دارد، می توان با توسعه روش پیشنهادی، مدلی را برای تشخیص ناسازگاری در طراحی بر اساس شرط های مسیر و نامتغیرهای کلاس به دست آورد.

به منظور توسعه روش پیشنهادی برای تشخیص کارکردهای کاشته شده در نرم افزار (به عبارت دیگر وظیفه مندیهایی خواسته نشده از

نرم افزار) با توجه به این که هر مسیر در نرم افزار به یک دستگاه معادلات نگاشت می شود، می توان با تحلیل نرم افزار و درآوردن مستندات طراحی مورد نیاز از طریق پیمایش کد به بررسی کارکردهای موجود در نرم افزار و کارکردهای مورد نیاز در طراحی پرداخت. به عنوان مثال با بالا بردن سطح انتزاع کد می توان طراحی را از روی آن حدس زده و با طراحی موجود مقایسه کرد. همچنین می توان با نگاشت بین تحلیل و طراحی علاوه بر بررسی درستی طراحی به درستی نگاشت بین تحلیل و طراحی نیز پرداخت.

در روش پیشنهادی امکان انجام آزمون رگرسیون وجود ندارد که می تواند به عنوان تحقیق در آینده مورد بررسی قرار گیرد. به این منظور باید بتوان بین نسخه های مختلف نرم افزار، نگاشتی برای تولید و یا انتخاب داده های تولید شده قبلی ایجاد کرد.

برای خودکارسازی کامل تحلیل نرم افزار در این زمینه می بایست بررسی گسترده ای در مورد تعداد مقدار موارد آزمون برای پوشش آزمون مورد نظر در نرم افزارهای گوناگون انجام شود.

در حال حاضر در روش پیشنهادی، امکان آزمون فرآیندهای موازی وجود ندارد که می توان این ویژگی را با اعمال تغییراتی بر روی الگوریتم تولید درخت فراخوانی متد و الگوریتم تولید داده و اجرای آزمون آن را پیاده سازی کرد.

در روش پیشنهادی خطوطی به سورت نرم افزار به منظور ثبت مقادیر پیش شرط ها، پس شرط ها و نامتغیرها اضافه می شود. این خطوط به صورت غیر فعال عمل می کنند به این معنا که در روند اجرای برنامه دخالتی نکرده و فقط مقادیر مورد نیاز را در فایل های کنترلی ثبت می کنند. با فعال کردن این خطوط می توان به واریاسی حین اجرای برنامه پرداخت. برای این منظور به هر کدام از شرط ها و متغیرها یک درجه اهمیت نسبت داده می شود که این درجه اهمیت به عنوان مثال می تواند متوسط، مهم و بسیار مهم باشد. با توجه به این درجه اهمیت در حین اجرای برنامه واکنش لازم صورت می گیرد. به عنوان مثال در صورت نقض شرط های بسیار مهم علاوه بر ثبت آن در فایل کنترلی می توان از کاربر درخواست کرد که آیا مایل به ادامه اجرای برنامه می باشد یا خیر.

رویکرد جنبه گرا [۲۱]، رویکردی نوین در تولید نرم افزار می باشد که با توسعه رویکرد شی گرا سعی در بهبود روش های تولید نرم افزار دارد. یکی از جنبه های مهم این رویکرد جداسازی قطعات بر اساس کارکرد و سپردن هر کارکرد به یک قطعه خاص می باشد. با استفاده از این رویکرد در سیستم پیشنهادی می توان بدون اضافه کردن کدهای مورد نیاز به برنامه به صورت فیزیکی، با تهیه قطعه ای جدا و با فراخوانی متدها به صورت خودکار عمل ثبت مقادیر پیش شرط ها، پس شرط ها و نامتغیرها را انجام داد.

مراجع

[۱] Edvardsson J. and M. Kamkar, "Analysis of the Constraint Solver in UNA Based Test Data Generation", 9th ACM

Software Testing and Analysis (ISSTA), Rome, Italy, 2002.

- [۱۶] Antoy S. and D. Hamlet, "Automatically Checking an Implementation against Its Formal Specification", IEEE Transactions on Software Engineering, 26(1), pp.55-69, 2000.
- [۱۷] Chan W.K., T.Y.Chen and T.H.Tse, "An overview of integration testing techniques for object-oriented programs", 2nd ACIS Annual International Conference on Computer and Information Science(ICIS 2002), pp.696-701, 2002.

[۱۸] سعید جلیلی و آرش کربلایی، تولید داده آزمون با استفاده از UML و ضوابط OCL، دهمین کنفرانس انجمن کامپیوتر ایران، مرکز تحقیقات مخابرات ایران، اسفند ۱۳۸۳.

[۱۹] Richard A. Silverman., Modern Calculus And Analytic Geometry, pp.608-615, 1969.

[۲۰] آرش کربلایی، آزمون خودکار نرم افزارهای شی گرا با استفاده از ضوابط UML، پایان نامه کارشناسی ارشد، دانشگاه تربیت مدرس، بهمن ۱۳۸۳.

[۲۱] Elrad. T., Filman R.E. and A. Bader, Aspect-oriented programming :Introduction, Communication of the ACM, 44(10), pp.24-32, 2001.

پیوست الف

الف - ۱ - مقدمه

نرم‌افزاری بر اساس روش پیشنهادی شرح داده شده در مقاله تهیه شده است. در این بخش یک مثال را به منظور بررسی نتایج حاصل از آزمون نرم‌افزار با استفاده از نرم‌افزار تولید شده، ارائه می‌کنیم. کلیه ضوابط UML مورد نیاز از طریق فایل مدل Rational Rose دریافت شده و اطلاعات مربوط به ضوابط OCL نیز به صورت دستی در نرم‌افزار نمونه آزمون وارد می‌شود. نتایج حاصل از اجرا در متن مقاله ارائه شده‌اند. در این قسمت تنها به شرح مثال و ارائه ورودی‌های سیستم می‌پردازیم.

الف - ۲ - مدیریت ریسک وام

نرم‌افزار مورد نظر براساس نحوه اخذ وام از یک بانک و بررسی مقدار حداکثر دریافتی وام بنا شده است. دیاگرام مورد کاربری در شکل شماره ۱-۱ مشخص شده است. نمودار کلاس این مورد کاربری در شکل شماره ۱-۲ ارائه شده است. در این دیاگرام، کلاس‌های Account، Customer، LoanManager، frmGetLoan و LoanControler وجود دارند که در زیر به شرح هر کدام از متدها و ویژگی‌های مهم مربوط به هر کلاس می‌پردازیم. این متدها در فرآیند تعیین حداکثر وام قابل دریافت با یکدیگر همکاری می‌کنند. در کلاس frmGetLoan (فرم درخواست) متد ShowLoanReq درخواست وام را دریافت کرده و پس از محاسبه مقدار حداکثر وام قابل دریافت نتیجه درخواست را نمایش می‌دهد در کلاس LoanManager (کلاس کنترلی وام) متد IsLoanAccept با نوع خروجی Boolean نمایانگر قبولی و یا عدم قبولی در خواست وام می‌باشد. در کلاس Customer

SIGSOFT International Symposium on the Foundations of Software Engineering (FSE-9), pp.237 - 245, 1998.

- [۲] Michael C. C. and McGraw G., "Automated software test data generation for complex programs", Proceedings of Automated Software Engineering, pp.136-146, 1998.
- [۳] Chen H.Y., T.Y.Chen and T.H.Tse, TACCLE: "a methodology for object-oriented software testing at the class and cluster levels. ACM Transactions on Software Engineering and Methodology", 10(1):pp.56-109, 2001.
- [۴] Doong R.K. and P.G. Frankl, "The ASTOOT approach to testing object oriented programs, ACM Transactions on Software Engineering and Methodology", 3(2):pp.101-130, 1994.
- [۵] Chen H.Y., T.Y.Chen, T.H.Tse and F.T. Chan, In black and white: "an integrated approach to class-level testing of object-oriented programs, ACM Transactions on Software Engineering and Methodology", 7(3):pp.250- 295, 1998.
- [۶] Helm, R., I. M. Holland, and D. Gangopadhyay, "Contract: specifying behavioral compositions in object-oriented systems. In Proceedings of the 5th Annual Conference on Object-Oriented programming Systems, Languages and Applications OOPSALA '90", ACM SIGPLAN Notices 25(10): pp.169-180, 1990.

[۷] Chen H. Y. and T. H. Tse and Y. T. Deng, ROCS: "an object-oriented class-level testing system based on the relevant observable contexts technique, Information and Software Technology", 42(10):PP.677-686, 2000.

[۸] Murray L., J. McDonald and P. Strooper, "Specification-based Class Testing with ClassBench", proceedings of Asia-Pacific Software Engineering Conference, pp.164-173, 1998.

[۹] Kim Y.G., H.S. Hong, S.M. Cho, D.H. Bae, and S.D. Cha, "Test Case Generation from UML State Diagrams". IEE Proceedings: Software, 146(4) pp.187-192, 1999.

[۱۰] Offutt J. and A. Abdurazik, Generating Test from UML Specification. Proceedings of UML'99 -The Unified Modeling Language. Beyond the Standard. Second International Conference, Springer 1723:pp.416-429, 1999.

[۱۱] Offutt J. and A. Abdurazik, "Using UML Collaboration Diagrams for Static Checking and Test Generation". UML 2000 - The Unified Modeling Language, Advancing the Standard, Third International Conference, Springer vol. 1939, pp.385-395, 2000.

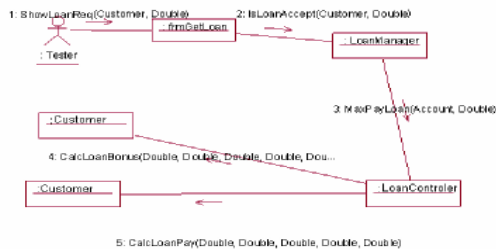
[۱۲] Offutt J., S. Liu, A. Abdurazik and P. Ammann, "Generating test data from state-based specifications, Software Testing Verification and Reliability", 13:pp.23-53, 2003.

[۱۳] Fraikin F. and T. Leonhardt, "SeDiTeC-testing based on sequence diagrams", International Conference on Automated Software Engineering, pp.261-266, 2002.

[۱۴] Marinov D. and S. Khurshid, TestEra: "A Novel Framework for Testing Java Programs", 16th IEEE Conference on Automated Software Engineering, pp. 22-31, 2001.

[۱۵] Boyapati C., S. Khurshid and D. Marinov, Korat: "Automated testing based on Java predicates", In Proceedings of the 2002 International Symposium on

دیگرام همکاری مربوط به سناریوی شکل شماره ۱-۱ در شکل شماره ۳-۱ نمایش داده شده است. همان گونه که در این شکل مشخص است، ابتدا کاربر درخواست وام می‌کند و مقدار وام درخواستی با هویت مشتری به متد کنترل کننده فرستاده می‌شود. این متد با ارسال پیغامی محاسبه بیشترین مقدار وام قابل دریافت توسط شخص مورد نظر را درخواست می‌کند. سپس با مقایسه این مقدار با مقدار وام درخواستی، امکان دریافت وام مورد نظر توسط این کاربر مشخص شده، نتیجه به متد آغاز کننده فرستاده می‌شود. این متد بر اساس نتیجه، پیغام مناسب را به کاربر نمایش می‌دهد. دیگرام‌های فعالیت مربوط به متدهای همکاری کننده در شکل‌های شماره ۱-۴ تا ۱-۸ نمایش داده شده‌اند. شکل شماره ۱-۹ هم بیانگر قیود کنترلی ورودی می‌باشد.

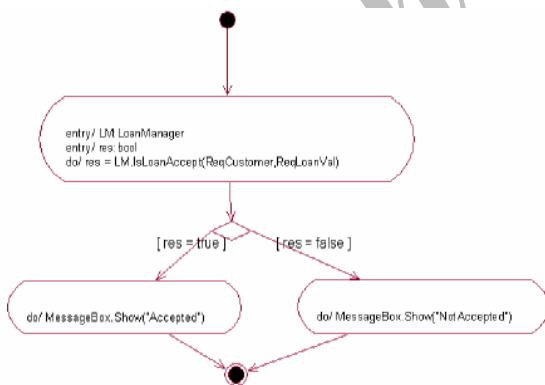


شکل (الف-۳): دیگرام همکاری سناریوی مورد آزمون

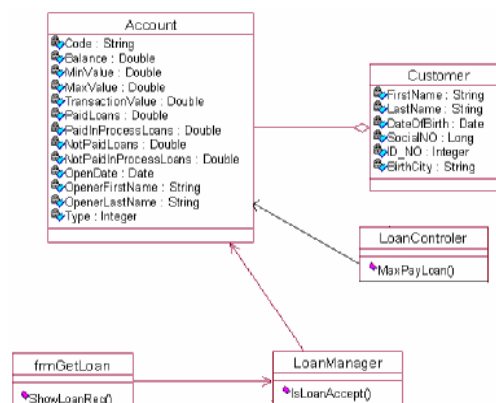
(مشتری) ویژگی‌های **FirstName**، **LastName** و **SocialNo** به ترتیب نمایانگر نام، نام خانوادگی و شماره ملی مشتری و متدهای **CalcLoanPay** و **CalcLoanBonus** با نوع برگشتی **double** به ترتیب محاسبه جوایز بر اساس سوابق و محاسبه جریمه بر اساس دیرکردها را برعهده دارند. در کلاس **LoanControler** متد **MaxPayLoan** با نوع بازگشتی وظیفه محاسبه حداکثر وام دریافتی را برعهده دارد. در کلاس **Account** (حساب بانکی) نیز **Code**، **Balance**، **OpenDate**، **MaxValue**، **MinValue**، **TransactionValue**، **PaidLoans**، **NotPaidInProcessLoans** و **PaidInProcessLoans** به ترتیب نمایانگر کد حساب، موجودی، حداقل میزان موجودی کلی، حداکثر میزان موجودی کلی، تاریخ افتتاح حساب، مقدار کلی تراکنش‌های مالی، مبلغ وام‌های درحال پرداخت که با مشکل مواجه شده‌اند، مبلغ وام‌های پرداختی، مبلغ وام‌های در حال پرداخت و نوع حساب می‌باشد.



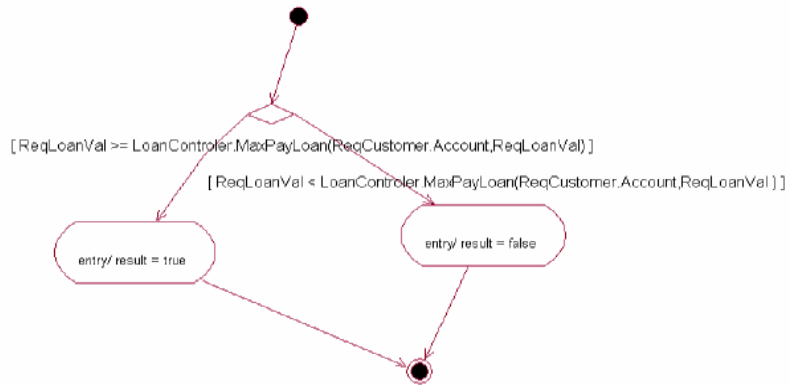
شکل (الف-۱): دیگرام مورد کاربری



شکل (الف-۴): دیگرام فعالیت متد **ShowLoanReq**



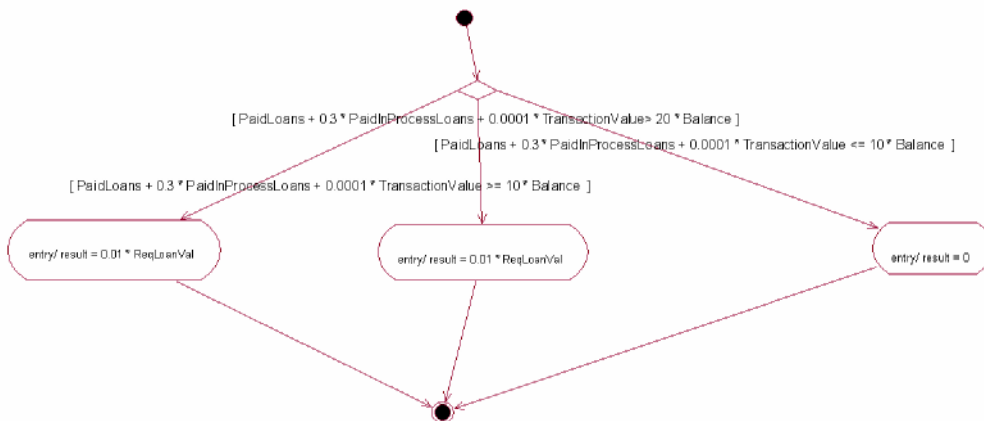
شکل (الف-۲): نمودار کلاس مورد کاربری



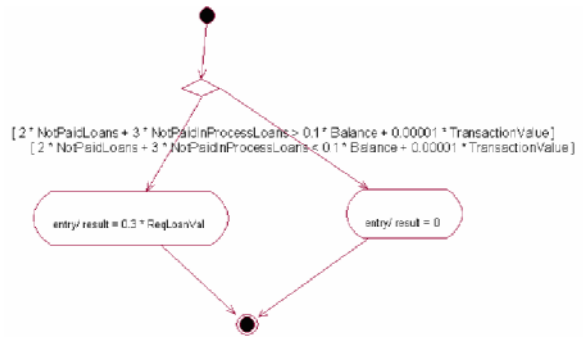
شکل (الف - ۵): دیاگرام فعالیت متد IsLoanAccept



شکل (الف - ۶): دیاگرام فعالیت متد MaxLoanPay



شکل (الف - ۷): دیاگرام فعالیت متد Calc Loan Bonus



شکل (الف-۸): دیاگرام فعالیت متد Calc LoanPay

Customer.Account.Balance >=0
 Customer.Account.MinValue >=0
 Customer.Account.MaxValue >=0
 Customer.Account.NotPaidLoans >=0
 Customer.Account.PaidLoans >=0
 Customer.Account.MaxValue >= Customer.Account.MinValue
 Customer.Account.MaxValue >= Customer.Account.Balance
 Customer.Account.Balance >= Customer.Account.MinValue

شکل (الف-۹): قیود موجود بروی کلاس ها

زیر نویسها

- 1 Object Constraint Language(OCL)
- 2 Unit Test
- 3 Use Cases
- 4 Sequence Diagram
- 5 Activity Diagram
- 6 Invariant
- 7 Instrumented Code
- 8 Algorithmic Level
- 9 Method Level
- 10 Class Level
- 11 Cluster Level
- 12 System Level
- 13 Algebraic Specification
- 14 Term Rewriting Rules
- 15 Equational axioms
- 16 Data Member Relevance Graph
- 17 Extended Finite State Machine
- 18 Global variable definition-use
- 19 Object creation-use
- 20 Object use-destruction
- 21 Object construction-destruction
- 22 Abstract Data Type
- 23 attribute
- 24 Invariant
- 25 Precondition
- 26 PostCondition
- 27 Component
- 28 Guard