رویکردی مدل-رانه برای خودکارسازی آزمون رگرسیون با استفاده از تبدیل مدل افزایشی

مریم نورائی آباده[۱]، دکتری؛ سید حسن میریان حسین‌آبادی[۲]، دانشیار

۱- گروه مهندسی کامپیوتر- واحد علوم و تحقیقات - دانشگاه آزاد اسلامی- تهران- ایران-mnooraei@ iauabadan.ac.ir
۲- دانشکده مهندسی کامپیوتر- دانشگاه صنعتی شریف- تهران- ایران-hmirian@sharif.ac.ir

چکیده: پیچیدگی سیستم‌های نرم‌افزاری و وابستگی جوامع به این سیستم‌ها رو به افزایش است. با توسعه فن‌آوری‌های تحت وب و رویکردهای سرویس‌گرا، ضرورت تطبیق با نیازهای کاربران در هنگام درخواست برای اِعمال تغییرات و تکامل سیستم‌ها، بر این پیچیدگی افزوده است. روش‌های توسعه نرم‌افزار مدل-رانه با تمرکز بر استفاده از مدل به‌عنوان مصنوع اصلی و به‌کارگیری رویکردهای خودکار، توسعه محصولات نرم‌افزاری دارای کیفیت بالا را وعده داده است. هدف از این مقاله ارائه رویکردی مدل-رانه برای انتخاب خودکار زیرمجموعه مناسب از موارد آزمون برای آزمون رگرسیون مبتنی بر مدل با استفاده از انتشار تغییرات و تبدیل مدل افزایشی است. استفاده از تبدیل مدل افزایشی امکان انتشار خودکار تغییرات مدل و درنهایت انتخاب مجموعه موارد آزمون سازگار جهت انجام آزمون بعد از تغییرات را در سطح انتزاعی فراهم می‌آورد. دقت و کارایی چارچوب پیشنهادی با معرفی معیارهای بسندگی جدیدی بر اساس مدل تغییرات بر روی سه موردمطالعه ارزیابی و تحلیل شده است. از مزایای این روش تخمین زودهنگام میزان تلاش برای تجزیه و تحلیل تأثیر تغییرات، کاهش هزینه آزمون رگرسیون مستقل از سکو و انتخاب زیرمجموعه مناسب برای آزمون رگرسیون به منظور تشخیص زودهنگام خطای تولید محصول نرم‌افزاری است.

واژگان کلیدی: توسعه مدل-رانه، آزمون رگرسیون، تبدیل مدل افزایشی، مدل تغییرات، سازگاری، معیار پوشش.

A Model Driven Approach to Automate Software Regression Testing Using Incremental Model Transformation

M. Nooraei Abadeh[1]; S. H. Mirian Hosseinabadi[2]

1- Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran, Email: mnooraei@ iauabadan.ac.ir
2- Department of Computer Science, Sharif University of Technology, Tehran, Iran, Email: hmirian@sharif.ac.ir

Abstract: The increase in complexity and the rate of technological changes in modern software development have led to a demand for systematic methods that raise the abstraction level for system maintenance and regression testing. Model Driven Engineering (MDE) has promised to reduce extra coding efforts in software maintenance activities using traceable change management, especially in rapidly changing application. The paper presents a Z-notation based framework, called Changed-based Regression Testing (ChbRT), for formal modeling of regression testing in the context of MDE. The framework proposes to automatically propagate the changes from a software specification to testing artifacts in order to preserve consistency after system evolution. The framework is enriched by providing a new category of coverage metrics for change-based regression testing. The proposed framework is expected to be beneficial in both platform independent and specific levels of ChbRT by identifying the suitable coverage according to available testing resources. The accuracy and efficiency of the proposed framework have been evaluated and analyzed on three case studies.

Keywords: Model Driven Development**,** Regression Testing, Incremental Model Transformation, Change Model, Consistency, Coverage Criteria.

تاریخ ارسال مقاله: ۱۳۹۶/۰۹/۲۶
تاریخ اصلاح مقاله: ۱۳۹۶/۱۲/۲۲
تاریخ پذیرش مقاله: ۱۳۹۷/۰۳/۲۷
نام نویسنده مسئول: ، سید حسن میریان حسین‌آبادی
نشانی نویسنده مسئول: ایران - تهران - دانشگاه صنعتی شریف - دانشکده مهندسی برق و کامپیوتر.

۱- مقدمه

امروزه رویکردهای توسعه نرم‌افزار به دنبال افزایش سطح تجرید در راستای کاهش پیچیدگی‌های برنامه‌های کاربردی هستند[۱، ۲]. مهندسی مدل-رانه یا توسعه نرم‌افزار مدل-رانه روشی برای توسعه نرم‌افزار با تاکید بر استفاده از مدل‌ها به‌منظور توصیف، توسعه، تحلیل، ارزیابی و مدیریت تغییرات سیستم‌های نرم‌افزاری است. استفاده از تبدیل مدل به‌منظور به دست آوردن مصنوعات جدیدی مانند مدل‌هایی در سطوح متفاوتی از انتزاع و همچنین کدهای اجرایی از مصنوعات قبلی، از ویژگی‌های کلیدی این روش توسعه است. مهندسی مدل-رانه افزایش قابلیت جابجایی بر سکوهای مختلف، ایجاد قابلیت ردیابی بین مصنوعات متعدد، کاهش هزینه‌های دستی، توسعه نرم‌افزار به‌صورت خودکار و بهبود کیفیت محصولات نرم‌افزاری را وعده داده است[۳].

استفاده از رویکرد توسعه مدل-رانه و مزایای آن، « آزمون نرم‌افزار» را نیز وارد مرحله جدیدی کرده که منجر به افزایش بهره‌وری از مدل‌ها برای آزمون زودهنگام نرم‌افزار شده است. در این مقاله برای اولین بار روش نوینی به‌منظور انتخاب زیرمجموعه مناسب برای آزمون رگرسیون مدل-رانه با استفاده از تبدیل مدل افزایشی ارائه شده است. استفاده از رویکرد آزمون مبتنی بر مدل در کنار تبدیل خودکار مدل‌ها، امکان تولید خودکار کدهای آزمون وابسته به پیاده‌سازی با استفاده از مدل‌های آزمون مستقل از سکو را فراهم می‌کند. یکی از چالش‌های آزمون مبتنی بر مدل با دیدگاه مدل-رانه حفظ ارتباط بین مصنوعات متفاوت مراحل آزمون است. این مقاله از ایجاد ارتباط بین فرامدل‌ها برای ردیابی ارتباط بین مصنوعات مختلف سیستم در فازهای گوناگون استفاده کرده است. تبدیل مدل افزایشی و تاریخچه تغییرات دو مفهوم جدیدی هستند که در این مقاله از آن‌ها برای مدیریت تغییرات و آزمون رگرسیون مدل-رانه استفاده خواهیم کرد.

تعیین معیارهای بسندگی[۱] یکی از نیازمندی‌های حیاتی فرایند آزمون نرم‌افزار است. این معیارها به‌عنوان قوانینی برای ارزیابی قابلیت اطمینان موارد آزمون مشتق شده از مدل‌ها و به‌عنوان پیشگو برای تعیین زمان مناسب متوقف کردن آزمون استفاده می‌شوند[۴]. ازاین‌رو، هنگامی‌که مورد آزمون وابسته به پیاده‌سازی، اجرا می‌گردند، اندازه‌گیری پوشش[۲] به‌دست‌آمده با توجه به مدل (به‌جای کد) دارای اهمیت بالایی است. این موضوع چالش‌های جدیدی را برای فعالیت‌های آزمون به وجود آورده است که می‌توان یک‌راه حل آن را ایجاد و نگهداری روابط بین عناصر مدل و کد در آزمون دانست. از قابلیت‌های این رویکرد، امکان اندازه‌گیری پوشش مدل به‌دست‌آمده بر اساس تاریخچه تغییرات است. درنهایت با اجرای موارد آزمون به‌روزرسانی شده به کمک تبدیل‌های مدل-رانه، آزمون رگرسیون خودکارسازی خواهد شد. رویکرد فراهم شده در این مقاله، مجموعه‌ای از قابلیت‌ها را برای توسعه‌دهندگان نرم‌افزار در راستای انجام وظایف زیر ارائه می‌کند:

- ایجاد و ویرایش مدل‌های اجزای نرم‌افزار. توسعه‌دهندگان می‌توانند از مدل ایجاد شده توسط ابزار مربوطه به‌عنوان پایه‌ای برای آزمون در دامنه‌های کاربردی مختلف استفاده کنند. طراح آزمون می‌تواند به‌منظور کشف مشکلات طراحی قبل از آزمودن سطح پیاده‌سازی، عملکرد محصول انتزاعی را اصلاح و ارزیابی کند.
- تولید خودکار مجموعه آزمون برای آزمون رگرسیون. چارچوب پیشنهادی با استفاده از معیارهای جدید پوشش مبتنی بر مدل تغییرات، موارد آزمون رگرسیون را به‌صورت خودکار تولید می‌کند. پوشش به‌دست‌آمده توسط این روش، برتری آن را نسبت به روش‌های مشابه مبتنی بر مدل نشان می‌دهد.
- اجرای خودکار مجموعه آزمون برای مصنوعات نرم‌افزار. این چارچوب شامل موتور خودکار جهت اجرای موارد آزمون است که به‌طور نتایج مشاهده شده را با نتایج پیش‌بینی‌شده توسط مدل مقایسه می‌کند.

بدون داشتن توصیف‌های دقیقی برای آزمودن ویژگی‌ها و رفتارهای ایستا و پویای مدل‌ها، صحت و اعتبار سیستم‌های نرم‌افزاری مورد تردید خواهند بود. استفاده از توصیف‌های صوری باوجود پیچیدگی‌های آن‌ها، دقیق‌ترین روش برای توسعه تئوری‌ها و رویکردهای بدون ابهام و دقیق در سطوح انتزاعی است. برای اعتبارسنجی رویکرد این مقاله و ویژگی‌هایی مانند سازگاری، تمامی توصیف‌های این مقاله با استفاده از زبان Z [۵] توصیف و در محیط Z/EVES اعتبارسنجی و اثبات شده‌اند. Z/EVES [۶] یک سیستم محاوره‌ای برای ایجاد، تحلیل و اثبات توصیف‌های Z است.

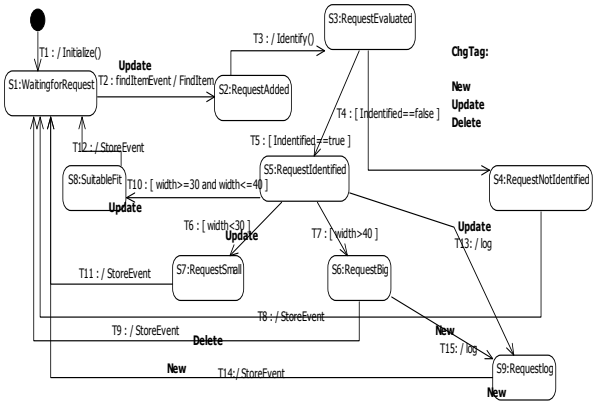
در این مقاله بعد از بررسی انگیزه کار با استفاده از یک مثال در بخش دوم، به توصیف صوری چارچوب پیشنهادی در بخش سوم می‌پردازیم. بخش چهارم به معناشناسی آزمون مبتنی بر تغییرات اختصاص یافته است. دسته‌بندی پیشنهادی برای موارد آزمون رگرسیون در بخش پنجم بررسی می‌شود. نتایج تحلیل و ارزیابی روش پیشنهادی در بخش ششم ارائه می‌شوند. بررسی تحقیقات مرتبط و نتیجه‌گیری نیز بخش‌های پایانی را تشکیل می‌دهند.

۲- بررسی چارچوب با استفاده از یک مثال

ماشین حالت رفتار یک درخواست برای خدمات تحت وب را که با استفاده از UML2 مدل شده است در نظر بگیرید (شکل ۱). همان‌طورکه دیده می‌شود، Si ها حالات سیستم و Ti ها گذار بین حالات را نشان می‌دهند. این درخواست، پس از دریافت هر درخواست از حالت بیکار (حالت S1:WaitingforRequest) خارج شده و عملیات پردازش درخواست را شروع می‌کند (حالت S3:RequestEvaluated). در صورت عدم رخداد وقفه در عملیات، درخواست خدمات با موفقیت به پایان می‌رسد. درغیر این‌صورت، اگر خطا و یا وقفه‌ای در سیستم رخ دهد ممکن است سرویس لغو شده (S4:RequestNotIdentified) و یا

بعد از برطرف شده عامل وقفه از حالت تعلیق (حالت S5:RequestIdentified) به پردازش ادامه دهد. فرض کنید مدتی بعد از پیاده‌سازی، طراح تصمیم به تغییر رفتار این سیستم می‌گیرد. به‌این‌ترتیب که وضعیت هر درخواست (موفق و یا ناموفق) با ضبط تاریخچه‌ای گزارش داده شود. همان‌طورکه دیده می‌شود، تغییرات با برچسب New، Delete و Update مشخص شده‌اند.

با هر تغییر در این مدل (مدل مبدأ)، لازم است برای صحت‌سنجی عملکرد مدل تغییریافته، نیازمندی‌های آزمون (مدل مقصد) به‌روز شده‌ای ایجاد گردد و همچنین معیارهای پوشش مناسبی بسندگی مجموعه آزمون به‌روز شده را بررسی کنند. رویکرد چارچوب این مقاله، انتشار خودکار و افزایشی تغییرات از مدل رفتاری مستقل از سکو به مدل موارد آزمون است. راهکارهای جدید ارائه شده در این پژوهش برای تحقق عملی آزمون رگرسیون مدل‌رانه عبارت‌اند از:



شکل۱: رفتار یک درخواست برای خدمات تحت وب

- ایجاد سازگاری خودکار بین مصنوعات مختلف فرایند توسعه نرم‌افزار (طراحی تا آزمون) با ایجاد روابط پیگیری بین مدل‌ها در فازهای مختلف توسعه بر اساس فرامدل‌های خوش‌تعریف.
- ردیابی خودکار خطاها با استفاده از روابط پیگیری رو به عقب، که آن‌ها را لینک‌های ردیابی خطا می‌نامیم، برای پاسخ به این پرسش‌ها که: چه عناصری سبب خطا شده‌اند؟ و یا چه عناصری به این مسبب‌های خطا وابسته و درنتیجه تأثیرپذیر از خطا هستند؟
- استفاده از روابط پیگیری روبه‌جلو، که آن‌ها را لینک‌های نگهداری می‌نامیم، برای پاسخ به این پرسش‌ها که چه عناصری تحت تأثیر یک تغییر قرار می‌گیرند و یا چه عناصری ممکن است از این عنصر تأثیر بگیرند؟
- خلق معیار پوشش آزمون مبتنی بر ساختار مدل برای اجرای زودهنگام آزمون به‌ویژه برای سیستم‌های بحرانی- ایمن[۳] (در حال حاضر هیچ روشی برای تجزیه و تحلیل پوشش برای سطح طراحی (سطح مستقل از سکو) مطرح نشده است).
- ارائه زبان‌های پرس‌وجو برای پوشش الگوهای خاص در سطح طراحی و پیاده‌سازی.

۳- چارچوب پیشنهادی به زبان Z

در محیط فرامدل‌سازی MOF، هر مدل الزاماً با فرامدل خود مطابقت دارد. در چارچوب آزمون رگرسیون مدل-رانه، ماشین‌های حالت UML2 و تئوری گراف‌های دارای صفت و نوع[۴] [۷] با یکدیگر یکپارچه شده‌اند و مدلی به نام ماشین حالت برچسب دار[۵] (LSM) به‌عنوان مدل رفتاری آزمون تعریف شده است. البته صوری‌سازی در این مقاله به‌گونه‌ای انجام شده که قابلیت انتقال بین دامنه‌های مدل‌سازی متفاوتی را فراهم کند. از LSM برای استخراج موارد آزمون انتزاعی استفاده می‌شود؛ ازاین‌رو، آن را مدل آزمون نیز می‌نامیم.

تعریف ۱ (مدل آزمون). یک مدل آزمون، رفتار یک سیستم را در قالب یک ماشین حالت برچسب‌دار نشان می‌دهد که آزمون مبتنی بر مدل از آن برای استخراج نیازمندی‌های آزمون استفاده می‌کند. هنگامی‌که این مدل در یک حالت فعال قرار داشته می‌تواند با رخداد یک رویداد درحالی‌که پیش‌شرایط رویداد را برآورده می‌کند، گذار مربوطه را سپری کرده، تأثیر خود را بر سیستم گذشته و وارد حالت جدیدی شود. یک مدل آزمون با یک شِما، با نام *IndependentTestModel* تعریف می‌شود که عناصر آن مجموعه‌های محدودی از نوع آزاد *TestTemplate* هستند. این نوع آزاد به‌عنوان فرامدل برای تعریف عناصر یک مدل آزمون مطرح می‌شود. بنابراین هر مدل آزمون یک مدل نمونه است که عناصر آن مطابق با *TestTemplate* تعریف می‌شوند. این فرامدل شامل *STATE*، *TRANSITION*، *EVENTT*، *GCON* و *EFFECT* می‌باشد. دانه‌بندی فرامدل *TestTemplate* در سطح ریزدانه تنظیم شده است تا امکان کشف خطا در آزمون بعد از تغییرات و همچنین قابلیت کشف محل خطا را افزایش دهد. برای نام‌گذاری عناصر هر مدل از یک شناسه یکتا *ID* استفاده می‌شود. برای کنترل تغییرات حالت ایستای مدل بر اساس تئوری گراف‌های دارای صفت و نوع، یک نوع *typeID* و مجموعه‌ای از ویژگی‌ها به همراه مقادیر آن‌ها، *attinstanseset*، توسط توابع *type* و *value* به هر یک از عناصر مدل نسبت داده می‌شود. در این سطح از انتزاع *ID* و *Value* را به‌عنوان مجموعه‌های داده شده تعریف می‌کنیم.

[ID ,Value]

TestTemplate ::= *STATE* 《*ID*》 | *TRANSITION* 《*ID*》
| *ACTION* 《*ID*》 | *GCON* 《*ID*》 | *EFFECT* 《*ID*》

در مدل آزمون دو رابطه *FromSource* و *ToTarget* تعریف شده است که به ترتیب حالت مبدأ و مقصد یک گذار را نشان می‌دهند. مجموعه عناصر خاص یک مدل آزمون بر اساس تعریف فرامدل با مجموعه‌های *State*، *Transition*، *ACTION*، *GuardCon* و *Effect* تعریف می‌شوند. مطابق تعریف فوق، هر نمونه از مدل آزمون شامل مجموعه حالت، گذار، رویداد، شرایط و تأثیرات بر روی مدل است که با Stt، Trans، Eve، Gaurdcon و EffectT توصیف می‌شود. بخش گزاره شِمای مدل آزمون، شرایط و محدودیت‌های دامنه و برد روابط معرفی

شده را بیان می‌کند. همه عناصر یک مدل آزمون مجموعه‌ای به نام TotalElem را تشکیل می‌دهند که برای پیگیری و تحلیل تغییرات روی مدل مورد استفاده قرار می‌گیرند. برای مدیریت تغییرات در مدل آزمون از برچسب تغییرات *ChgTag* استفاده می‌شود. درنهایت، رابطه *labelE* در توصیف مدل آزمون برای انتساب یک برچسب مناسب شامل رویداد، شرایط محافظ و تأثیر هر رویداد برای یک گذار استفاده می‌شود.

$State == \text{ran } STATE$
$Transition == \text{ran } TRANSITION$
$ACTION == \text{ran } EVENT$
$GuardCon == \text{ran } GCON$
$Effect == \text{ran } EFFECT$
$typeID == ID$
$attrID == ID$
$attinstanseset == attrID \twoheadrightarrow Value$
$ChgTag ::= New \mid Update \mid Delete$

IndependentTestModel

$TotalElem: \mathbb{P}\ TestTemplate$
$Stt: \mathbb{F}\ State$
$Eve: \mathbb{F}\ Event$
$Guardcon: \mathbb{F}\ GuardCon$
$EffectT: \mathbb{F}\ Effect$
$Trans: \mathbb{F}\ Transition$
$FromSource: Transition \rightarrow State$
$ToTarget: Transition \rightarrow State$
$type: TestTemplate \nrightarrow typeID$
$value: TestTemplate \nrightarrow attributeset$
$labelE: Transition \leftrightarrow TestTemplate$
$TagFunc: TestTemplate \nrightarrow ChgTag$

$\forall trans: Trans \bullet \exists s_1, s_2: Stt \bullet trans \mapsto s_1 \in ToTarget \wedge trans \mapsto s_2 \in FromSource$
$\text{ran } ToTarget \subseteq Stt;\ \text{ran } FromSource \subseteq Stt\ ;\text{ran } labelE = EffectT \cup Eve \cup Guardcon;\ \text{dom } ToTarget = Trans$
$\text{dom } FromSource = Trans;\text{dom } labelE \subseteq Trans$
$TotalElem = Stt \cup Trans \cup Eve \cup Guardcon \cup EffectT$

تعریف ۲ (همگام‌سازی مدل). دو فرامدل مبدأ و مقصد SM و TM و همچنین رابطه سازگاری بین این دو مدل را در نظر بگیرید. یک همگام کننده چند سطحی چند جهته روی این مدل با نام xDML[۶] با استفاده از ConsistentR تعریف می‌شود. نوع دو جهته این همگام کننده که BiDiML نامیده می‌شود، دو مدل مبدأ و مقصدی را که در رابطه ConsistentR صدق می‌کنند و همچنین مدل تغییرات را دریافت می‌کند و مدل تغییرات را چنان روی مدل‌های ورودی اِعمال می‌کند که مدل‌های مبدأ و مقصد همچنان سازگار باقی بمانند. تبدیل‌های هماهنگ‌کننده پیشرو FDML و عقب‌گرد BDML به ترتیب تغییرات مدل مبدأ را به مدل مقصد و تغییرات مدل مقصد را به مدل مبدأ چنان اعمال می‌کنند که مدل‌های خروجی همچنان هماهنگ باقی بمانند. تعاریف صوری همگام کننده مدل در ادامه آمده است:

$[\ SM\ ,\ TM\ ]$
$ConsistentR ::= BasicML\ 《SM \times TM》$
$\mid FDML\ 《ConsistentR \times \mathbb{P}\ SM》$
$\mid BiDiML\ 《ConsistentR \times \mathbb{P}\ SM \times \mathbb{P}\ TM》$
$\mid \text{BDML}\ 《\text{ConsistentR} \times \mathbb{P}\ \text{TM}》$

تعریف ۳ (تبدیل‌های نگهدارنده سازگاری). یک تبدیل مدل نگهدارنده سازگار است اگر یک مدل مبدأ را چنان به مدل مقصد تبدیل کند که سازگاری قبل از تغییرات بین آن‌ها نقض نشود. اثبات برخی ویژگی‌های تبدیل مدل مانند سازگاری در مرجع [۸] ذکر شده است. نوع آزاد ConsistentR چنین تبدیلاتی را نشان می‌دهد.

$\text{ConsistentR} ::= \text{BaseMT}\ 《\text{ConsistentR}》$
$\mid \text{UpdateTarget}\ 《\text{ConsistentR} \times \text{DirectDeltaTrans}》$
$\mid \text{Sync}\ 《\text{ConsistentR} \times \text{DirectTransSource} \times \text{DirectTransTarget}\ 》$
$\text{DirectDeltaTrans} ::= \text{DirectTransSource}\ 《\text{TM} \times \mathbb{P}\ \text{TM}》$
$\mid \text{DirectTransTarget}\ 《\text{SM} \times \mathbb{P}\ \text{SM}》$

DirectDeltaTrans یک تبدیل مدل مستقیم است که یک مدل ورودی (مدل مبدأ یا مقصد) و یک مدل تغییرات را به‌عنوان ورودی دریافت می‌کند و مدل به‌روز شده را منطبق با همان فرامدل اولیه برمی‌گرداند. UpdateTarget یک تبدیل انتشار دهنده تغییرات یک جهته و Sync یک تبدیل انتشار دهنده دو جهته را نشان می‌دهند.

۳-۱ - مدل تغییرات (دلتا)

از نظر ساختاری، یک مدل از تغییرات با دو تکنیک اصلی قابل نمایش است: مدل تغییرات مستقیم و مدل تغییرات متقارن. اولی تغییرات را به‌عنوان دنباله‌ای از عملیات روی یک مدل سازماندهی می‌کند درحالی‌که دومی برآیند عملیات تغییر را روی مدل به‌عنوان اختلاف مجموعه بین دو نسخه مقایسه شده نشان می‌دهد. در این مقاله، تکنیک تغییرات مستقیم و همچنین نوع توسعه یافته آن، تغییرات افزایشی، مورد نظر است.

تعریف ۴ (تغییر). یک تغییر به‌عنوان یک گذار در راستای بهبود و تکامل یک سیستم از زیرمجموعه‌ای از عناصر مدل که پیش‌حالت نامیده می‌شود، به مجموعه دیگری در همان فرامدل که پس‌حالت نامیده می‌شود، تعریف می‌گردد. ابعاد و پارامترهای متفاوتی مانند اندازه تغییر، نوع تغییر، زمان تغییر و ... برای توصیف تغییرات وجود دارد که در پژوهش‌هایی مانند [۹] و [۱۰] به‌خوبی بیان شده‌اند. در این مقاله ابعاد مورد نظر برای هر تغییر عبارت‌اند از: نوع تغییر، ترتیب تغییرات و عنصر تغییر یافته. یک مدل تغییر که همه این ابعاد را برای هر درخواست تغییر در نظر می‌گیرد با *ChgModel* نام‌گذاری می‌شود.

تعریف ۵ (عناصر دلتا). برای هر مدل دلتا، سه مجموعه عنصر برای پوشش دادن انواع تغییرات در نظر می‌گیریم: مجموعه عناصر درج شده یا عناصری که در حالت کنونی مدل وجود ندارند و پس از اجرای قوانین مربوطه، به مدل اضافه می‌شوند، مجموعه عناصر حذف شده یا عناصری که در حالت کنونی مدل وجود دارند و پس از اجرای قوانین

مربوطه، از مدل حذف می‌شوند و عناصر به‌روز شده یا عناصری که پس از اجرای قانون تغییر خواهند کرد. عناصر به‌روز شده می‌توانند بر اساس تغییر ویژگی‌هایشان بازتولید شوند.

تعریف ۶ (عمل دلتا). یک عمل دلتا یک تبدیل مدل مستقیم[۷] است برای بازتولید مدل‌ها در همان نحو انتزاعی. به همین دلیل در ابتدای نام آن‌ها از پیشوند MT استفاده می‌شود. در واقع منشأ تغییرات عملیات دلتا هستند. عملگرهای دلتا به دو دسته اصلی تقسیم می‌شوند: عملگر افزایش"Add" و عملگر حذف"Delete". نوع دیگری از عملگرهای دلتا عمل به‌روزرسانی "Update" است که در واقع دنباله‌ای از عملیات حذف و افزایش روی یک عنصر مشخص در نظر گرفته می‌شود. هر عمل دلتا دارای پیش‌شرایط و پس‌شرایط است. هنگامی یک تبدیل مدل اجرا می‌شود که پیش‌شرایط آن تغییر برآورده شده باشد. در توصیف چارچوب این مقاله، متغیرهای پس‌شرط دارای نام مشابه با متغیر پیش‌شرط هستند و فقط علامت(') به آن‌ها متصل می‌شود. در مدل دلتا می‌توان از الگوهای پیچیده در پیش و پس‌شرایط و رویکردهای انطباق الگو برای بررسی تطبیق آن‌ها بهره برد. همان‌طورکه بیان شد مدل دلتا به‌صورت دنباله‌ای از تبدیلات، یک مدل آزمون را به مدل آزمون به‌روز شده تبدیل می‌کند. اعمال تغییرات به‌صورت دسته‌ای بر اساس کل مدل دلتا و یا به‌صورت افزایشی[۱۱] بر اساس هر رکورد تغییرات اعمال شود.

برای عملیات دلتا MTOpi که $i \geq 1$ به‌صورت غیر صوری می‌توان نوشت:

$$\text{DeltaPreCon} = \Lambda_{i=1}^{n} \text{Pre-condition}(\text{MTOp}_i)$$

$$\text{DeltaPostCon} = \Lambda_{i=1}^{n} \text{Post-condition}(\text{MTOp}_i)$$

به‌عبارت‌دیگر انطباق یک مدل دلتا به معنای انطباق همه پیش‌شرایط آن در مدل اولیه آزمون و تأمین همه پس‌شرایط آن در مدل تغییر یافته آزمون است. همچنین، عملیات دلتای ناامن به‌عنوان عملیاتی تعریف می‌شوند که پیش‌شرایط آن‌ها در یک مدل برآورده نشده باشد، از این جهت آن‌ها نمی‌توانند به هیچ ترتیبی اجرا شوند.

هر عمل دلتا شامل پیش‌شرایط، پس‌شرایط و لیستی از اعمال قابل اجرا بر روی متغیرهای تعریف شده در عمل می‌باشد. مطابق با حساب پالایش در Z، یک عمل دلتا مطابق با تعریف DeltaOperation قابل‌تعریف است.

DeltaOperation عملی را تعریف می‌کند که در حالتی که شرایط اولیه pre-conditionElem را برآورده می‌کند آغاز شده و در حالتی که شرایط post-conditionElem در سیستم برقرار می‌شود خاتمه می‌پذیرد. متغیرهایی که تحت تأثیر تغییرات قرار می‌گیرند در chgSubjectModelElements لیست شده‌اند.

تعریف ۱.۶ (عمل افزودن به مدل دلتا). عمل افزایشی در دلتا تبدیل مدلی است به شکل Add: IndependentTestModel → IndependentTestModel' که برای هر مدل آزمون دارای عناصر IndependentTestModel < Stt, Eve, Guardcon, EffectT, Trans>، مدل آزمون به‌روز شده IndependentTestModel' را چنان ایجاد می‌کند که هرکدام از مجموعه عناصر قبل از تغییرات زیرمجموعه‌ای از مجموعه عناصر بعد از تغییرات است.

این عمل یک الگوی گراف با تطبیق عنصر جدید در پس‌شرایط را توصیف می‌کند. به‌منظور فراهم آوردن یک‌راه حل کلی در پرس‌وجوی اطلاعات در مدل به‌روز شده، تابع TagRelation یک برچسب "New" به عناصر جدید نسبت می‌دهد.

AddStt

Δ*IndependentTestModel*
s?: State
v?: Value
attribute?: attrID

$s? \notin Stt; Stt' = Stt \cup \{s?\}$
$value' = value \cup \{(s? \mapsto \{(attribute? \mapsto v?)\})\}$
$TagFunc' = TagFunc \cup \{(s? \mapsto New)\}$
$Eve' = Eve; Guardcon' = Guardcon$
$EffectT' = EffectT; Trans' = Trans$

تعریف ۲.۶ (عمل حذف از مدل دلتا). عمل حذفی در دلتا تبدیل مدلی است به شکل Del: IndependentTestModel → IndependentTestModel' که برای هر مدل آزمون دارای عناصر IndependentTestModel < Stt, Eve, Guardcon, EffectT, Trans>، مدل آزمون به‌روز شده IndependentTestModel' را چنان ایجاد می‌کند که هرکدام از عناصرش زیرمجموعه‌ای از مجموعه متناظر قبل از تغییرات است. این عمل یک الگوی گراف با تطبیق عنصر جدید در پیش‌شرایط را توصیف می‌کند. به‌منظور فراهم آوردن یک‌راه حل کلی در پرس‌وجوی اطلاعات در مدل به‌روز شده، یک برچسب "Delete" برای عناصر حذف شده توسط تابع TagRelation نسبت داده می‌شود.

DelStt

Δ*IndependentTestModel*
s?: State
type?: typeID
attribute?: attrID

$s? \in St; Stt' = Stt \setminus \{s?\}$
$FromSource' = FromSource \rhd \{s?\}$
$ToTarget' = ToTarget \rhd \{s?\}$
$value' = \{s?\} \lhd value; type' = \{s?\} \lhd type$
$TagFunc' = TagFunc \cup \{(s? \mapsto Delete)\}$
$Eve' = Eve; Guardcon' = Guardcon$
$EffectT' = EffectT; Trans' = Trans$

درنهایت هر عنصر به‌روز شده بر اساس ترکیبی از عملیات حذف و درج قابل‌تعریف است و با استفاده از برچسب "Update" مورد پرس‌وجو قرار می‌گیرد. مطابق با فرامدل آزمون تعریف شده در این مقاله، عملیات ریزدانه دلتا متفاوتی برای بازتولید مدل قابل‌تعریف است. با ترکیب شِماهای مختلف می‌توان الگوهای تغییر پیچیده‌ای تعریف و اعمال کرد. به‌عنوان نمونه دو عمل دلتا "Add"و "Del" را روی مجموعه حالات مدل آزمون IndependentTestModel در نظر بگیرید. برای افزودن یک حالت ورودی ?s به مجموعه حالات، شِما AddStt تعریف شده است. پیش‌شرایط این شِما الزام می‌کند که حالت جدید ?s نباید در مجموعه حالات کنونی مدل وجود داشته باشد و پس‌شرایط این عمل الزام می‌کند که بعد از اجرای این عمل، عنصر جدید ?s باید به مدل آزمون افزوده شده باشد. به‌طور مشابه پیش‌شرایط و پس‌شرایط توسط شِما عملیاتی DelStt برای حذف عنصری از فضای حالت مدل آزمون IndependentTestModel تعریف شده است که الزام می‌کند قبل از رخداد عمل، عنصر باید در فضای حالت وجود داشته باشد و پس از رخداد عمل، لازم است چنین عنصری از مدل آزمون حذف شده باشد. عمل تبدیل به‌روزرسانی می‌تواند مستقیماً توسط شِمایی مانند UpdateStt تعریف شود و یا از ترکیب دو عمل حذف و افزودن حاصل شود. ترکیب و تفکیک شِماها این امکان را فراهم می‌کند که رفتارهای پیچیده یک سیستم را بر اساس رفتارهای پایه آن‌ها تعریف کرد. بر این اساس اصول بازتولید و دستکاری یک مدل را بر طبق مجموعه عملیات پایه دلتا به‌صورت زیر می‌توان تعریف کرد:

$$ChangeState \triangleq AddStt \vee DelStt \vee UpdateStt$$
$$ChangeTransition \triangleq AddTransition \vee DelTransition \vee UpdateTransition \vee AddLabel \vee UpdateLabel \vee DelLabel$$

ابعاد مفهومی متفاوتی برای هر مدل تغییرات لازم است تعریف شود. این ابعاد با توجه به مفاهیم توسعه مدل-رانه عبارت‌اند از نحو انتزاعی دلتا، نحو جزئی مدل دلتا[7] مدل دلتا از نظر معنایی و نگاشت از سطح انتزاعی به سطح واقعی. نحو انتزاعی دلتا بیانگر دامنه مفهومی مدل دلتا می‌باشد که عناصر اصلی مدل، قیود بین عناصر مدل و محدودیت‌ها، روابط را نشان می‌دهد و معمولاً با فرامدل‌سازی همراه است. نحو واقعی مدل دلتا مدل تغییرات دنیای واقعی را به‌صورت متنی، گرافیکی یا ترکیبی از این دو نشان می‌دهد. نحو جزئی دلتا پس از تعیین محیط پیاده‌سازی و در سطح وابسته به سکوی خاص و با انتساب مقادیر واقعی به عناصر انتزاعی مدل دلتا، حاصل می‌شود. مدل دلتا از نظر معنایی با استفاده از توصیف‌های صوری مبتنی بر ریاضی برای بیان معانی دقیق ارکان یک مدل دلتا به کار می‌رود.

تعریف ۷ (نحو انتزاعی دلتا). از لحاظ نحوی هر مدل دلتا شامل مجموعه‌ای از سه‌تایی‌ها به شکل $(DeltaSIG \times dependency \times DeltaSIG)$ است که هر عنصر مدل دلتا، *DeltaSIG*، به‌صورت چندتایی زیر تعریف می‌شود:

$$DeltaSIG == TestTemplate \times DeltaOp \times Value \times Value$$

این عناصر به ترتیب عبارت‌اند از عنصری از مدل که دچار تغییر شده، نوع عمل تغییر مقدار قبل از تغییر و مقدار بعد از تغییر. انواع وابستگی بین عناصر یک مدل دلتا و عملیات تغییر DeltaOp به‌صورت زیر تعریف می‌شود:

DeltaOp ::= *AddMT* | *DelMT* | *UpdateMT*

dependency ::= *Direct* | *Indirect* | *Indep* | *Conflict*

بین هر دو عنصر در مدل دلتا وابستگی‌های متفاوتی ممکن است وجود داشته باشد که در ادامه توضیح داده می‌شود. این وابستگی‌ها برای اعمال صحیح تغییرات در سیستم و همچنین بهینه‌سازی مدل تغییرات استفاده می‌گردد. در واقع روابط بین عناصر مدل یک رابطه ترتیب جزئی است.

۳-۲- بهینه‌سازی مدل تغییرات

هر تغییر در سطح دانه‌بندی خود دارای یک عملکرد خاص است و در سطح دانه‌بندی بزرگ‌دانه می‌تواند دارای معانی متفاوت دیگری باشد. مثلاً تغییر Add در سطح دانه‌بندی خود دارای تأثیر افزایشی و در سطح بزرگ‌دانه (مثلاً رابطه کلاس و بسته) دارای تأثیر به‌روزرسانی است. به همین ترتیب تغییرات حذف و به‌روزرسانی در سطح دانه‌بندی بزرگ‌تر دارای تأثیر به‌روزرسانی هستند. انواع وابستگی ساختاری بین تغییرات موجود در مدل دلتا دارای روابط ذکر شده در جدول ۱ هستند. به‌عنوان نمونه دو تغییر افزودنی از نوع AddMT ممکن است با هم رابطه شمول[8] داشته باشند، روی یک عنصر عمل کرده و با هم برخورد داشته و یا هیچ وابستگی با یکدیگر نداشته باشند. در کل هر دو عمل تغییر ممکن است دارای روابط زیر باشند: (۱) از هم مستقل باشند که در این صورت با هر ترتیب قابل اجرا هستند، (۲) روی یک عنصر اعمال شوند که دارای رابطه تعریف/استفاده[9] هستند و با هم برخورد دارند که باید مدیریت شوند و (۳) رابطه شامل و مشمول با هم داشته باشند که در این صورت ابتدا باید عمل افزودن موجودیت اعمال شونده انجام گیرد. برای افزایش کارایی آزمون رگرسیون نیازمندی‌های آزمون باید مدل دلتا بهینه شده را پیمایش کنند.

بر اساس این توضیحات، قسمتی از مدل دلتا برای تغییراتی که در فاز طراحی بر رفتار خدمات وب شکل ۱ انجام شد، در جدول ۲ نشان داده شده است. به‌منظور بهینه‌سازی مدل تغییرات باید تغییرات روی T19 نادیده گرفته شده و تغییرات روی T6 برحسب نوع تغییر مدیریت شوند.

جدول ۱: بهینه‌سازی وابستگی‌های داخلی بین عناصر دلتا

Op1/Op2	AddMT	UpdateMT	DelMT
AddMT	Conflict for op2	Sequential(def/use)	None
UpdateMT	Conflict for op1	Sequential	DelMT
DelMT	AddMT	Conflict for op2	Conflict for op2

۴- معناشناسی آزمون مستقل از سکو

در این بخش به توصیف مفاهیم آزمون مستقل از سکو می‌پردازیم. روش ارائه شده برای آزمون رگرسیون مبتنی بر تغییرات، با تعریف معیارهای پوشش متفاوت، به دنبال برآوردن نیازمندی‌های آزمون، اجرای آزمون و بررسی نتایج حاصل بر اساس اوراکل مورد نظر است.

تعریف ۸ (الگوی اتمیک مورد آزمون). برای هر مدل رفتاری تعریف شده بر اساس فرامدل ماشین حالت دارای برچسب، الگوی اتمیک آزمون با پنج‌تایی زیر تعریف می‌شود.

AtomicTestCasePattern: $\mathbb{P}$ (*State* × *Event* ×*GuardCon* × *Effect* × *Transition*)

$\forall$*tc: AtomicTestCasePattern*
- $\exists$*m: IndependentTestModel*
  - *tc* . 1 = *m* . *FromSource tc* . 5
    $\wedge$ *tc* . 2 = *m* . *labelE tc* . 5
    $\wedge$ *tc* . 3 = *m* . *labelE tc* . 5
    $\wedge$ *tc* . 4 = *m* . *labelE tc* . 5

جدول ۲: قسمتی از مدل تغییرات مربوط به فاز تکامل شکل ۱

(T2: *Transition*, ***UpdateMT***, empty, NewVal)
(T6: *Transition*, ***UpdateMT***, empty, NewVal)
(T13: *Transition*, ***AddMT***, empty, NewVal)
(T14: *Transition*, ***AddMT***, empty, NewVal)
(T15: *Transition*, ***AddMT***, empty, NewVal)
(T16: *Transition*, ***AddMT***, empty, NewVal)
(T6: *Transition*, ***UpdateMT(event)***, OldValue, NewVal)
(T16: *Transition*, ***DelMT***, OldVal, empty)
(T8: *Transition*, ***UpdateMT***, OldValue, NewVal)
(S8: *State*, ***UpdateMT***, empty, NewVal)
(S9: *State*, ***AddMT***, empty, NewVal)

تعریف ۹ (الگوی آزمون انتزاعی). یک مورد آزمون انتزاعی دنباله‌ای از الگوی اتمیک آزمون است که یک مسیر را در یک مدل آزمون می‌سازد. هر الگوی آزمون انتزاعی بر اساس معیار پوشش تعیین شده، بیانگر یک نیازمندی آزمون است که باید روی مدل مورد بررسی قرارگرفته و برآورده شود.

ATC == seq *AtomicTestCasePattern*

بر اساس این تعریف، یک مورد آزمون سطح پیاده‌سازی دارای جزییاتی بیشتر نسبت به یک *ATC* است. زیرا اطلاعاتی مانند مقادیر متغیرها و ویژگی‌های لازم بر اساس دامنه ورودی معتبر به متغیرهای مسیر پیمایش شونده انتساب داده شده است.

تعریف ۱۰ (توصیف معنایی موارد آزمون). از نظر معناشناسی یک مورد آزمون بر اساس مجموعه ورودی‌ها و خروجی‌هایش با دوتایی <*Inputs*, *Outputs*> توصیف می‌شود[۱۲]. ورودی‌ها شامل پیش‌شرایط و گام‌ها (دنباله‌ای از عملیات که در مسیر آزمون اجرا می‌گردند) بوده و خروجی‌ها شامل مقادیر مورد انتظار و پس‌شرایط می‌باشند که وابسته به جزییات اجرای موارد آزمون هستند. پیش‌شرایط تضمین می‌کند که موارد آزمون در حالت کنونی سیستم قابل اجرا هستند و گام‌ها دنباله‌ای از عملیات اجرایی در مسیر هر مورد آزمون را نشان می‌دهند. خروجی مورد انتظار، نتایج حاصل از اجرای یک مورد آزمون؛ و پس‌شرایط مقادیر متغیرهای مورد آزمون پس از اجرا می‌باشد.

۵- دسته‌بندی موارد آزمون رگرسیون

روش آزمون رگرسیون مبتنی بر دلتا به‌عنوان یک تکنیک امن، همه بخش‌های تغییر یافته یک مدل آزمون را مورد آزمون مجدد قرار می‌دهد. به این منظور یک دسته‌بندی کلی برای موارد آزمون رگرسیون مبتنی بر دلتا عبارت است: موارد آزمونی که مدل دلتا را پیمایش می‌کنند و موارد آزمونی که مدل دلتا را پیمایش نمی‌کنند. دسته اول شامل موارد آزمونی که بخش‌های حذف شده مدل را پیمایش می‌کنند و موارد آزمونی که بخش‌های به‌روز شده مدل را پیمایش می‌نمایند. به دسته دوم موارد آزمون قابل اعمال مجدد نیز می‌گوییم. این دسته موارد آزمون برای نسخه جدید سیستم قابل اعمال هستند ولی اجرای مجدد آن‌ها ضرورت ندارد. همچنین برای حفظ سازگاری لازم است موارد آزمونی برای پیمایش بخش‌های جدید مدل نیز تعریف شوند. برای توصیف موارد آزمون رگرسیون بر اساس تعاریف پیش دو مجموعه totalTrace وAllTest به ترتیب با مجموعه همه عناصر مدل که توسط هر ATC پیمایش می‌شوند و مجموعه آزمون اولیه برای مدل آزمون تعریف می‌کنیم. بر این اساس موارد آزمون پیمایش کننده دلتا مواردی هستند که عمل دلتا آن‌ها عمل DelMT و/یا UpdateMT بوده و اشتراک عناصر آن‌ها با عناصر مدل دلتا غیر تهی است.

ChMeeted

Ξ*ChgModel*
Ξ*TestSuite*
traverse!: $\mathbb{P}$ *ATC*

$\exists$ *ChModel:* $\mathbb{P}$ *TestTemplate* | *ChModel* = { *e:* *TestTemplate* | $\exists$*s: TotalDeltaSig* • *s* . 1 = *e* $\wedge$ *s*. 2 $\in$ {*DelMT*, *UpdateMT*} } • *traverse!* = { *s: ATC* | *s* $\in$ *AllTest!* $\wedge$ *totalTrace! s* $\cap$ *ChModel* $\neq \emptyset$ }

NonChMeeted

Ξ*ChgModel*
Ξ*TestSuite*
nontraverse!: $\mathbb{P}$ *ATC*

$\exists$ *ChModel:* $\mathbb{P}$ *TestTemplate* | *ChModel* = { *e:* *TestTemplate* | $\exists$*s:TotalDeltaSig* • *s* . 1 = *e* } • *nontraverse!* = { *s:ATC* | *s* $\in$ *AllTest!* $\wedge$ *totalTrace! s* $\cap$ *ChModel* = $\emptyset$ }

تعریف ۱۱ (موارد آزمون قابل استفاده مجدد). موارد آزمون قابل استفاده مجدد، موارد آزمونی هستند که اشتراک عناصر آن‌ها با عناصر مدل دلتا تهی است.

ReusableATC

Ξ*ChgModel*

$\Xi TestSuite$
$reusable!: \mathbb{P}\ ATC$

$\exists\ ChModel: \mathbb{P}\ TestTemplate \mid ChModel = \{\ e: TestTemplate \mid \exists s: TotalDeltaSig \bullet s.1 = e\ \} \bullet reusable! = \{\ s: ATC \mid s \in AllTest! \wedge totalTrace!\ s \cap ChModel = \emptyset\ \}$

تعریف ۱۲ (موارد آزمون قابل اجرای مجدد). موارد آزمون قابل اجرای مجدد، موارد آزمون پیمایش کننده دلتا هستند که عمل دلتا آن‌ها عمل UpdateMT است.

UpdatableATC
$\Xi ChgModel$
$\Xi TestSuite$
$updatable!: \mathbb{P}\ ATC$

$updatable! \subseteq AllTest!$
$\exists\ ChModel: \mathbb{P}\ TestTemplate \mid ChModel = \{e: TestTemplate \mid \exists s: TotalDeltaSig \bullet s.1 = e \wedge s.2 = UpdateMT\ \} \bullet updatable! = \{\ s: ATC \mid s \in AllTest! \wedge totalTrace!\ s \cap ChModel \neq \emptyset\ \}$

تعریف ۱۳ (موارد آزمون منسوخ). موارد آزمون منسوخ، موارد آزمون پیمایش کننده دلتا هستند که عمل دلتا آن‌ها عمل DelMT است.

RemovableATC
$\Xi ChgModel$
$\Xi TestSuite$
$removable!: \mathbb{P}\ ATC$

$\exists\ ChModel: \mathbb{P}\ TestTemplate \mid ChModel = \{\ e: TestTemplate \mid \exists s: TotalDeltaSig \bullet s.1 = e \wedge s.2 = DelMT\ \} \bullet removable! = \{\ s: ATC \mid s \in AllTest! \wedge totalTrace!\ s \cap ChModel \neq \emptyset\ \}$

درنهایت لازم است برای عناصری که جدیداً به مدل افزوده شده‌اند موارد آزمون جدیدی تعریف شود.

NewATC
$\Xi ChgModel$
$\Xi TestSuite$
$newtest!: \mathbb{P}\ ATC$

$\exists\ ChModel: \mathbb{P}\ TestTemplate \mid ChModel = \{\ e: TestTemplate \mid \exists s: TotalDeltaSig \bullet s.1 = e \wedge s.2 = AddMT\ \} \bullet newtest! = \{\ s: ATC \mid s \notin AllTest! \wedge totalTrace!\ s \cap ChModel \neq \emptyset\ \}$

۵-۱- معیارهای پوشش مبتنی بر دلتا

برای هر معیار یا الگوی پوشش دو جنبه تجریدی و واقعی قابل‌تعریف است. جنبه تجریدی شامل اطلاعات تجریدی در مورد مسیرهایی است که باید پیمایش شوند و جنبه واقعی شامل جنبه تجریدی به‌علاوه اطلاعات لازم برای اجرای واقعی نیازمندی‌های آزمون، به‌عنوان نمونه اطلاعاتی راجع به مقادیر متغیرها و توابع. متغیرها و عناصر لازم برای هر معیار پوشش مبتنی بر دلتا شامل نوع عمل تغییر، عنصر تغییر یافته و وابستگی بین عناصر دلتا، داده‌های تجریدی محسوب می‌شوند. به‌منظور اِعمال قوانین پوشش دلتا روی نیازمندی‌های آزمون لازم است پیش‌شرایط هر قانون برآورده شود. پیش‌شرایط شامل الگویی برای بررسی انطباق هم‌زمان عنصر، نوع تغییر و نوع وابستگی است. در واقع ما شرایط هر معیار پوشش را به‌صورت یک الگوی پرس‌وجو تعریف می‌کنیم که لازم است برای برآورده شدن آن، انطباق بین الگوی پوشش و نیازمندی‌های آزمون وجود داشته باشد. در مورد معیارهای پوشش مبتنی بر دلتا، علاوه بر عناصر مورد هدف معیار آزمون، عمل تغییر و وابستگی بین تغییرات نیز اهمیت پیدا می‌کند. یعنی هر معیار پوشش مبتنی بر دلتا تابعی از مدل تغییرات و مدل آزمون به مجموعه گزاره‌های روی عناصر درگیر تغییرات است. هر گزاره ممکن است چندین عنصر هدف داشته باشد. بر این اساس با هر تغییر یک عنصر از مدل، مدل مقصد مربوط به این عنصر هم تغییر می‌کند. مطابق با معیارهای پوششی که در مدل‌های رفتاری گذار-حالت تعریف می‌شود، معیارهای پوشش متفاوتی برای مدل دلتا نیز می‌توان تعریف کرد که تکنیک‌های پوشش مبتنی بر دلتا نامیده شده‌اند. یک معیار پوشش مبتنی بر دلتا، یک مجموعه نیازمندی‌های آزمون برای اندازه‌گیری کیفیت آزمون رگرسیون را مشخص می‌کند. دو دسته معیار پوشش مبتنی بر دلتا برای آزمون رگرسیون تعریف می‌کنیم: معیارهای پوشش دلتا مبتنی بر استراتژی و معیارهای پوشش دلتا تک منظوره، که هر دو بر اساس نحو زیر قابل توصیف است:

$CovCriteria: IndependentTestModel \rightarrow \mathbb{P}\ TestRequirementPred$

$CCCommit: CovCriteria \twoheadrightarrow \mathbb{P} \mid ATC$

$\forall pred: TestRequirementPred \bullet \{pred\} \in \text{ran}\ CovCriteria \Leftrightarrow (\exists\ t: ATC \bullet \{t\} \in \text{ran} \mid CCCommit)$

$CCCategory ::= StrategyBasedCC \mid AdHocCC$

تعریف ۱۴ (معیارهای پوشش دلتا مبتنی بر استراتژی). این معیارها بر اساس معیارهای پوشش ساختاری فاز آزمون نرم‌افزار [۱۳] تعریف می‌شوند. این معیارها معمولاً برای نشان دادن ویژگی‌های نیازمندی‌های آزمون ضروری هستند و معمولاً با اجرای آن‌ها به موارد "همه"، "وجود دارد"، "هیچ‌کدام" و ... در نیازمندی‌های آزمون دست پیدا کنیم، به‌عنوان نمونه پوشش همه حالات یا رویدادهای تغییر یافته. در ادامه الگوهای پوشش متفاوت بیان شده‌اند. البته الگوهای پوشش به این موارد محدود نمی‌شوند بلکه برای همه عناصر مدل رفتاری قابل توصیف هستند ولی به علت تعداد زیاد و شباهت در توصیف، تنها موارد اصلی ذکر شده‌اند.

الگوی پوشش ۱. پوشش حالت-دلتا[۱۰] یک مورد آزمون رگرسیون[۱۱] این معیار پوشش را برآورده می‌کند به شرطی که هر حالت

(نود) غیر محذوف تغییر یافته در مدل دلتا در مجموعه آزمون ملاقات شود. کوچک‌ترین زیرمجموعه‌ای از نیازمندی‌های آزمون پوشا که این معیار پوشش را برآورده می‌کند مجموعه آزمون رگرسیون امن و کارا است. برای عناصر اضافه شده به مدل، نیازمندی‌های آزمون جدید باید به این مجموعه اضافه شود.

CoverageAllStates

ChgModel

$$\forall e: State \bullet \exists s: TotalDeltaSig \bullet s.1 = e \wedge s.2 \in \{AddMT, UpdateMT\} \Rightarrow \exists t: AtomicTestCasePattern \bullet t.1 = e \wedge \langle t \rangle \in CCSet$$

الگوی پوشش ۲. پوشش گذار-دلتا[۱۲]: یک آزمون رگرسیون این معیار پوشش را برآورده می‌کند به شرطی که هر گذار (لبه) غیر محذوف تغییر یافته در مدل دلتا در مجموعه آزمون ملاقات شود. برای لبه‌های اضافه شده به مدل، نیازمندی‌های آزمون جدید باید به این مجموعه افزوده شوند. گذارهای تغییر کرده در این معیار پوشش دارای طول یک یا صفر می‌باشند. با توجه به اینکه نودها به‌عنوان لبه‌های دارای طول صفر در نظر گرفته می‌شوند، این معیار پوشش می‌تواند جایگزینی برای معیار پوشش تغییرات حالت (الگوی پوشش ۱) باشد.

CoverageAllTransitions

ChgModel

$$\forall e: Transition \bullet \exists s: TotalDeltaSig \bullet s.1 = e \wedge s.2 \in \{AddMT, UpdateMT\} \Rightarrow \exists t: AtomicTestCasePattern \bullet t.1 = e \wedge \langle t \rangle \in CCSet$$

الگوی پوشش ۳. پوشش رویداد-دلتا: یک مورد آزمون رگرسیون این معیار پوشش را برآورده می‌کند به شرطی که هر رویداد تغییر یافته در مدل دلتا در مجموعه آزمون ملاقات شود.

الگوی پوشش ۴. پوشش n-گذار-دلتا[۱۳]: یک مورد آزمون رگرسیون این معیار پوشش را برآورده می‌کند به شرطی که نیازمندی‌های آزمونی وجود داشته باشد که n گذار غیر محذوف تغییر یافته را پیمایش کنند (اگر n=۰ باشد الگوی پوشش ۱ و اگر n=۱ باشد الگوی پوشش ۲ حاصل می‌شود).

۶- نتایج ارزیابی

به‌منظور ارزیابی روش پیشنهادی لازم است که رویکرد ارائه شده با برخی روش‌های موفق در آزمون رگرسیون مبتنی بر مدل مقایسه شود. اگر فرض شود هزینه‌ها یکسان و یا دارای تغییر نامحسوس باشد، آنگاه به‌درستی رویکرد به‌عنوان نخستین رویکردهای مدل-رانه در راستای خودکارسازی آزمون نرم‌افزار به‌ویژه آزمون رگرسیون، بسنده می‌کنیم.

به‌منظور پیاده‌سازی تبدیل مدل افزایشی برای آزمون رگرسیون مبتنی بر دلتا از VIATRA2 [۱۴] به‌عنوان موتور تبدیل افزایشی قدرتمندی که تحت Eclipse ارائه شده است استفاده کردیم. VIATRA2 یک چارچوب تبدیل مدل با قابلیت پردازش پرس‌وجو بر روی مدل‌ها، به‌عنوان مبنایی برای تطبیق الگو و انتشار تغییرات فراهم می‌کند. لیست ۱ علاوه بر تعریف فرامدل‌های ماشین حالت توسعه یافته و مدل نیازمندی‌های آزمون به‌عنوان مدل‌های مبدأ و مقصد، قسمتی از قوانین تبدیل مدل افزایشی به زبان VIATRA2 را توصیف می‌کند. رویکرد فرامدل‌سازی بصری و دقیق در چارچوب VIATRA2 از مدل‌سازی چندسطحی با روابط نمونه‌ایِ صریح و تعمیم‌یافته، الگوهای مختلف تبدیل مدل و تعریف جریان‌های کنترل پیچیده حمایت می‌کند .در این مقاله با ایجاد سه موردمطالعه تجاری به ارزیابی رویکرد ارائه شده پرداختیم. در این سناریوها، آزمون مبتنی بر مدل و معماری مدل-رانه را با استفاده از قابلیت‌هایی مثل مدل‌های دلتا، قابلیت ردیابی و انتشار تغییر برای ایجاد آزمون رگرسیون مبتنی بر مدل ادغام شده است. از نظر کیفی، این پژوهش در مقایسه با روش‌های مختلف آزمون رگرسیون فاکتورهای کاربردی ازجمله: (۱) دقت، (۲) ایمنی، (۳) کارایی، (۴) عمومیت یا استقلال، (۵) آزمون حاصل از پرس‌وجو، (۶) آزمون رویشی، (۷) یافتن محل خطا مبتنی بر ردیابی، (۸) اندازه‌گیری قابلیت پوشش آزمون، (۹) امکان پالایش خودکار و (۱۰) مدل‌سازی خطا ارائه می‌دهد. سایر دلایل مقبولیت چارچوب آزمون رگرسیون مدل-رانه در این مقاله عبارت‌اند از: کاهش هزینه آزمون رگرسیون، انتخاب زیرمجموعه‌ای مناسب برای آزمون رگرسیون با هزینه کمتر و یا، حداکثر مشابه با روش‌های دیگر در ادبیات موضوع، به‌کارگیری قابلیت‌های خودکار توسعه مدل-رانه در فاز آزمون، ایجاد سطوح متفاوتی از انتزاع در تولید آزمون مبتنی بر مدل، یکپارچه‌سازی آزمون رگرسیون در فرایند عمومی تولید آزمون، انتخاب موارد آزمون مناسب برای فاز نگهداری بر اساس موارد آزمونی که تحت تأثیر تغییرات قرار گرفته‌اند، بهبود کارایی در تشخیص خطای تولید محصول نرم‌افزاری.

جدول شماره ۳ اطلاعات کاملی از تغییرات اعمال شده بر روی موارد مطالعه شامل CS1 تا CS3 را نشان می‌دهد. این جدول، برای هر تغییر ایجاد شده در موارد مطالعه، تعداد، عنصر هدف و نوع تغییر اعمال شده را نشان می‌دهد. ابتدا تغییراتی را با ایجاد مدل دلتا بهینه شده به مدل رفتاری ایجاد شده، اعمال کردیم و سپس برای انتخاب یک مجموعه آزمون برای اجرای آزمون رگرسیون از چارچوب آزمون رگرسیون مبتنی بر دلتا استفاده کردیم. در مطالعات ما، تعداد موارد آزمون بعد از تغییرات با استفاده از تکنیک‌های: آزمون مجدد-همه[۱۴]، تکنیک انتخاب بر اساس مدل دلتا (ChbRTST[۱۵]) و رویکرد بهینه شده انتخاب بر اساس مدل دلتا، اجرا شد. روش بهینه‌شده رویکردی است که موارد آزمون تکراری پیمایش کننده دلتا را از مجموعه موارد آزمون حذف می‌کند. این روش از تکنیک‌های الویت‌دهی مانند رخداد تعداد تغییرات بیشتر و یا طول بیشتر مورد آزمون برای حذف موارد آزمون تکراری استفاده کند. جدول ۴ نتایج حاصل از این مقایسه را نشان می‌دهد.

جدول ۳: عملیات دلتا روی موارد مطالعه

	AddS	AddT	DelS	DelT	UpdateS	UpdateT
CS 1	۴	۲	۱	۲	۰	۴
CS 2	۶	۸	۰	۱	۳	۸
CS 3	۱۱	۹	۴	۵	۴	۸

نتایج شکل ۲ نشان می‌دهد که استفاده از روش انتخاب موارد آزمون مبتنی بر دلتای بهینه‌سازی شده، کاهش قابل استدلالی در موارد آزمون قابل اجرای مجدد در آزمون رگرسیون خواهد داشت. محور افقی موارد مطالعه و محور افقی تعداد موارد آزمون رگرسیون برای روش‌های مختلف را نشان می‌دهد.

۷- ارزیابی مقایسه‌ای با تحقیقات مشابه

به‌طور معمول دو روش برای ارزیابی رگرسیون مبتنی بر مشخصات، وجود دارد: روش‌های غیر صوری ازجمله روش‌هایی که مبتنی بر UML هستند و روش‌های صوری. اگرچه استراتژی‌های یکپارچه ناشی از ترکیب مزایای UML و روش‌های صوری، توجه فزاینده‌ای را به خود جلب کرده‌اند، اما هنوز جای خالی دیدگاه‌های یکپارچه در آزمون رگرسیون وجود دارد. در یک دسته‌بندی کلی کارهای انجام شده با استفاده از روش‌های صوری و روش‌های غیر صوری در زمینه آزمون رگرسیون مبتنی بر مدل را به دو دسته می‌توان تقسیم کرد که هرکدام چارچوب تحلیلی منحصر به خود را دارند.

```
namespace SM2TF;
import statemachine.metamodel;
import TestCaseMM.metamodel;
@incremental
machine TriggerAllState
{ gtrule testTrigger(in SM, inout State) =
  { rule main() =      call transform(); rule transform() = seq {
            let TestFramework = undef in
             forall SM with apply transformSM(SM, TestFramework)
do  try seq {          call checkTransformation(SM);
println("*** UML SM " + name(SM) + " has been transformed into
Test Req "      + fqn(TestFramework));          }
println("*** Cound not transform UML SM: " + name(SM)); }
          gtrule transformSM(out SM, out TestFramework) = {
            precondition pattern unmappedSM(SM)   ={
           'StateMachine'(SM); neg  find   stateMapping(SM,
NoTestFramework);  }
            postcondition find stateMapping(SM, TestFramework)
            action {   move(SM, statemachine.models);
                       call copyName(SM, TestFramework);
                       call TransFunc(SM, TestFramework); }      }
rule TransFunc(in SM, in TestFramework) = seq {
forall X in SM with apply additionalState(SM, State,TestFramework)
do println("AddPred to satisfy CC:All State " + fqn(State));
forall X in SM with apply additionalTransition(SM,Transition,
TestFramework) do println("AddPred to satisfy CC:All Transition " +
fqn(Transition));
forall X in SM with apply UpdatedState(SM, State,TestFramework)
do println("Retest Test Cases that trace the state " + fqn(State));
forall X in SM with apply UpdatedTransitions(SM,Transition,
TestFramework)
do println("Retest Test Cases that travese the transition " +
fqn(Transition));
forall X in SM with apply RemovalState(SM,State, TestFramework)
do println("Delete Test cases that travese the state" + fqn(State));
forall X in SM with apply RemovalTransformation(SM,Transition,
TestFramework) do println("Delete Test cases that travese the
transition " + fqn(Transition));          } } }
```

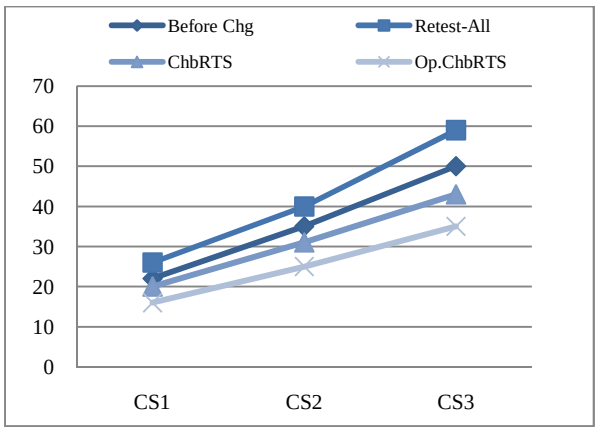
لیست ۱: تبدیل مدل افزایشی مدل رفتاری به مدل آزمون

۷-۱ روش‌های غیر صوری

UML به‌عنوان یک زبان مدل‌سازی به‌عنوان راهی استاندارد برای تجسم بخشیدن و بصری‌سازی طراحی یک سیستم مطرح شده است. تکنیک‌های مختلفی برای آزمون رگرسیون مبتنی بر مدل‌های UML در تحقیقاتی مانند [۱۵]، [۱۶]، [۱۷] [۱۸]، [۱۹] و [۲۰] ارائه شده است. همچنین در حوزه‌های تحقیقاتی مختلفی در زمینه آزمون رگرسیون مانند تکنیک‌های اولویت‌بندی و کمینه‌سازی آزمون رگرسیون مبتنی بر UML پژوهش‌هایی ازجمله [۲۱]، [۲۲]، [۲۳]، [۲۴] و [۲۵]، [۲۶]، [۲۷] و [۲۸] گزارش شده است. یک بررسی پیرامون تکنیک‌های مختلف انتخاب و اولویت‌بندی در آزمون رگرسیون توسط نویسندگان [۲۹] انجام شده و نیز یک بازبینی سیستماتیک جامع نیز بر اساس طبقه‌بندی روش‌ها به روش‌های مبتنی بر کد و مبتنی بر مدل توسط [۱۶] ارائه شده است. نویسندگان در [۱۸] یک RTST[۱۶] بر اساس تجزیه و تحلیل نمودارهای دنباله و کلاس UML پیشنهاد دادند.

جدول ۴: تحلیل نتایج با استفاده از تکنیک‌های انتخابی

	قبل از تغییرات	بعد از تغییرات			روش تست مجدد همه	روش پیشنهادی	روش پیشنهادی بر اساس دلتا بهینه شده
	# ATCs	#RemATCs	#UpATCs	#NewATCs	#ATCs	#ATCs	#ATCs
CS1	۲۲	۵	۱۲	۹	۲۶	۱۹	۱۶
CS2	۳۵	۱۲	۱۶	۱۲	۴۰	۳۱	۲۵
CS3	۵۱	۱۳	۲۰	۱۶	۵۹	۴۳	۳۵



شکل ۲: مقایسه روش‌های RTST و ChbRTST

روش آن‌ها مبتنی بر این فرض بود که بین مدل‌های طراحی، کد و موارد آزمون قابلیت ردیابی کامل وجود دارد. در [۱۷] یک روش RTS بر اساس تغییرات معین در هر دو نمودار حالت و کلاس UML که برای آزمون رگرسیون مبتنی بر مدل استفاده شده است، پیشنهاد شده است. چن و دیگران [۱۹] هم یک تکنیک RTS مبتنی بر مشخصات ارائه دادند که بر اساس نمودارهای فعالیت UML برای مدل‌سازی رفتار سیستم و نیازهایی که بالقوه تأثیر پذیرند، ارائه دادند. آن‌ها همچنین موارد آزمون رگرسیون و موارد آزمون امن را بر اساس تجزیه و تحلیل تغییرات، تقسیم‌بندی کردند. وو و اوفوت [۲۰] یک تکنیک مبتنی بر UML ارائه دادند که در این تکنیک سعی بر آن است که به حل مشکلات گزارش شده برای پیاده‌سازی سیستم‌های نرم‌افزاری مبتنی بر مؤلفه پرداخته شود. کورل و دیگران [۲۳] چند فن‌آوری هوشمند برای آزمون مبتنی بر مدل را ارائه کردند. آن‌ها همچنین با استفاده از یک مطالعه آزمایشی، فن‌آوری‌های هوشمند اولویت‌بندی[۳۰] را در آزمون مبتنی بر مدل مقایسه کردند تا تأثیر آن‌ها را در کشف خطای اولیه نشان دهند. همتی و دیگران [۲۱] الگوریتم‌های مختلف مبتنی بر تنوع را که بر اساس تشابه بین مسیرهایی که موارد آزمون در مدل حالت عمل می‌کنند، بررسی کردند. تهت و دیگران [۲۲] دو روش انتخابی مبتنی بر مدل و نیز یک روش اولویت‌بندی آزمون مبتنی بر وابستگی با استفاده از مدل حالت سیستم تحت کنترل را ارائه کرده و ارزیابی کردند. پیلس کالنز و دیگران [۳۱] روش مدل‌سازی دیگری را ارائه کردند که در آن به‌جای تست کردن کدهای سطح پیاده‌سازی، مدل‌های فاز طراحی را برای آزمون رگرسیون تست می‌کنند. برخی از مطالعات نیز از زبان‌های مدل‌سازی دامنه‌های خاص برای استخراج آزمون رگرسیون استفاده نموده‌اند؛ مثلاً یوآن و دیگران [۳۲]. در [۳۳] یک دیدگاه مبتنی بر قانون برای آزمون رگرسیون بر اساس نمودارهای فعالیت، کلاس، دنباله و مورد کاربرد پیشنهاد شده است. در [۳۴] و [۳] با استفاده از نمودارهای UML1.4 و تمرکز بر پالایش و بررسی سازگاری بین مدل‌ها برای تحلیل تأثیر تغییرات پرداخته شده است. اما فقط تعداد بسیار محدودی از روش‌های مبتنی بر UML، جنبه‌های MDA را پوشش می‌دهند.

۷-۲- روش‌های صوری

روش‌های صوری یک رویکرد تکمیلی بدون ابهام برای اعتبار بخشیدن و تائید کردن مصنوعات نرم‌افزاری در سطح مناسبی از مفهوم و انتزاع، فراهم می‌کنند. آزمون نرم‌افزار بر اساس مشخصات صوری می‌تواند با کشف زودهنگام خطاهای توصیف و طراحی، کیفیت نرم‌افزار را افزایش دهد. روش‌های مختلفی برای تولید مورد آزمون با استفاده از روش‌های صوری مخصوصاً Z، انجام شده است مانند روش‌های ارائه شده در [۱۳] و [۳۵]؛ اما کارهای مرتبط بسیار کمی در زمینه آزمون رگرسیون وجود دارد که توصیف صوری را حمایت می‌کند. چن و دیگران [۳۶] با استفاده از توصیف شی‌گرا زبان Z روشی برای آزمون رگرسیون ارائه کردند. لیانگ[۳۷] بر اساس مشخصات صوری TCOZ، روشی را برای این آزمون پیشنهاد کرد. توصیف‌های TCOZ از ادغام Object-Z و CSP[۱۷] زمان‌دار استفاده می‌کند.

بر اساس تحقیقات نویسندگان این مقاله، تا کنون در ادبیات مربوطه روشی برای ادغام MDA و نمادگذاری صوری برای آزمون رگرسیون پژوهش‌های متعددی وجود ندارد. پژوهش پیشین نویسندگان این مقاله [۳۸]، رویکردی مبتنی بر تغییرات برای آزمون رگرسیون پیشنهاد داده‌اند که قابلیت تولید خودکار موارد آزمون با استفاده قوانین تبدیل مدل و همچنین تبدیل خودکار روبه‌عقب برای یافتن محل خطا در سطح انتزاعی را فراهم نکرده است. در این پژوهش علاوه بر خودکارسازی انتشار تغییرات، با استفاده از تبدیل مدل دوطرفه امکان خطایابی خودکار بعد از تغییر محصول فراهم شده است. رویکرد پیشنهادی جدید، با استفاده از روش پالایش خودکار به تحلیل چند موردمطالعه دنیای واقعی پرداخته است. همچنین این پژوهش به‌منظور بهبود عملکرد آزمون رگرسیون مبتنی بر مدل، با استفاده از نسخه توسعه‌یافته VIATRA2 ، امکان استفاده از پرس‌وجوی مدل-رانه بر مدل دلتا، لینک‌های ردیابی و مدل مقصد را عملی کرده است.

۸- نتیجه‌گیری و راهکارهای آینده

هدف از انتشار به‌روزرسانی، محاسبه تغییرات ضمنی مدل در حین دستکاری صریح عناصر مدل و به دست آوردن مدل مقصد سازگار با مدل مبدأ و مدل تغییرات است. ازآنجاکه در اکثر موارد به‌روزرسانی تنها بخش کوچکی از مدل مبدأ را تحت تأثیر قرار می‌دهد، محاسبه تغییرات ناشی از آن با مقایسه حالت جدید و قدیم مدل، فرایندی منطقی به نظر نمی‌رسد. در این پژوهش از انتشار افزایشی تغییرات برای خودکارسازی انتخاب مجموعه سازگار موارد آزمون رگرسیون استفاده شده است. تعیین کوچک‌ترین سطح دانه‌بندی « بدون برخورد» در به‌روزرسانی، برای حمایت از طیف وسیعی از برنامه‌های کاربردی در معرض تغییر، از اهداف این مقاله بوده است. به‌کارگیری این رویکرد در فاز آزمون منجر به انتخاب زیرمجموعه‌ای از موارد آزمون که بر سیستم تکامل‌یافته قابل اعمال بوده و خطاهای موجود در سیستم را به‌صورت زودهنگام کشف می‌کند، شده است. زیرمجموعه منتخب آزمون رگرسیون نسبت به روش‌های معمول دارای کاهش قابل قبولی بوده است. برخی از اهداف آتی این پژوهش عبارت‌اند از: ارائه رویکردی برای خودکارسازی پالایش چارچوب با استفاده از حساب پالایش، استفاده از ابزار خودکار مناسبی برای خلق موارد آزمون جدید بر اساس نیازمندی‌های جدید خلق شده و ارائه پرس‌وجوهای پیچیده‌تر و یکپارچه‌سازی آن با ابزارهای آزمون جهت توسعه مفهومی با نام « آزمون رگرسیون پرس‌وجوگرا».

پیوست الف

برخی نشانه‌گذاری‌های اصلی زبان Z در ادامه توصیف شده‌اند.

نماد	تعریف
first e	first(e1,e2) = e1
second e	second(e1,e2) = e2

$\mathbb{P}\ e$	$\{i : \mathbb{W} \mid i \subseteq e\}$
$e1 \lhd e2$	$\{i : e2 \mid first\ i \in e1\}$
$e1 \lhd\!\!\!\!- e2$	$\{i : e2 \mid first\ i \notin e1\}$
$e1 \rightarrow\!\!\!\!\!\!+\ e2$	$\{i1 : P\ (e1 \times e2) \mid \forall\ i2, i3 : i1 \mid first\ i2 = first\ i3 \bullet second\ i2 = second\ i3\}$
$e1 \rightarrow e2$	$\{i : e1 \rightarrow\!\!\!\!\!\!+\ e2 \mid dom\ i = e1\}$
ΔS	Shortcut for $S \wedge S'$
ΞS	Shortcut for $S \wedge S' \wedge \theta S = \theta S'$

مراجع

[۱] زینب اسمعیل‌پور، اشکان سامی، «گسترش ابزارهای خودکار شناسایی الگوهای طراحی با عملگر تصحیح برچسب»، مجله مهندسی برق دانشگاه تبریز، دوره ۴۵، شماره ۲، صفحه ۱۱-۲۶، ۲۰۱۴.

[۲] لیلا صمیمی دهکردی، بهمن زمانی و شکوفه کلاهدوز رحیمی، «ارائه روشی جدید برای تبدیل مدل دوسویه بر اساس چارچوب اپسیلون و تکنیک‌های ردیابی‌پذیری»، مجله مهندسی برق دانشگاه تبریز، دوره ۴۷، شماره ۳، صفحه ۱۱۲۱-۱۱۳۲، ۲۰۱۷.

[3] L. C. Briand, Y. Labiche, and T. Yue, "Automated Traceability Analysis for UML Model Refinements," Information and Software Technology, vol. 51, no. 2, pp. 512-527, 2009.

[4] H. Zhu, P .A. Hall, and H. R. May, "Software unit test coverage and adequacy," ACM Computing Surveys, vol. 29, pp. 336–427, 1997.

[5] J. M. Spivey, *The Z Notation: A Reference Manual*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[6] M. Saaltink, "The Z/EVES System," in Proceedings of the 10th International Conference of Z Users on The Z formal Specification Notation, Lecture Notes in Computer Science 1212, pp. 72-85, 1997.

[7] R. Heckel, J. M. Kuster, and G. Taentzer, "Confluence of typed attributed graph transformation systems. In: Graph Transformation," in First International Conference. Lecture Notes in Computer Science 2505, pp. 161–176, 2002.

[8] P. Stevens, "Bidirectional model transformations in QVT: semantic issues and open questions," Software & Systems Modeling, vol. 9, p. 7, 2010.

[9] A. Agrawal, G. Karsai, S. Neema, F. Shi, and A. Vizhanyo, "The Design of a Language for Model Transformations," Journal of Software and System Modeling, vol. 5, pp. 261-288 2005.

[10] J. Buckley, T. Mens, M. Zenger, A. Rashid, and G. Kniesel, "Towards a taxonomy of software change," Software Maintenance and Evolution, vol. 17, pp. 309-332, 2005.

[11] G. Bergmann, I. Ráth, G. Varró, and D. Varró, "Change-driven model transformations," Software & Systems Modeling, vol. 11, pp. 431-461, 2012.

[12] G. Myers, *The Art of Software Testing*, John Wiley & Sons, Inc., 2004.

[13] P. Ammann and J. Offutt, "Using formal methods to derive test frames in category-partition testing," in Proceeding of the 9th Annual Conference on Computer Assurance, pp. 69-80, 1994.

[14] D. Varró and A. Balogh, "The Model Transformation Language of the VIATRA2 Framework," Science of Computer Programming, vol. 68, pp. 214–234, 2007.

[15] G. Rothermel and M. J. Harrold, "Analysing Regression Test Selection Techniques," IEEE Transactions on Software Engineering, vol. 22, pp. 529-551, 1996.

[16] E. Engström, M. Skoglund, and P. Runeson, "Empirical evaluations of regression test selection techniques: a systematic review. Proceedings of the Empirical software engineering and measurement," in ACM-IEEE international symposium, pp. 22–31, 2008.

[17] Q. Farooq, "A Model Driven Approach to Test Evolving Business Process based Systems," in 13th International Conference on Model Driven Engineering Languages and Systems, pp. 6-24, 2010.

[18] L. C. Briand, Y. Labiche, and S. He, "Automating regression test selection based on UML designs," Information & Software Technology, vol. 51, pp. 16-30, 2009.

[19] Y. Chen, R. Probert, and D. Sims, "Specification-based regression test selection with risk analysis," in Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research, p. 1, 2002.

[20] Y. Wu and J. Offutt, "Maintaining Evolving Component based software with UML," in Proceeding of 7th European conference on software maintenance and reengineering, pp. 133-142, 2003.

[21] H. Hemmati, A. Arcuri, and L. Briand, "Achieving scalable model-based testing through test case diversity," ACM Trans. Softw. Eng. Methodol, vol. 22, p. 6 pages, 2013.

[22] L. Tahat, B. Korel, H. M, and H. Ural, "Regression test suite prioritization using system models," Softw. Test., Verif. Reliab, vol. 22, pp. 481-506, 2012.

[23] L. B. Kore, G. Koutsogiannakis, and L. Tahat, "Model-based test prioritization heuristic methods and their evaluation," in the 3rd International Workshop on Advances in Model-Based Testing, pp. 34–43, 2007.

[24] M. Khatibsyarbini, M. A. Isa, D. N. Jawawi, and R. Tumeng, "Test case prioritization approaches in regression testing: A systematic literature review," Information and Software Technology, vol. 93, pp. 74-93, 2017.

[25] J. Chen, L. Zhu, T. Y. Chen, D. Towey, F.-C. Kuo, R. Huang, et al., "Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering," Journal of Systems and Software, vol. 135, pp. 107-125, 2018.

[26] A. Ansari, A. Khan, A. Khan, and K. Mukadam, "Optimized regression test using test case prioritization," Procedia Computer Science, vol. 79, pp. 152-160, 2016.

[27] P. Sapna and A. Balakrishnan, "An approach for generating minimal test cases for regression testing," Procedia computer science, vol. 47, pp. 188-196, 2015.

[28] A. Schwartz and H. Do, "Cost-effective regression testing through Adaptive Test Prioritization strategies," Journal of Systems and Software, vol. 115, pp. 61-81, 2016.

[29] G. Rothermel and M. Harrold, "A safe, efficient regression test selection technique," ACM Trans. On Softw. Eng. and Methodology, vol. 6, pp. 173-210, 1997.

[30] K. Beck, *Test Driven Development: By Example*, Addison-Wesley Professional, 2002.

[31] O. Pilskalns, G. Uyan, and A. Andrews, “Regression Testing UML Designs,” in ICSM. 22nd IEEE International Conference, pp. 254-264, 2006.

[32] Q. Yuan, J. Wu, C. Liu, and L. Zhang, “A model driven approach toward business process test case generation,” in 10th International Symposium on Web Site Evolution, pp. 41-44, 2008.

[33] D. Deng, P. C. Sheu, and T. Wang, “Model-based testing and maintenance,” in Proceedings of IEEE Sixth International Symposium on Multimedia Software Engineering, pp. 278- 285, 2004.

[34] N. Mansour and H. Takkoush, “UML based regression testing technique for OO software,” in Proceedings of the IASTED International Conference on Software Engineering and Applications, Boston, pp. 96-101, 2007.

[35] P. A. Stocks and D. Carrington, “Test templates: a specification-based testing framework,” in Proceeding of 15th Int. Conf. on Software Engineering, pp. 405-414, 1996.

[36] C.-Y. Chen, R. Chapman, and K. H. Chang, “Test scenario and regression test suite generation from Object-Z formal specification for object-oriented program testing,” in Proceedings of the 37th annual Southeast regional conference (CD-ROM), p. 37, 1999.

[37] H. Liang, “Regression Testing of Classes Based on TCOZ Specifications,” in Proceedings of the 10th International Conference on Engineering of Complex Computer Systems, pp. 450-457, 2005.

[38] M. Nooraei Abadeh and S. H. Mirian-Hosseinabadi, “Delta-based regression testing: a formal framework towards model-driven regression testing,” Journal of Software: Evolution and Process, vol. 27, pp. 913-952, 2015.

زیرنویس‌ها

[1] Adequacy Criteria
[2] Coverage
[3] Safety-Critical
[4] Typed Attributed Graph
[5] Labeled State Machine
[6] X-Directional Multi Level
[7] Direct Model Transformation
[8] Container/contained
[9] Definition/use
[10] Delta-State (Node) Coverage
[11] Regression Test Case
[12] Delta-Transition (Edge) Coverage
[13] Delta-N-Edge Coverage
[14] Retest-All
[15] Change-based Regression Test Selection Technique
[16] Regression Test Selection Technique
[17] Constraint Satisfaction Problem