آزمون خودکار نرم‌افزارهای شی‌گرا
با استفاده از موردهای کاربری و ضوابط OCL

سعید جلیلی[۱] آرش کربلایی[۲]

۱- استادیارگروه کامپیوتر، دانشکده فنی و مهندسی– دانشگاه تربیت‌مدرس – تهران– ایران
SJalili@modares.ac.ir

۲- دانش‌آموخته کارشناسی ارشد گروه کامپیوتر، دانشکده فنی و مهندسی– دانشگاه تربیت‌مدرس- تهران- ایران
Arash.Karbalaei@gmail.com

چکیده : آزمون خودکار با استفاده از ضوابط نرم‌افزار از مهمترین زمینه‌های تحقیقاتی آزمون نرم‌افزار می‌باشد. اهمیت آزمون خودکار در کاهش هزینه‌های تولید نرم‌افزار و بالا بردن ضریب اطمینان به نرم‌افزار است. روش‌های موجود برای آزمون خودکار با تعریف معیارهای مورد نیاز که وابسته به کد و یا ضوابط نرم‌افزار می‌باشد، 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 نرم‌افزارهای شی‌گرا برپایه مورد کاربری نرم‌افزار ارایه شده است. معماری این روش از سه بخش تولید خودکار داده آزمون، آزمون خودکار برنامه و تحلیل نتایج تشکیل می‌شود. با تعریف سناریوی کاربری مورد نظر در قالب دیاگرام ترتیب مرحله طراحی و دریافت دیاگرام‌های کلاس، و فعالیت مورد نیاز برنامه در مرحله طراحی و کد برنامه، سیستم به تولید داده آزمون، اجرای آن و تحلیل نتایج آزمون می‌پردازد. از نتایج مهم این روش امکان تولید الگوی آزمون، تولید داده آزمون بر اساس سناریوهای کاربری، امکان انجام آزمون‌های مختلف و استفاده از ساختار یکپارچه برای مدل‌سازی می‌باشد.

کلمات کلیدی: آزمون خودکار نرم‌افزار، تولیـد داده آزمون، آزمون نرم‌افزار شی‌گرا، مورد کاربری، ضوابط OCL

تاریخ دریافت مقاله : ۱۳۸۴/۱۲/ ۲۹
تاریخ پذیرش مقاله : ۱۳۸۷/۶/۳۰
نام نویسنده‌ی مسئول : سعید جلیلی
نشانی نویسنده‌ی مسئول : تهران- تقاطع بزرگراه چمران و جلال آل‌احمد- دانشگاه تربیت‌مدرس- دانشکده‌ی فنی و مهندسی- گروه کامپیوتر

۱- مقدمه

بطور کلی آزمون نرم‌افزار به دلیل صرف وقت زیاد و هزینه مالی بالایی که به روند تولید نرم‌افزار تحمیل می‌کند، معمولا در فرآیند تولید نرم‌افزار مورد توجه کافی قرار نمی‌گیرد و همین امر از کیفیت نرم‌افزارهای تولیدی تا حد زیادی می‌کاهد. بر اساس تحقیقات آماری انجام شده، اگر آزمون نرم‌افزار به صورت کامل انجام شود در حدود ۵۰٪ هزینه تولید نرم‌افزار را به خود اختصاص خواهد داد[۱]. به همین دلیل تولیدکنندگان نرم‌افزار و پژوهشگران در زمینه فرآیندهای تولید نرم‌افزار، به دنبال ارایه راه‌کارهایی به منظور خودکارسازی آزمون نرم‌افزار می‌باشند. با وجود تحقیقات فراوانی که در این زمینه انجام شده، تاکنون هیچ روشی که بتواند تمام مراحل آزمون نرم‌افزارهای شی‌گرا و یا ساختیافته را به صورت خودکار انجام دهد ارایه نشده است[۲]. علاوه بر نرم‌افزارهای تحقیقاتی، تعدادی نرم‌افزار تجاری نیز به این منظور به وجود آمده‌اند که موفق‌ترین آن‌ها تنها به انجام آزمون کلاس در سطح کد می‌پردازند.

آن‌چه در این مقاله به آن پرداخته شده، ارایه روشی در زمینه آزمون خودکار نرم‌افزارهای شی‌گرا می‌باشد. این روش قادر به آزمون نرم‌افزار در سطوح کلاس، خوشه وسیستم بوده و برای دریافت ضوابط نرم‌افزار از زبان مدل‌سازی UML به همراه زبان قید شی[۱] استفاده می‌کند. آزمون سطح الگوریتمی(به عبارت دیگر آزمون واحد یا آزمون سطح متد[۲]) در این مقاله مد نظر نمی‌باشد. روش ارایه شده ، برپایه موردهای کاربری[۳] و ضوابط UML بوده در نتیجه به کاربری‌های کاربران (یا سایر سیستم‌ها) از نرم‌افزار اهمیت می‌دهد. برای هر مورد کاربری نرم‌افزار یک دیاگرام ترتیب[۴] در مرحله طراحی و برای هر متد مشارکت کننده در دیاگرام ترتیب نیز یک دیاگرام فعالیت[۵] دریافت می‌شود. یادآوری می‌گردد در دیاگرام فعالیت، قیود و شرط‌های مورد نظر توسط زبان قید شی(OCL) مشخص شده‌اند. با دریافت این دیاگرام‌ها و دیاگرام کلاس مربوط به هر مورد کاربری با استفاده از روش‌های ارایه شده در این مقاله، از تلفیق دیاگرام‌های ترتیب و فعالیت، خط مسیر اجرای سناریوی مورد نظر بر اساس متدهای همکاری کننده به دست می‌آید. (لازم به یادآوری است که هر مورد کاربری از چند سناریوی کاربری تشکیل شده که هر کدام دارای دیاگرام ترتیب تشریح کننده آن می‌باشد). پس از آن با استفاده از شرط‌های موجود در خط مسیر اجرای هر سناریوی کاربری، دستگاه معادلاتی به دست می‌آید که با حل هر دستگاه می‌توان داده‌های آزمون مربوط به خط مسیر اجرای سناریوی کاربری را تولید کرد. به منظور وارسی صحت عملکرد نرم‌افزار در حین اجرای داده‌های آزمون در سورس برنامه نیز خطوطی به منظور ثبت وضعیت پیش‌شرط‌ها، وضعیت پس‌شرط‌ها، وضعیت نامتغیرها[۶] و روند اجرای متدها گنجانده می‌شود تا وضعیت‌های اشاره شده در فایل‌های کنترلی ذخیره شوند(این پیش‌شرط‌ها، پس‌شرط‌ها و نامتغیرها توسط طراح نرم‌افزار تهیه شده و به صورت ضوابط OCL می‌باشند)، این سورس برنامه دوباره کامپایل شده و کد مجهزشده‌ی[۷] برنامه را ایجاد می‌کند. سپس داده‌های آزمون تولید شده از روی دستگاه معادلات بر روی این کد برنامه اجرا می‌شود و در نتیجه اطلاعات مربوط به وضعیت‌های فوق‌الذکر به تفکیک هر سناریوی کاربری در یک فایل کنترلی ذخیره می‌شوند. در نهایت بخش تحلیل آزمون با بررسی اطلاعات ذخیره شده، به تحلیل درستی اجرای آزمون می‌پردازد. در رویکرد پیشنهادی اراکل آزمون به عبارت دیگرهمان تلفیق پیش‌شرط‌ها و پس‌شرط‌های متدهای همکاری کننده در سناریوی کاربری و نامتغیرهای کلاس‌های همکاری کننده در سناریو و روند فراخوانی متدها در سناریوی کاربری است.

در ادامه در بخش دوم، سابقه تحقیقات انجام شده در آزمون خودکار نرم‌افزار را مورد بررسی قرار می‌دهیم. در بخش سوم، روش پیشنهادی معرفی شده و اجزای آن به تفصیل تشریح می‌شوند. در بخش چهارم، روش پیشنهادی مورد ارزیابی قرار می‌گیرد. در بخش پنجم، تحقیقات آینده در این زمینه و همچنین نقاط قابل توسعه روش پیشنهادی معرفی می‌گردد. در بخش آخر نیز نتیجه‌گیری ارایه خواهد شد. پیوست شماره یک حاوی شرح یک نرم‌افزار نمونه می‌باشد که به منظور تشریح نحوه عمل روش خودکار آزمون، ارایه شده است. شرح کلی این نرم‌افزار نمونه به همراه دیاگرام‌های ورودی آن در این پیوست قرار دارد و نتایج اجرای روش بر روی نرم‌افزار نمونه در بخش‌های مختلف مقاله ارایه شده است. در این نرم‌افزار نمونه به دلیل کاهش صفحات مقاله از دیاگرام همکاری به جای دیاگرام ترتیب(که معادل یکدیگر هستند) استفاده شده است.

۲- روش‌های آزمون خودکار نرم‌افزارهای شی‌گرا

آزمون نرم افزارهای شی گرا دارای چهار سطح می باشد که این سطوح از پایین‌ترین واحد شی‌گرایی آغاز شده و به بالاترین واحد که سیستم می‌باشد ختم می‌شوند. 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 می‌گیرند. آزمون در سطح کلاس به کنترل درستی کوچکترین واحد شی‌گرایی یعنی کلاسها می‌پردازد. در این سطح کلیه کلاسها از لحاظ درستی کارکرد و برآوردن نیازهایی که طبق آن‌ها تعریف شده‌اند بررسی می‌شوند. در سطح خوشه کلاسهای همکار در قالب مفهومی به نام خوشه گروه‌بندی شده، و روابط بین اشیای موجود در خوشه بررسی می‌شود. در سطح سیستم کل نرم‌افزار از لحاظ کارکرد مورد آزمون قرار می‌گیرد. برای این منظور روابط بین زیرسیستم‌ها (خوشه‌ها) در جهت انجام کارآیی مورد نیاز بررسی می‌گردد. آزمون سطح سیستم با ورودیها و خروجیهای قابل درک برای کاربران نهایی سیستم انجام می‌شود[۳].

از قدیمی‌ترین تحقیقات انجام شده در خودکارسازی آزمون نرم‌افزارهای شی‌گرا روش ASTOOT می‌باشد که در سال ۱۹۹۴

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 برچسب‌هایی 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 شده‌است (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 [۴و۵] 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 برپایه 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(DRG[۱۶]) 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 دیاگرام‌های 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 دیاگرام‌های 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 دیاگرام‌های 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 بر پایه حالت ارایه شده است.

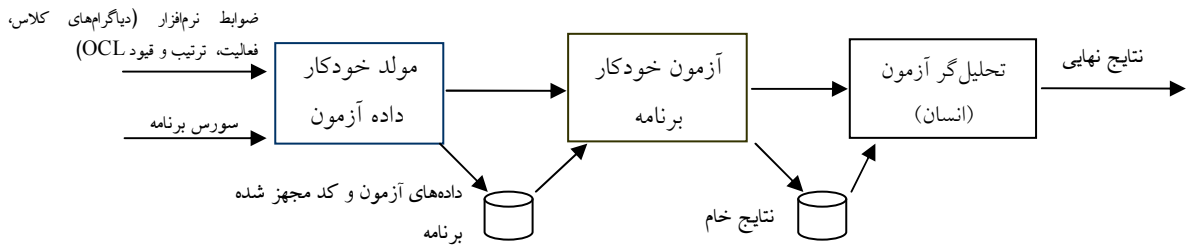
Fraikin و همکارش [۱۳] ابزاری را به منظور تولید داده آزمون برای نرم‌افزارهای شی‌گرا در سطح الگوریتمی برپایه Java با استفاده از دیاگرام‌های ترتیب UML معرفی کرده‌اند. در این روش از مقادیر ورودی در دیاگرام‌های ترتیب برای تولید داده آزمون استفاده می‌شود. برای آزمون برنامه‌های Java، Marinov و همکارش چارچوبی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

در [۱۶] روی طریقه آزمون خودکار نرم‌افزار از روی ضوابط جبری تحقیقی صورت گرفته است. در این مقاله برروی 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 دیاگرام‌های 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)

طراحی تفضیلی 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- 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 مجهزشده 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.

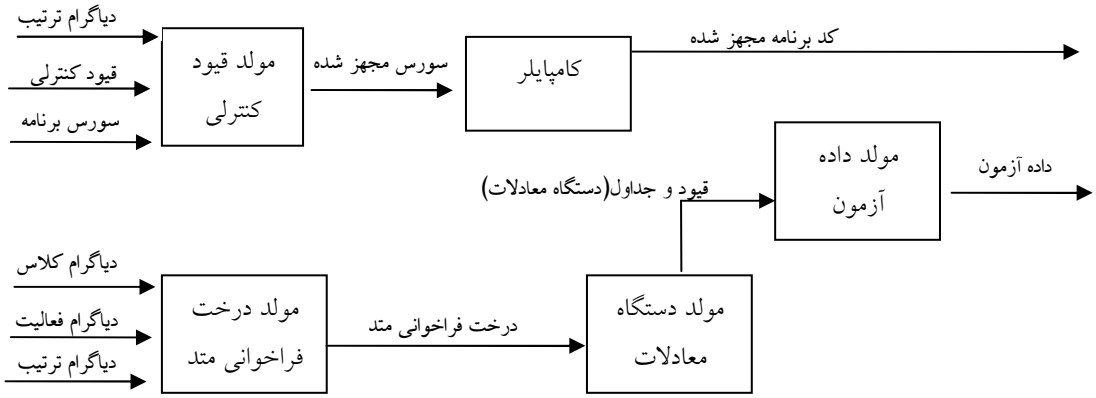
زیربخش مولد دستگاه معادلات با دریافت درخت فراخوانی متد، دستگاه معادلات برای اجرای این سناریو را تولید می‌کند. این دستگاه معادلات شرط‌هایی می‌باشند که ارضا شدن آن‌ها باعث اجرای مسیر مورد نظر در درخت فراخوانی متد و یا به عبارتی اجرای سناریوی کاربری مورد نظر می‌شود.

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.

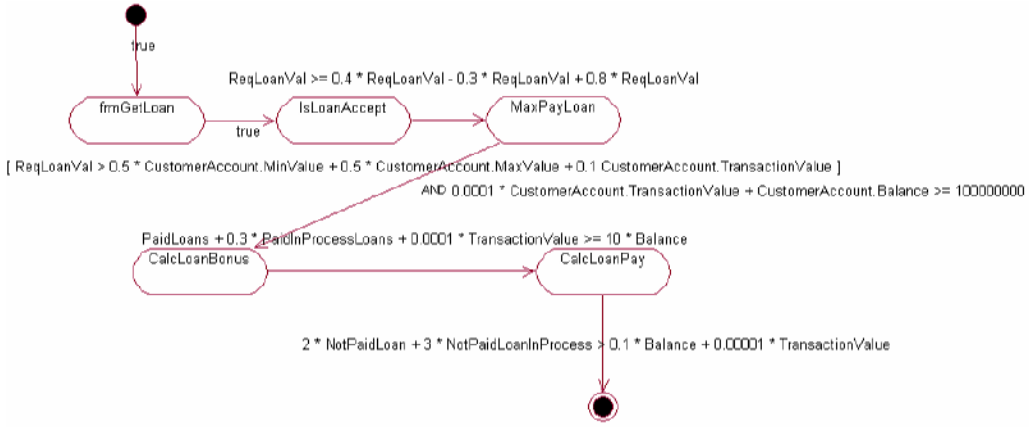
۳-۱-۱- مولد درخت فراخوانی متد

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 (event ، exit ، do ،entry) ایجاد می‌کند. گره‌های entry، do و exit به ترتیب به هم متصل می‌شود. از هرکدام 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].



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

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 ازنوع ورودی‌ها و خروجی‌های متدهای همکاری کننده و همچنین ویژگی‌های[۲۳] کلاس‌های مربوط می‌باشند. لازم به یادآوری است که داده آزمون متشکل از مقادیر ورودی متد آغاز کننده سناریو و ویژگی‌های شی‌های کلاس‌های همکاری کننده می‌باشد. به همین دلیل با استفاده از دیاگرام فعالیت متد آغازکننده که نحوه فراخوانی 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 ارایه شده است.

```
ReqLoanVal >= 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 >= 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 وارسی شود. مولد قیود کنترلی ابتدا، دیاگرام ترتیب و سورس 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 آن‌را در یک فایل کنترلی ثبت می‌کنند. 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 سورس برنامه به همراه کدهای اضافه شده، به نام سورس 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 می کنیم. برای این کار ابتدا تمام مجهول ها و ثابتهای موجود در معادله را به یک طرف برده، تمام نامعادلات را به صورت $ax+by+...\geq 0$ یا $ax+by+...>0$ تبدیل می کنیم آن گاه متغیر c را به صورت زیر به آن اضافه می کنیم.

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

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

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

پس از تبدیل دستگاه نامعادلات به دستگاه معادلات برای حل ساده دستگاه ابتدا تعداد معادلات و تعداد مجهولات را با هم برابر می کنیم. برای انجام این کار، ابتدا پارامترهای آزاد (پارامترهایی که ورودی تابع آغازکننده مورد کاربری نیستند) را به سمت راست معادلات می‌بریم، اگر بازهم 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 ویا روش حذفی می‌توان به الگوی آزمون دست یافت. اگر دستگاه معادلات ناسازگار باشد دارای جواب نمی باشد. 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 می کنیم. جواب به دست آمده الگوی آزمون برای متغیرها می باشد. با توجه به پارامترهای c_i اضافه شده برای هر متغیر یک دسته الگو پیدا می شود که با مقداردهی c_i می‌توان الگوها را به دست آورد.

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 >= 0

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

0.0001* CustomerAccount.TransactionValue
+ CustomerAccount.Balance >= 100000000

CustomerAccount.PaidLoans
+ 0.3 *  CustomerAccount.PaidInProcessLoans
+ 0.0001 * CustomerAccount.TransactionValue >=
10 * CustomerAccount.Balance

2 * CustomerAccount.NotPaidLoans
+ 3 * CustomerAccount.NotPaidInProcessLoans >
0.1 * CustomerAccount.Balance
+ 0.00001 * CustomerAccount.TransactionValue
```

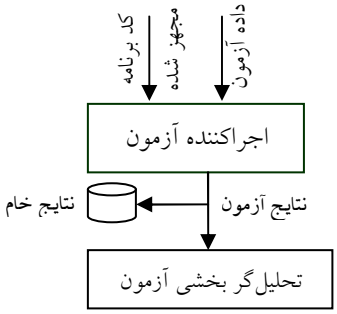
شکل (۵): الگوی تولید داده آزمون برای مسیر انتخابی در شکل شماره ۳

```
ReqLoanVal = 24398578485.54
CustomerAccount.MinValue = 346283.74
CustomerAccount.MaxValue = 7883034.64
CustomerAccount.TransactionValue = 3453037345.59
CustomerAccount.Balance = 4313464.95
CustomerAccount.PaidLoans = 45314847.34
CustomerAccount.PaidInProcessLoans = 3493.12
CustomerAccount.NotPaidLoans = 457304.39
CustomerAccount.NotPaidInProcessLoans=
998694031.23
```

شکل (۶): داده آزمون تولیدی برای مسیر انتخابی در شکل شماره ۳

۳-۲- 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 مجهزشده 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.



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 پیش‌شرط 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 اشیا 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

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 پیشهادی 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 موردهای 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 ۸و۹ ارایه شده‌اند. پیش‌شرط‌ها و پس‌شرط‌های متدها و نامتغیرهای موجود بر روی کلاس‌های نرم‌افزار نمونه در شکل شماره ۱-۹ 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 همکاری‌کننده، پیش‌شرط و پس‌شرط موجود در متدها و همچنین مقادیر نامتغیرها جهت درستی مسیر اجرای آزمون در نظر گرفتیم. با توجه به این‌که 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

ساختار مورد استفاده در مدل‌سازی برنامه که در روش‌های مبتنی بر ضوابط بررسی می‌شود خود می‌تواند به دوبخش: ۱)یکپارچگی در مدل‌سازی و ۲)میزان استفاده از روش‌های معمول مهندسی نرم‌افزار در مدل‌سازی (مانند زبان UML) تقسیم شود.

میزان خودکارسازی را می‌توان 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 کرده‌اند.

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

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 می‌باشد به عنوان سناریوی آزمون استفاده شده است.

از دید معیار مولد تولید داده آزمون، الگوی تولید داده آزمون از دستاوردها و قابلیت‌های مهم روش پیشنهادی می‌باشد که با پالایش شرط اجرای مسیر به دست می‌آید. الگوی تولید داده آزمون را هم می‌توان به منظور تولید خودکار داده آزمون مورد استفاده قرار داد و هم این الگو می‌تواند به عنوان یک مرجع مقایسه‌ای برای درستی طراحی توسط طراح مورد استفاده قرار گیرد. تنها روش پیشنهادی در بین روش‌های موجود است که به استخراج الگوی آزمون می‌پردازد. با توجه به این که الگوی تولید داده آزمون در روش پیشنهادی از تلفیق مدل‌های ایستا و پویای نرم‌افزار به دست می‌آید می‌توان به برتری آن در برابر روش‌هایی که از تبادل پیغام و یا دیاگرام همکاری (که تنها مدل‌های پویا می‌باشند) بهره می برند، اشاره کرد. همانگونه که در جدول ۱ اشاره شده، تنها روش پیشنهادی است که از این رویکرد استفاده می‌کند.

از دید معیار میزان خودکارسازی آزمون، روش پیشنهادی دارای خودکارسازی با درجه متوسط می‌باشد که درجه بالایی از خودکارسازی است. در بین روش‌های موجود تنها روش [۳] دارای این درجه از خودکار سازی است. یادآوری می گردد، با ارایه کامل پیش‌شرط‌ها و پس‌شرط‌های 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 تقریبا خودکار نمود.

از دید معیار ساختار یکپارچه مدل سازی، در روش پیشنهادی از زبان مدل‌سازی UML که استاندارد غیررسمی مدل‌سازی نرم‌افزارهای شی‌گرا است به منظور مدل‌سازی نرم‌افزار استفاده شده که به همین دلیل یکپارچگی در مدل‌سازی نیز وجود دارد. OCLنیز زبان ایجاد قید برروی اشیا است که به همراه زبان مدل‌سازی UML و به منظور ارایه امکاناتی برای توصیف بهتر مدل‌های شی‌گرا، ارایه شده است. با توجه به گسترش استفاده از UML، پژوهش‌ها بر روی روش‌های آزمون با استفاده از آن در حال گسترش است. روش‌های [۹]،[۱۰] و [۱۱] نیز به منظور مدلسازی از UML و دیگر روش‌ها از مدل‌های ضوابط جبری استفاده کرده‌اند. روش [۷] با وجود استفاده از ضوابط جبری دارای ساختار یکپارچه می‌باشد

تقریبا تمامی روش‌هایی که نیازمند به دست آوری مقادیر آزمون در حین اجرا و یا وارسی حین اجرای نرم‌افزار می‌باشند از روش مجهزسازی کد استفاده می‌کنند. 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 می‌توان به تولید داده آزمون پرداخت و بر روی دیاگرام‌های فعالیت محل اضافه کردن کدهای ثبت کننده مقادیر پیش‌شرط‌ها، پس شرط‌ها و نامتغیرها را مشخص کرد.

۵- نتیجه‌گیری

تولید داده آزمون و آزمون خودکار نرم‌افزار به منظور تعیین درستی نرم افزار از گام‌های مهم در آزمون نرم افزار می‌باشد. در صورتی که گام تولید داده آزمون، داده‌های 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 عمدتا در زمینه تولید داده آزمون به منظور ایجاد پوشش در یک زمینه خاص مانند پوشش شاخه می‌باشد. در زمینه شی‌گرایی نیز تحقیقات عمدتا بر روی تولید خودکار داده آزمون برای کلاس‌ها تکیه دارند. درصورتیکه با تکیه بر آزمون نرم افزار بر اساس سناریوهای کاربری می‌توان بسیاری از خطاهای سطح بالای نرم افزار را شناسایی کرد که این مساله منجر به شناسایی خطاهای سطح پایین‌تر خواهد شد.

در این مقاله روشی به منظور تولید خودکار داده آزمون و آزمون خودکار، برای نرم‌افزارهای شی‌گرا ارایه شده است. این روش بر اساس تولید داده برای سناریوهای کاربری طراحی شده است. در روش پیشنهادی، سناریوی کاربری در مرحله طراحی بر اساس دیاگرام‌های ترتیب مرحله طراحی دریافت شده و ضوابط هر متد همکاری کننده در سناریو از طریق OCL و دیاگرام فعالیت مشخص می‌گردد. بر اساس دیاگرام فعالیت و مسیری که برای اجرای سناریو مورد نظر، نرم افزار از آن می‌گذرد، قیود کنترلی برای ارضای سناریو مشخص می‌شود. از حل این قیود می‌توان مقادیر داده مورد نظر برای آزمون را به دست آورد.

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

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

به منظور توسعه روش پیشنهادی برای تشخیص کارکردهای کاشته شده در نرم‌افزار (به عبارت دیگر وظیفه‌مندیهای خواسته نشده از نرم‌افزار) با توجه به این‌که هر مسیر در نرم‌افزار به یک دستگاه معادلات نگاشت می‌شود، می‌توان با تحلیل نرم‌افزار و درآوردن مستندات طراحی مورد نیاز از طریق پیمایش کد به بررسی کارکردهای موجود در نرم‌افزار و کارکردهای مورد نیاز در طراحی پرداخت. به عنوان مثال با بالا بردن سطح انتزاع کد می‌توان طراحی را از روی آن حدس زده و با طراحی موجود مقایسه کرد. همچنین می‌توان 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 می‌تواند به عنوان تحقیق در آینده مورد بررسی قرار گیرد. به این منظور باید بتوان بین نسخه‌های مختلف نرم‌افزار، نگاشتی برای تولید و یا انتخاب داده‌های تولید شده قبلی ایجاد کرد.

برای خودکارسازی کامل تحلیل نرم‌افزار در این زمینه می‌بایست بررسی گسترده‌ای در مورد تعداد مقدار موارد آزمون برای پوشش آزمون مورد نظر در نرم‌افزارهای گوناگون انجام شود.

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

در روش پیشنهادی خطوطی به سورس نرم‌افزار به منظور ثبت مقادیر پیش‌شرط‌ها، پس‌شرط‌ها و نامتغیرها اضافه می‌شود. این خطوط به صورت غیر فعال عمل می‌کنند به این معنا که در روند اجرای برنامه دخالتی نکرده و فقط مقادیر مورد نیاز را در فایل‌های کنترلی ثبت می‌کنند. با فعال کردن این خطوط می‌توان به وارسی حین اجرای برنامه پرداخت. برای این منظور به هر کدام از شرط‌ها و متغیرها یک درجه اهمیت نسبت داده می‌شود که این درجه اهمیت به عنوان مثال می‌تواند متوسط، مهم و بسیار مهم باشد. با توجه به این درجه اهمیت در حین اجرای برنامه واکنش لازم صورت می‌گیرد. به عنوان مثال در صورت نقض شرط‌های بسیار مهم علاوه بر ثبت آن در فایل کنترلی می‌توان از کاربر درخواست کرد که آیا مایل به ادامه اجرای برنامه می‌باشد یا خیر.

رویکرد جنبه‌گرا[۲۱]، رویکردی نوین در تولید نرم‌افزار می‌باشد که با توسعه رویکرد شی‌گرا سعی در بهبود روش‌های تولید نرم‌افزار دارد. یکی از جنبه‌های مهم این رویکرد جداسازی قطعات بر اساس کارکرد و سپردن هر کارکرد به یک قطعه خاص می‌باشد. با استفاده از این رویکرد در سیستم پیشنهادی می‌توان بدون اضافه کردن کدهای مورد نیاز به برنامه به صورت فیزیکی، با تهیه قطعه‌ای جدا و با فراخوانی متدها به صورت خودکار عمل ثبت مقادیر پیش‌شرط‌ها، پس‌شرط‌ها 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

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 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

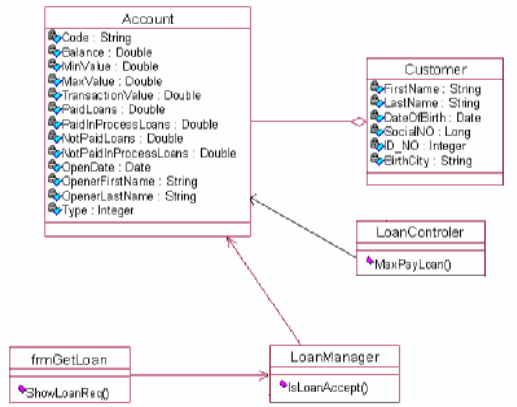
نرم‌افزاری بر اساس روش پیشنهادی شرح داده شده در مقاله تهیه شده است. در این بخش یک مثال را به منظور بررسی نتایج حاصل از آزمون نرم‌افزار با استفاده از نرم‌افزار تولید شده، ارایه می‌کنیم. کلیه ضوابط UML مورد نیاز از طریق فایل مدل Rational Rose دریافت شده و اطلاعات مربوط به ضوابط OCL نیز به صورت دستی در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

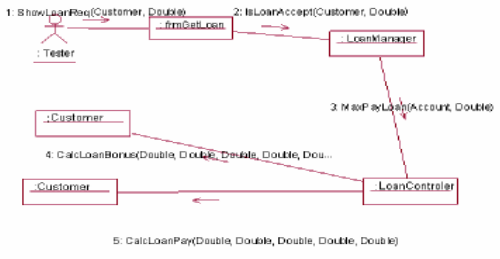
(مشتری) ویژگی‌های FirstName، LastName و SocialNo به ترتیب نمایانگر نام، نام خانوادگی و شماره ملی مشتری و متدهای CalcLoanBonus و CalcLoanPay با نوع برگشتی double به ترتیب محاسبه جوایز بر اساس سوابق و محاسبه جریمه بر اساس دیرکردها را برعهده دارند. در کلاس LoanControler متد MaxPayLoan با نوع بازگشتی وظیفه محاسبه حداکثر وام دریافتی را برعهده دارد. در کلاس Account (حساب بانکی) نیز Code، Balance، MinValue، MaxValue، OpenDate، TransactionValue، NotPaidInProcessLoans، PaidLoans، PaidInProcessLoans و Type 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): دیاگرام مورد کاربری

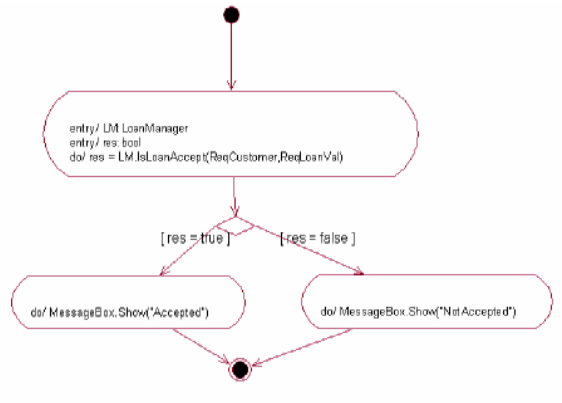


شکل (الف-۲): نمودار کلاس مورد کاربری

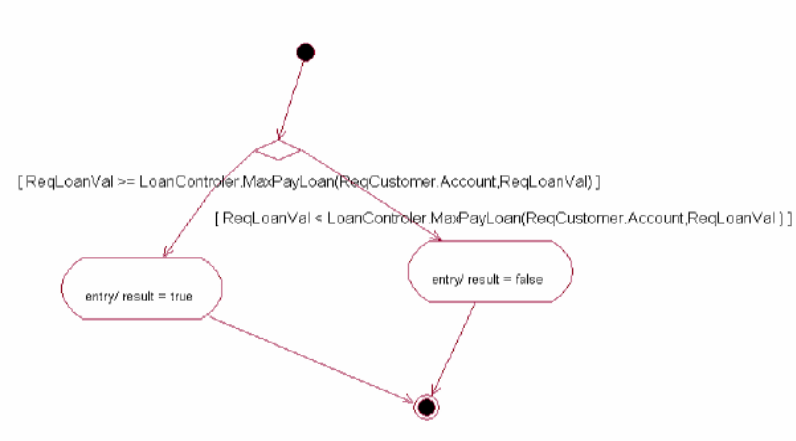
دیاگرام همکاری مربوط به سناریوی شکل شماره ۱-۱ در شکل شماره ۱-۳ نمایش داده شده است. همان‌گونه که در این شکل مشخص است، ابتدا کاربر درخواست وام می‌کند و مقدار وام درخواستی با هویت مشتری به متد کنترل کننده فرستاده می‌شود. این متد با ارسال پیغامی محاسبه بیشترین مقدار وام قابل دریافت توسط شخص مورد نظر را درخواست 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. دیاگرام‌های فعالیت مربوط به متدهای همکاری کننده در شکل‌های شماره ۱-۴ تا۱-۸ نمایش داده شده‌اند. شکل شماره ۱-۹ هم بیانگر قیود کنترلی ورودی می‌باشد.



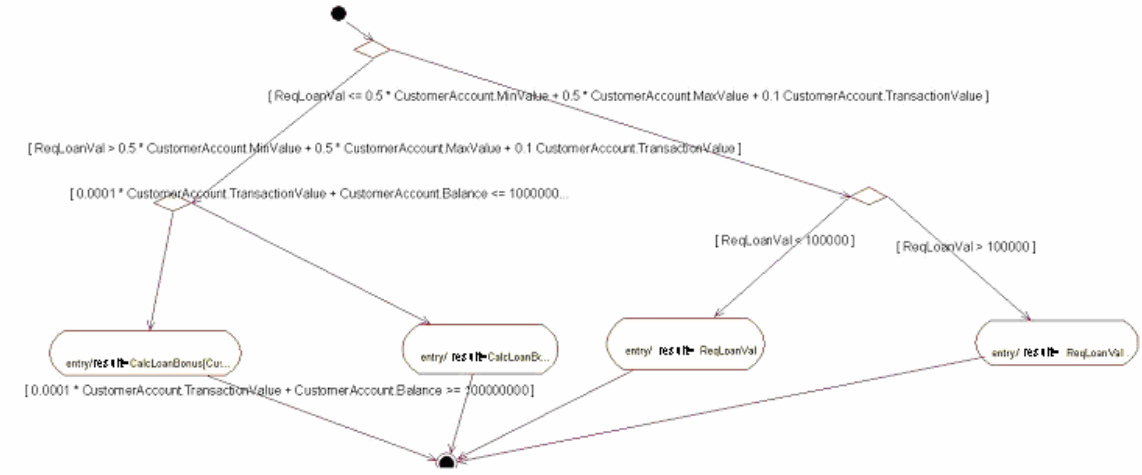
شکل (الف-۳): دیاگرام همکاری سناریوی مورد آزمون



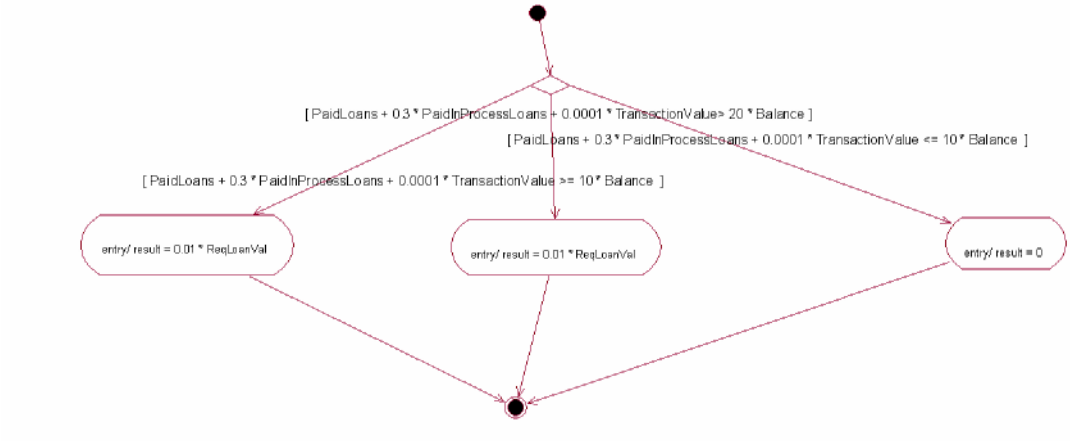
شکل (الف-۴): دیاگرام فعالیت متد ShowLaonReq



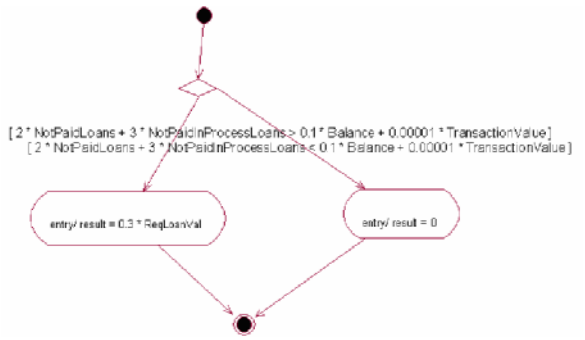
شکل (الف- ۵): دیاگرام فعالیت متد IsLoanAccept



شکل (الف-۶): دیاگرام فعالیت متد MaxLoanPay



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



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

CustomerAccount.Balance >=0
CustomerAccount.MinValue >=0
CustomerAccount.MaxValue >=0
CustomerAccount.NotPaidLoans >=0
CustomerAccount.PaidLoans >=0
CustomerAccount.MaxValue >= CustomerAccount.MinValue
CustomerAccount.MaxValue >= CustomerAccount.Balance
CustomerAccount.Balance >= CustomerAccount.MinValue

شکل (الف- ۹): قیود موجود برروی کلاس ها

زیرنویسها

[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