ارائه روشی جدید برای تبدیل مدل دوسویه بر اساس چارچوب اپسیلون و تکنیک‌های ردیابی‌پذیری

لیلا صمیمی دهکردی[۱]، دانشجوی دکتری؛ بهمن زمانی[۲]، استادیار؛ شکوفه کلاهدوز رحیمی[۳]، استادیار

۱- گروه پژوهشی مهندسی نرم‌افزار مدل‌رانده - دانشکده مهندسی کامپیوتر - دانشگاه اصفهان - اصفهان - ایران - samimi@eng.ui.ac.ir
۲- گروه پژوهشی مهندسی نرم‌افزار مدل‌رانده - دانشکده مهندسی کامپیوتر - دانشگاه اصفهان - اصفهان - ایران - zamani@eng.ui.ac.ir
۳- گروه پژوهشی مهندسی نرم‌افزار مدل‌رانده - دانشکده مهندسی کامپیوتر - دانشگاه اصفهان - اصفهان - ایران - shrahimi@eng.ui.ac.ir

چکیده: توسعه مدل رانده روش نوینی در تولید نرم‌افزار است که در آن، از روی مدل و با به‌کارگیری مجموعه روش‌هایی تحت عنوان تبدیل مدل، کد پیاده‌سازی به‌صورت خودکار/ نیمه‌خودکار تولید می‌شود. کلیه فعالیت‌های ممکن در روش‌های مدل رانده با استفاده از تبدیل‌ها انجام می‌گیرد. یکی از این فعالیت‌ها، تبدیل مدل‌به‌مدل است که در ساده‌ترین حالت، ارتباطی تک‌سویه را بین مدل مبدأ و مقصد تعریف می‌کند. در این حالت، فقط می‌توان مدل مقصد را از روی مدل مبدأ به دست آورد. اما در حالت کلی‌تر، مدل‌های مبدأ و مقصد مستقلاً تغییر می‌یابند. درنتیجه، برای سازگاری به تبدیل‌های پیشرفته‌تری به‌نام تبدیل دوسویه نیاز است. روش‌های تبدیل دوسویه دارای چالش‌هایی مانند ابهام در زبان تبدیل، نمادسازی خاص و صوری‌سازی گران هستند. جهت رفع این چالش‌ها، در این مقاله، روش جدیدی برای تبدیل دوسویه برمبنای چارچوب مدل رانده اپسیلون و تکنیک‌های ردیابی‌پذیری پیشنهاد می‌شود. در این روش، روابط بین مدل‌های مبدأ و مقصد صوری‌سازی شده، سپس روی محکی شناخته‌شده پیاده‌سازی می‌شود. این روش بر اساس ۱۰ معیار با سه روش تبدیل دوسویه مقایسه می‌گردد. نتیجه مقایسه‌ها برتری‌های روش پیشنهادی از قبیل قابلیت حمل، نگه‌داری، مصالحه و پشتیبانی عملگری را نشان می‌دهد.

واژه‌های کلیدی: توسعه مدل رانده، تبدیل مدل دوسویه، ردیابی‌پذیری، چارچوب اپسیلون.

A New Approach for Bidirectional Model Transformation Based on the Epsilon Framework and Traceability Techniques

L. Samimi-Dehkordi[1], PhD Student; B. Zamani[2], Assistant Professor; S. Kolahdouz-Rahimi[3], Assistant Professor

1, 2, 3- MDSE Research Group, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran,
Emails: 1- samimi@eng.ui.ac.ir, 2- zamani@eng.ui.ac.ir, 3- shrahimi@eng.ui.ac.ir

Abstract: Model-driven development is a new paradigm in software development in which the implementation code is automatically\semi-automatically generated from the model, using a set of techniques named model transformations. All activities in model-driven approaches are performed via the application of transformations. One of such activities is model-to-model transformation, which defines an unidirectional relation between the source and target models in the simplest case studies. In this case, the only possibility is to produce the target model from the source model. However, in general, the source and target models are evolved independently. Therefore, a more advanced transformation called bidirectional transformation (Bx) is required to restore consistency. Bx approaches have several challanges such as ambiguities in transformation language, special notation, and expensive formalization. In order to cope with such challenges, in this paper, a new Bx approach is proposed based on the Epsilon model-driven framework and traceability techniques. In this approach, the relation between the source and target models is formalized, and then is implemented on a well-known benchmark. The proposed Bx approach is compared with three bidirectional approaches based on ten criteria. The result of comparison demonstrates the superiority of the proposed approach in terms of portability, preservation, reconciliation, and operation support.

Keywords: Model-driven development, bidirectional transformation (Bx), traceability, Epsilon framework.

تاریخ ارسال مقاله: ۱۳۹۵/۰۲/۲۲
تاریخ اصلاح مقاله: ۱۳۹۵/۰۴/۱۵
تاریخ پذیرش مقاله: ۱۳۹۵/۰۶/۰۴
نام نویسنده مسئول: بهمن زمانی
نشانی نویسنده مسئول: ایران - اصفهان - خیابان هزارجریب - دانشگاه اصفهان - دانشکده مهندسی کامپیوتر.

۱- مقدمه

امروزه روش‌های توسعه نرم‌افزار به دنبال ارتقای سطح تجرید در جهت کاهش پیچیدگی‌های برنامه‌ها هستند [۱، ۲]. یکی از جدیدترین روش‌هایی که از ایده ارتقای سطح تجرید به‌منظور خودکارسازی تولید کد استفاده می‌کند، توسعه مدل رانده[۱] است. این روش توسعه از روی مدل و با به‌کارگیری مجموعه روش‌هایی تحت عنوان تبدیل مدل، کد پیاده‌سازی را به‌صورت نیمه‌خودکار تولید می‌کند. با توجه به آنکه برنامه را می‌توان ترکیبی از ساختمان داده‌ها و الگوریتم در نظر گرفت، نرم‌افزار را نیز می‌توان ترکیبی از مدل‌ها و تبدیل‌ها دانست [۳]. با در نظر گرفتن کد و مدل به عنوان مصنوعات قابل تولید، چهار نوع تبدیل به‌صورت مدل‌به‌مدل، مدل‌به‌کد، کدبه‌مدل و کدبه‌کد بررسی می‌شود [۴]. تمرکز این مقاله روی تبدیل مدل‌به‌مدل است.

در ساده‌ترین حالت، تبدیل، برای ارتباطی تک‌سویه بین مدل مبدأ و مقصد به کار گرفته می‌شود. در حالت کلی‌تر، مدل‌های مبدأ و مقصد مستقلاً تغییر می‌یابند [۵]. در این حالت، چارچوب مدل رانده باید با استفاده از تبدیل مدل دوسویه غنی شود، تا بتواند سازگاری میان مدل مبدأ و مقصد را حفظ کند.

یک تبدیل مدل دوسویه دو وظیفه دارد [۶]: ۱) بررسی سازگاری: باید بتواند تعیین کند که چه زمانی دو مدل بر طبق تعریف تبدیل سازگار هستند؛ ۲) تحمیل (یا بازیابی) سازگاری: باید بتواند یک جفت مدل ناسازگار را گرفته و یکی از مدل‌ها (ازقبل‌تعیین‌شده) را تغییر دهد تا سازگاری برقرار شود.

روش‌های متعددی برای تعریف و اجرای تبدیل دوسویه در سال‌های اخیر ارائه شده است. اکثر این روش‌ها از نظر سبک زبان اعلانی هستند و همین باعث می‌شود که اجرای تبدیل در آن‌ها با ابهاماتی همراه باشد [۷]. همچنین زبان‌های اعلانی دارای نشانه‌گذاری خاص و پیچیده‌ای هستند که توسعه‌دهنده مجبور است، آن را فراگیرد [۵]. ازطرف‌دیگر، زبان‌های دستوری یا روش‌های تک‌سویه نیز نمی‌توانند اکثر نیازمندی‌های لازم دوسویگی را فراهم کنند.

در [۸]، ما روشی را بر مبنای زبان اعتبارسنجی اپسیلون[۲] و تکنیک‌های ردیابی‌پذیری[۳] به‌نام EVL+trace مطرح کردیم که به حل مشکلات روش‌های دوسویه همانند ابهامات ذاتی زبان‌های دوسویه پرداخته و ویژگی‌های لازم دوسویگی را سهولت می‌بخشد. در [۸]، فقط روابط یک به یک بین عناصر مبدأ و مقصد بررسی شد.

در این مقاله، روابط یک به چند و چند به یک صوری‌سازی می‌شود. سپس، نحوه کار با روش EVL+trace روی این روابط با توجه به محکی شناخته‌شده[۴] به‌نام Families2Persons موردمطالعه قرار می‌گیرد. سپس برای نمایش برتری‌های روش پیشنهادی، آن را از نقطه نظر ۱۰ معیار با سه روش شناخته‌شده تبدیل دوسویه مقایسه خواهیم کرد.

در ادامه مقاله، ابتدا مفاهیم اولیه مدل رانده از قبیل مهندسی مدل رانده، محصولات اولیه مدل رانده، تبدیل مدل و انواع آن، محک Families2Persons و چارچوب اپسیلون در بخش ۲ معرفی می‌شود. روش پیشنهادی EVL+trace، صوری‌سازی روابط بین مدل‌ها، چگونگی پیاده‌سازی روش پیشنهادی روی محک موردنظر و نتایج اجرا در بخش ۳ ارائه خواهد شد. بخش ۴، روش پیشنهادی را با سه روش شناخته شده تبدیل دوسویه بر طبق ۱۰ معیار مقایسه کرده و نتایج مقایسه را با رسم نمودار نشان می‌دهد. درنهایت، مقاله با نتیجه‌گیری در بخش ۵ به پایان می‌رسد.

۲- مفاهیم اولیه مدل رانده

به‌طور خلاصه، هر جهش در توسعه نرم افزار، با ارتقای سطح تجرید و درنتیجه، مستقل شدن از سکو منجر به نرم‌افزارهایی با قابلیت حمل بالا، افزایش کیفیت، بهره‌وری و قابلیت تعامل و درعین‌حال، کاهش زمان عرضه محصول نرم‌افزاری به بازار می‌شود [۳].

در دهه ۸۰، فناوری شیءگرا با اصل «هر چیز یک شیء است» بیان می‌شد، اما در مهندسی مدل رانده اصل مهم با عبارت «هر چیز یک مدل است» بیان می‌شود [۹]. می‌توان **مدل** را به‌عنوان انتزاعی از یک سیستم واقعی که بر طبق هدف خاصی به وجود آمده، تعریف کرد. به همان صورتی که برنامه‌های یک زبان برنامه‌نویسی بر اساس گرامر آن زبان نوشته می‌شوند، مدل نیز باید منطبق با زبان مدل که در اصطلاح به آن فرامدل[۵] می‌گوییم، باشد.

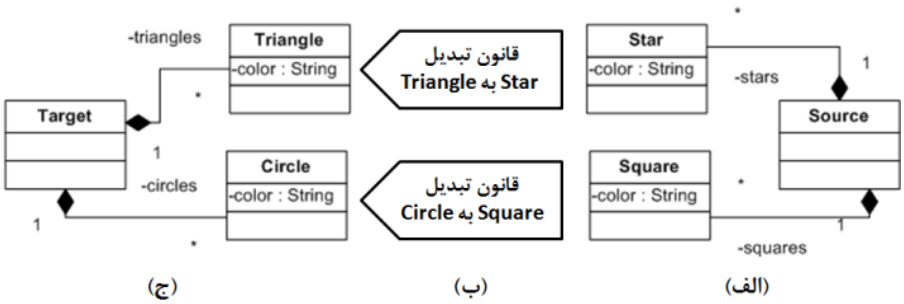
زبان مدل‌سازی یکپارچه می‌تواند به‌عنوان ایده‌ای برای تعریف فرامدل استفاده شود، البته نمی‌تواند به تنهایی در همه زمینه‌ها مفید باشد. بنابراین، برای جلوگیری از تنوع فرامدل‌های ناسازگار نیاز به زبانی احساس می‌شد که با آن بتوان همه فرامدل‌ها را تعریف کرد، که در اصطلاح به آن **فرا-فرامدل**[۶] گفته می‌شود [۳]. انجمن مدیریت شیء برای زبان فرا-فرامدل، استاندارد تسهیلات فرا شیء را معرفی کرد.

۲-۱- تبدیل مدل

طبق دیدگاه مدل رانده، مدل‌ها و فرامدل‌ها، شهروندان درجه یک هر فرآیند توسعه محسوب می‌شوند و سازوکاری که مدل‌های متفاوت را به هم متصل می‌کند، با عنوان تبدیل مدل شناخته می‌شود [۱۰]. زبان تبدیل، زبانی خاص قلمرو است که سازوکارهایی را برای تعریف، ترکیب و اجرای تبدیل فراهم می‌کند [۱۱]. با دنبال کردن اصل «هر چیزی یک مدل است»، تبدیل مدل نیز می‌تواند همانند یک مدل دیده شود. همان‌طورکه مدل‌های مبدأ و مقصد با فرامدل متناظرشان تطبیق دارند، قوانین تبدیل نیز با زبان تبدیل مطابقت دارند و همه فرامدل‌ها، یعنی فرامدل مبدأ، فرامدل مقصد و زبان (فرامدل) تبدیل، با فرا-فرامدل تسهیلات فرا شیء تطبیق دارند.

۲-۲- تبدیل مدل دوسویه

زبان‌های تبدیل از نظر جهت اجرا به دو دسته زبان تک سویه و زبان دوسویه تقسیم می‌شوند. **تبدیل دوسویه**[۷] سازوکاری برای نگه‌داری سازگاری میان دو (یا بیش‌تر) منبع اطلاعاتی مرتبط است [۱۲].



شکل ۱: الف) فرامدل مبدأ، ب) قوانین تبدیل و ج) فرامدل مقصد

پذیرش مفهوم دوسویگی در مهندسی مدل رانده در سال ۲۰۰۵ توسط OMG مطرح شد [۱۳]. زبان‌های تبدیل دوسویه روش رسمی برای تبدیل است که در آن هر برنامه، تبدیل روبه‌جلو و تبدیل روبه‌عقب را همزمان توصیف می‌کند.

در حال حاضر دو روش برای تبدیل مدل دوسویه وجود دارد [۶]: ۱) با نوشتن دو برنامه به زبان تبدیل تک‌سویه برای انجام تبدیل‌های روبه‌جلو و معکوس، ۲) با نوشتن برنامه‌ای به زبان تبدیل مدل دوسویه.

اهمیت دوسویگی در عمل به واسطه سناریوهای کاربردی آشکار شده است؛ ازجمله کاربردهای دوسویگی می‌توان به انتشار تغییر، همگام‌سازی، مدل‌سازی چندگانه ، تکامل نرم‌افزار و مهندسی رفت و برگشت اشاره کرد [۱۲].

۲-۳- مثالی ساده از تبدیل مدل

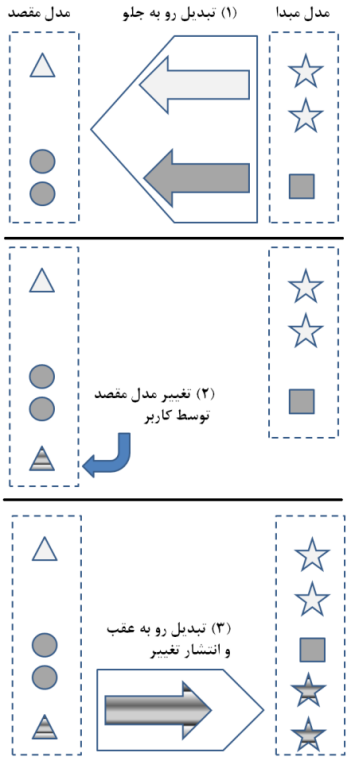
فرض کنید، بخواهیم مدلی از فرامدل مبدأ در شکل ۱ را به مدلی از فرامدل مقصد تبدیل کنیم. فرامدل مبدأ شامل فراکلاس‌های ستاره و مربع است و فرامدل مقصد شامل فراکلاس‌های مثلث و دایره است. بر طبق قوانین تبدیل به‌ازای ستاره‌ها در مدل مبدأ، مثلث همرنگ آن و به‌ازای مربع‌ها، دایره همرنگ در مقصد ایجاد می‌شود. همچنین باید توجه کرد که رابطه بین ستاره و مثلث رابطه چندبه‌یک و رابطه بین مربع و دایره یک‌به‌چند است.

نمونه‌ای از مدل مبدأ و مدل مقصد متناظر با آن در شکل ۲ نشان داده شده است. در مرحله (۱)، مدل مقصد با اجرای تبدیل روبه‌جلو براساس مدل مبدأ ایجاد می‌شود.

سپس، در مرحله (۲) ممکن است کاربر مدل مقصد را تغییر داده و یک شیء مثلث با رنگ متفاوت از مثلث قبلی ایجاد کند. در نتیجه، سازگاری بین دو مدل از بین می‌رود. برای بازیابی سازگاری بین مدل‌ها باید تبدیل را در جهت معکوس از مقصد به مبدأ (تبدیل روبه‌عقب) انجام دهیم تا تغییر ایجادشده در مدل مقصد به مبدأ انتشار یابد که در مرحله (۳) از شکل ۲ قابل مشاهده است.

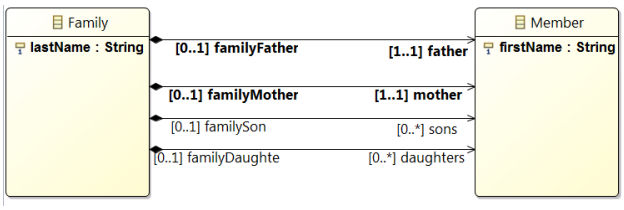
۲-۴- محک Families2Persons

این مثال برای ارزیابی و نمایش پیوندهای یک‌به‌چند (چندبه‌یک) در این مقاله استفاده می‌شود که در آن فرامدل‌های مبدأ (به‌نام Families) و مقصد (به‌نام Persons) بسیار ساده طراحی شده‌اند. فرامدل Families دارای فراکلاس Family است که با فراکلاس Member با چهار نوع رابطه در ارتباط است. هر شیء از نوع Family دارای صفت lastName از نوع رشته‌ای است؛ این صفت، شناسه محسوب می‌شود. هر شیء از نوع Member دارای صفت firstName است. هر خانواده دارای یک پدر، یک مادر، تعدادی فرزند پسر و فرزند دختر است. شکل ۳، فرامدل مبدأ را نمایش می‌دهد.

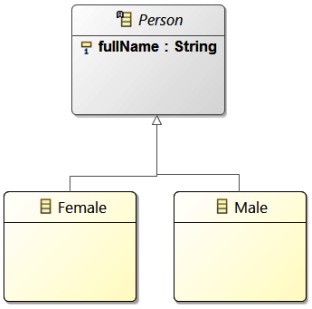


شکل ۲: مثالی ساده از تبدیل دوسویه

فرامدل Persons شامل فراکلاس انتزاعی Person می‌شود که هر فرد از این نوع دارای صفت fullName است. در این فرامدل، صفت fullName یک صفت کلیدی و منحصربه‌فرد برای هر شیء از نوع Person است. دو فراکلاس Female و Male از این فراکلاس به ارث می‌برند. در شکل ۴ تصویری از فرامدل مقصد نمایش داده شده است.



شکل ۳: فرامدل Families



شکل ۴: فرامدل Person

با توجه به هر دو فرامدل، به ازای هر عنصر از نوع Member، عنصری از نوع Person ایجاد می‌شود. در این حالت ترکیب صفت firstName مربوط به عنصر Member همراه با صفت lastName عنصر Family آن صفت fullName در Person مقداردهی می‌شود. درصورتی‌که، یک Member با رابطه father یا sons به Family مرتبط باشد، فرد متناظر در مدل مقصد از نوع Male خواهد شد و در غیر این صورت، از نوع Female می‌شود.

توجه کنید که اگر مقدار صفت firstName عضوی در مدل مبدأ تغییر کند، مقدار صفت fullName فرد متناظر با آن عضو باید تغییر یابد و بالعکس. همچنین، اگر مقدار صفت lastName مجموعه‌ای از اعضای خانواده‌ای تغییر یابد، مقدار صفت fullName همه افراد متناظر در مدل مقصد تغییر می‌کند. در صورتی که، بخش نام‌خانوادگی در صفت fullName مربوط به فردی در مدل مقصد تغییر کند، باید lastName خانواده متناظر با آن فرد و fullName همه افراد (در مدل مقصد) متناظر با آن خانواده نیز تغییر کنند.

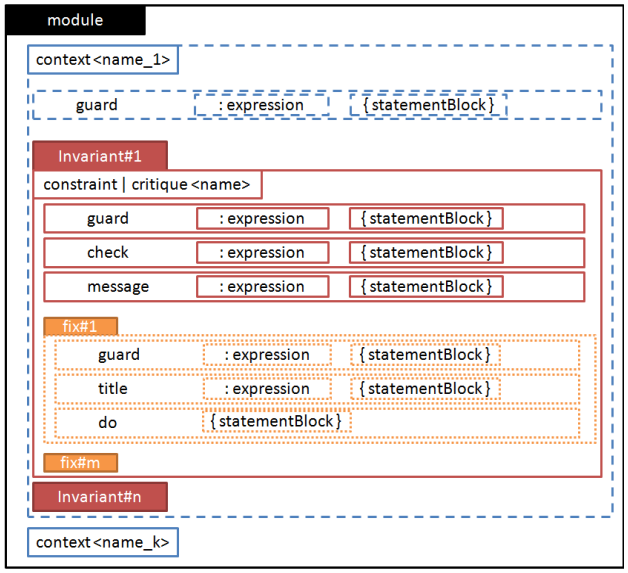
۲-۵- چارچوب اپسیلون

چارچوب اپسیلون زبان‌های متعددی را برای کار بر روی مدل‌ها فراهم می‌کند که یکی از این زبان‌ها برای ارزیابی مدل در نظر گرفته شده است. زبان اعتبارسنجی اپسیلون برای بیان قیود روی مدل‌هایی مبتنی بر فرامدل‌های مختلف به کار می‌رود که ارزیابی درون‌مدلی و بین‌مدلی را سهولت می‌بخشد [۱۴].

۲-۵-۱- زبان اعتبارسنجی اپسیلون

هدف از طراحی زبان اعتبارسنجی اپسیلون رفع محدودیت‌های روش‌های اعتبارسنجی دیگر همانند زبان قید شیء[۸] می‌باشد. ازجمله مزایای این زبان می‌توان به بازخورد بهتر به کاربر، شانس تعریف اخطار علاوه بر تعریف خطا، محدود کردن مجموعه نمونه‌ها با استفاده از نگهبان ، اصلاح ناسازگاری‌ها با روش‌های متعدد، پشتیبانی از مدل سازی بینابینی اشاره کرد. دو ویژگی آخر ما را قادر می‌سازد تا از این زبان برای تعریف تبدیل‌های دوسویه بهره ببریم. نحو عینی این زبان در شکل ۵ قابل مشاهده است.

این زبان به‌صورت مجموعه‌ای از پیمانه‌ها بیان می‌شود. هر پیمانه شامل مجموعه‌ای از قیود می‌باشد که در زمینه‌ها دسته‌بندی می‌شوند. در هر زمینه، یک نگهبان به‌طور اختیاری تعریف می‌شود که نمونه‌های موردبررسی توسط قیودِ آن زمینه را محدود می‌کند. در این حالت، اگر شرط آمده در نگهبان ارضا نشود، هیچکدام از قیود تعریف شده در آن زمینه موردبررسی قرار نمی‌گیرند.



شکل ۵: نحو عینی زبان اعتبارسنجی اپسیلون

هر قید می‌تواند در دو حالت خطا یا اخطار تعریف شود. ضرورت رفع اخطار از خطا کمتر است. در هر قید می‌توان به طور اختیاری نگهبان تعریف کرد تا مجموعه نمونه‌های موردبررسی را کوچک‌تر کرد. هر قید دارای دو قسمت بررسی و اصلاح است. در قسمت بررسی، شرطی تعریف می‌شود. در صورت عدم ارضای شرط، پیامی به کاربر نمایش داده می‌شود که متن پیام از قبل در برنامه آمده است و در آن دلیل رخداد خطا یا اخطار ذکر می‌شود.

قسمت اصلاح می‌تواند شامل چند راهکار باشد که هرکدام دارای دو بخش است: بخش عنوان و بخش اجرایی. در بخش عنوان پیامی نوشته می‌شود تا به کاربر بگوید قرار است چه کاری برای رفع خطا انجام گیرد. در بخش اجرایی دستوراتی که به رفع خطا منجر می‌شوند، آورده می‌شود.

۲-۵-۲- بحث و بررسی

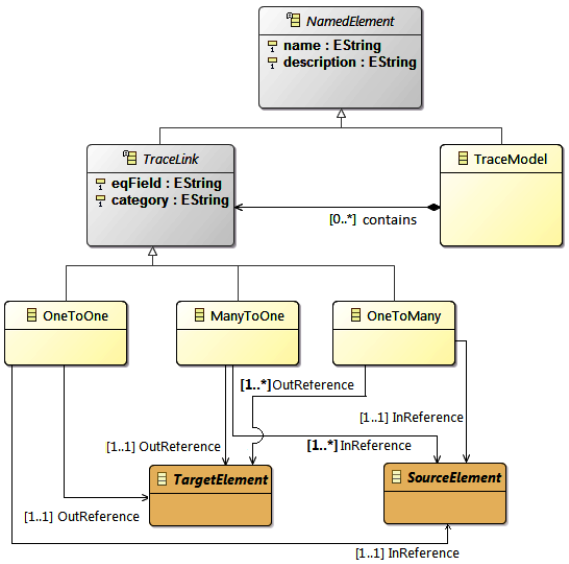
ایده استفاده از زبان اعتبارسنجی اپسیلون برای تبدیل‌های دوسویه در ابتدا در [۱۵] مطرح شد، اما این ایده در عمل چالش های متعددی دارد. یکی از این چالش‌ها این است که سازگاری تبدیل روبه‌جلو و روبه‌عقب باید با روش‌های وارسی گران‌قیمت بررسی شود. برای تعیین تبدیل دوسویه با استفاده از EVL باید برای هر عنصر در مدل مبدأ (مقصد) یک یا چند عنصر متناظر در مدل مقصد/ مبدأ وجود داشته باشد؛ درصورتی‌که این تناظر وجود نداشته باشد یک یا چند راه‌حل برای بازیابی ناسازگاری پیش‌آمده در بخش اصلاح مربوط به قید موردبررسی پیشنهاد می‌شود. اما با استفاده از زبان EVL نمی‌توان تغییرات را به درستی انتشار داد. با توجه به مثال تبدیل بخش ۲-۳ ، اگر در مدل مقصد، رنگ مثلث اول به قرمز تغییر یابد، برنامه EVL آن را به‌عنوان عنصری جدید در نظر می‌گیرد. درحالی‌که، عناصر ستاره متناظر با این عنصر مثلث در مدل مبدأ از قبل وجود داشته و

کافی‌ست فقط رنگ آن‌ها تغییر یابد. در بخش ۳، برای رفع این مشکل، روش پیشنهادی این مقاله را ارائه می‌دهیم.

۳- تبدیل دوسویه با استفاده از EVL+trace

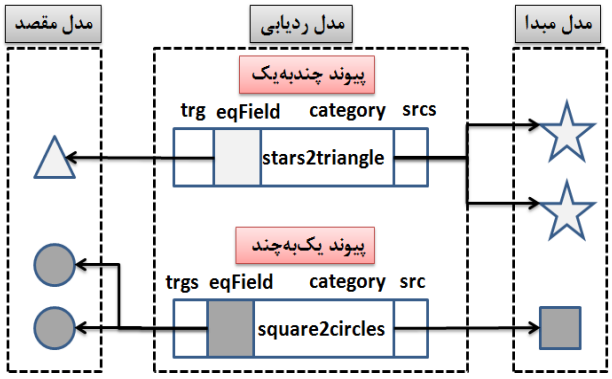
در بخش ۲-۵-۲ توضیح داده شد که استفاده از زبان اعتبارسنجی اپسیلون به تنهایی نمی‌تواند تغییر در مقدار صفات را انتشار دهد، بلکه فقط می‌تواند عناصر را حذف یا اضافه کند [۱۵]. به‌عنوان راه‌حل، ما در [۸] پیشنهاد کردیم که ارجاعاتی به عناصر متناظر مبدأ و مقصد در مدل سومی به‌نام مدل ردیابی ذخیره شود تا انتشار تغییر امکان‌پذیر باشد.

بر اساس اصول مدل رانده هر مدل مبتنی بر فرامدلی است؛ بنابراین، برای ایجاد مدل ردیابی نیازمند طراحی یک فرامدل ردیابی هستیم. این فرامدل، در شکل ۶ نمایش داده شده است. فرامدل ردیابی دارای ریشه‌ای به نام TraceModel است. این ریشه شامل تعدادی پیوند به‌نام TraceLink می‌باشد. بر اساس کاردینالیتی عناصر مبدأ و مقصد، هر پیوند می‌تواند از نوع یک‌به‌یک ، یک‌به‌چند و چندبه‌یک باشد. پیوند یک‌به‌یک برای ارجاع به یک عنصر مبدأ به یک عنصر مقصد، پیوند چندبه‌یک برای ارجاع به چند عنصر مبدأ به یک عنصر مقصد و پیوند یک‌به‌چند برای اتصال یک عنصر مبدأ به چند عنصر مقصد به کار می‌رود.



شکل ۶: فرامدل ردیابی روش پیشنهادی مربوط به EVL+trace [۸]

هر پیوند برای کوچک‌ترین بخش های مشترک (هم‌ارز) بین عناصر متناظر از مدل مبدأ و مقصد تعریف می‌شود. مقدار هم ارزی به‌صورت رشته‌ای در صفت eqField یا همان میدان هم‌ارزی ذخیره می‌شود. همچنین صفت category در TraceLink نمایانگر نوع (یا رسته) قانون تبدیل است. برای مثال برای تبدیل آمده در بخش ۲-۳ می‌توان مدل ردیابی را به صورت شکل ۷ ترسیم کرد. در شکل ۷، eqField همان رنگ مشترک بین عناصر مبدأ و مقصد است و category بر اساس اینکه چه نوع عنصری از مبدأ به چه عنصری از مقصد تبدیل می‌شود، تعیین می‌شود.



شکل ۷: مدل ردیابی برای تبدیل مثال بخش ۲-۳

با استفاده از فرامدل پیشنهادی می‌توان عملیات «حذف»، «اضافه» و «انتشار تغییر» را امکان‌پذیر کرد. در [۸]، فقط روابط یک به یک بررسی شد. در این مقاله، قصد داریم به پیوندهای یک‌به‌چند و چندبه‌یک و صوری‌سازی این روابط بپردازیم. برای چگونگی پیاده‌سازی روش روی این روابط از محک Families2Persons استفاده می‌کنیم.

۳-۱- صوری‌سازی روابط یک‌به‌چند و چندبه‌یک

در جدول ۱، نماد s یک عنصر و S مجموعه‌ای از عناصر همنوع از مدل مبدأ، r پیوند ردیابی از نوع یک‌به‌چند (یا چندبه‌یک) و نماد t یک عنصر و T مجموعه‌ای از عناصر همنوع از مدل مقصد است. با توجه به شرایط بیان‌شده، ۱۳ حالت با معنی برای رابطه یک‌به‌چند (حالات ۱ تا ۱۳) و ۱۳ حالت معنی‌دار برای رابطه چندبه‌یک (حالات ۱۴ تا ۲۶) وجود دارد. حالات بی‌معنی در جدول ۱ آورده نشده‌اند.

ابتدا، حالات ۱ تا ۱۳ را شرح می‌دهیم. حالت ۱، حالتی را بیان می‌کند که کاربر هیچ تغییری در عناصر مدل‌های مبدأ و مقصد نداده است؛ بنابراین، روش EVL+trace در مواجهه با آن هیچ عملی انجام نمی‌دهد.

طبق شرایط بیان‌شده در حالات ۲ تا ۵، تغییری در مقدار صفات عناصر مشاهده می‌شود و درنتیجه، انتظار داریم که EVL+trace، تغییر آشکار شده را انتشار دهد. حالت ۲ بیان می‌کند که عنصر مبدأ با عنصر ردیابی مساوی بوده، اما یکی (یا بیش‌تر) از عناصر مجموعه T با عنصر ردیابی مساوی نیست و درواقع، تغییر یافته است. بنابراین، برای بازیابی سازگاری باید عنصر مبدأ، ردیابی و همه عناصر T بر اساس مقدار t تغییر داده شوند. حالت ۳ نشان می‌دهد که عنصر مبدأ توسط کاربر تغییر یافته و EVL+trace باید این تغییر را به عنصر ردیابی و همه عناصر متناظر مقصد انتشار دهد. حالت ۴ بیان می‌کند که علاوه بر تغییر عنصر مبدأ، یک یا بیش‌تر از عناصر متناظر در مدل مقصد هم توسط کاربر تغییر یافته‌اند؛ درحالی‌که، این تغییرات سازگار و هم‌ارز هم هستند. دراین‌حالت، کافی است تغییر آشکارشده به عنصر ردیابی و بقیه عناصر مجموعه T انتشار یابد. در حالت ۵، تغییر در هر دو مدل مبدأ و مقصد تشخیص داده می‌شود، اما برخلاف حالت ۴ این تغییر سازگار نیست ($s \neq t$). بنابراین، EVL+trace پیغامی را به کاربر نشان می‌دهد تا او تصمیم بگیرد که کدام تغییر را ترجیح می‌دهد.

جدول ۱: صوری‌سازی روابط یک‌به‌چند و چندبه‌یک

حالت	مبدأ	ردیابی	مقصد	شرط
۱	s	r	T	$\forall t \in T: s = r = t$
۲	s	r	T	$\exists t \in T: s = r \wedge r \neq t$
۳	s	r	T	$\forall t \in T: s \neq r \wedge r = t$
۴	s	r	T	$\exists t \in T: s \neq r \wedge r \neq t \wedge s = t$
۵	s	r	T	$\exists t \in T: s \neq r \wedge r \neq t \wedge s = t$
۶	-	r	T	$\forall t \in T: r = t$
۷	-	r	T	$\exists t \in T: r \neq t$
۸	s	r	$\varnothing$	$s = r$
۹	s	r	$\varnothing$	$s \neq r$
۱۰	-	r	$\varnothing$	-
۱۱	s	-	$\varnothing$	-
۱۲	-	-	T	-
۱۳	s	-	T	$\forall t \in T: t = s$
۱۴	S	r	t	$\forall s \in S: s = r = t$
۱۵	S	r	t	$\forall s \in S: s = r \wedge r \neq t$
۱۶	S	r	t	$\exists s \in S: s \neq r \wedge r = t$
۱۷	S	r	t	$\exists s \in S: s \neq r \wedge r \neq t \wedge s = t$
۱۸	S	r	t	$\exists s \in S: s \neq r \wedge r \neq t \wedge s \neq t$
۱۹	$\varnothing$	r	t	$r = t$
۲۰	$\varnothing$	r	t	$r \neq t$
۲۱	S	r	-	$\forall s \in S: s = r$
۲۲	S	r	-	$\exists s \in S: s \neq r$
۲۳	$\varnothing$	r	-	-
۲۴	S	-	-	-
۲۵	$\varnothing$	-	t	-
۲۶	S	-	t	$\forall s \in S: s = t$

در حالات ۶ تا ۱۰، EVL+trace طبق شرط بیان‌شده، تشخیص می‌دهد که عنصری از مدل مبدأ یا همه عناصر T از مدل مقصد حذف شده‌اند. درنتیجه، انتظار می‌رود که تبدیل دوسویه برای بازیابی سازگاری عناصر متناظر باقی‌مانده در مدل‌های دیگر را حذف کند. در حالات ۶ و ۷، عنصر مبدأ حذف شده است؛ درحالی‌که در حالت ۶، عنصر ردیابی با عناصر مجموعه T از مقصد هم‌ارز است، اما در حالت ۷، این هم‌ارزی وجود ندارد. به عبارت دیگر، کاربر علاوه بر حذف عنصر مبدأ، یک (یا بیش‌تر) عنصر از مقصد را نیز تغییر داده است. چون در روش EVL+trace، عمل حذف بر تغییر ارجح است، این روش همه عناصر متناظر را حذف می‌کند. در حالات ۸ و ۹، عناصر مجموعه T توسط کاربر حذف شده‌اند و علاوه بر آن، حالت ۹ بیان می‌کند که عنصر s از مبدأ نیز تغییر یافته است. در حالت ۱۰، هم عنصر مبدأ و هم مجموعه T از مقصد حذف شده‌اند؛ درنتیجه، برای بازیابی سازگاری کافی‌ست عنصر r هم از مدل ردیابی حذف شود.

حالات ۱۱ تا ۱۳، نشان می‌دهند که کاربر عنصر جدیدی را در مدل مبدأ یا مقصد اضافه کرده است. این تشخیص به این دلیل است که برای عنصر جدید، هیچ پیوند متناظری در مدل ردیابی وجود ندارد؛ بنابراین، EVL+trace می‌فهمد که از قبل عنصری در مبدأ یا مقصد نبوده که متناظر آن در مدل ردیابی ایجاد شده باشد. در حالت ۱۱، کاربر عنصری جدید در مدل مبدأ اضافه کرده که متناظری در مدل ردیابی و مقصد برای آن وجود ندارد و باید توسط تبدیل دوسویه ایجاد شود. در حالت ۱۲، کاربر یک (یا بیش‌تر) عنصر همنوع و یکسان در مدل مقصد اضافه کرده که برای بازیابی سازگاری لازم است متناظر آن‌ها در مدل مبدأ و ردیابی ایجاد شود. در حالت ۱۳، کاربر عناصر متناظر و یکسانی را در مدل‌های مبدأ و مقصد ایجاد کرده اما در مدل ردیابی ثبت نشده‌اند. در این حالت، EVL+trace فقط یک پیوند ردیابی بین عناصر متناظر ایجاد می‌کند.

حالات ۱۴ تا ۲۶، پیوندهای چندبه‌یک را صوری می‌کنند. حالت ۱۴ همانند حالت ۱ است، با این تفاوت که در آن با مجموعه‌ای از عناصر مبدأ و عنصری از مدل مقصد سروکار داریم. حالات ۱۵ تا ۱۸، همانند حالات ۲ تا ۵، تغییر صورت‌گرفته توسط کاربر را تشخیص می‌دهند و برای بازیابی سازگاری، باید این تغییر انتشار یابد. حالات ۱۹ تا ۲۳ مانند حالات ۶ تا ۱۰، عمل حذف را تشخیص داده و در حالات ۲۴ تا ۲۶، کاربر عنصری را در مبدأ یا مقصد اضافه کرده است.

۳-۲- پیاده‌سازی EVL+trace روی محک Families2Persons

مدل ردیابی برای محک Families2Persons دارای دو نوع پیوند ردیابی است: ۱) پیوند ردیابی یک‌به‌چند با رسته FamilyPerson که میدان هم‌ارزی آن همان مقدار lastName مربوط به شیء Family می‌باشد. ۲) پیوند ردیابی یک‌به‌یک با رسته MemberPerson که میدان هم‌ارزی آن همان مقدار firstName مربوط به شیء Member می‌باشد. نمونه‌ای از مدل‌های مبدأ، ردیابی و مقصد در شکل ۸ نمایش داده شده است.

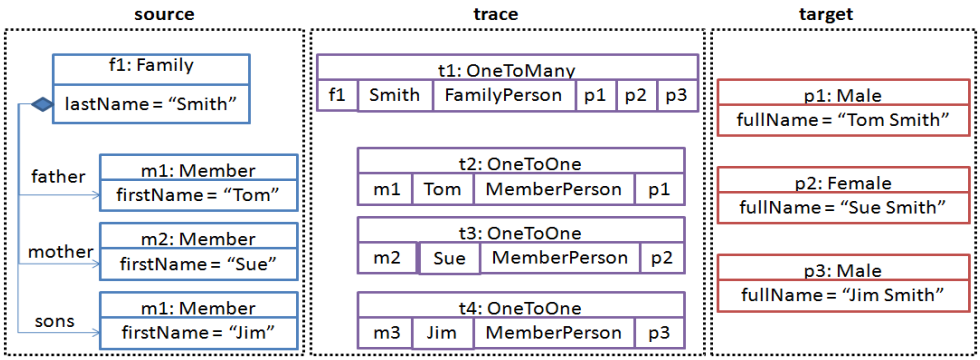
برای تعریف شناسه‌ای برای هر شیء در هر یک از مدل‌ها باید ویژگی xmi:id را برای هر عنصر فعال کنیم. این فعال سازی با دستوری ساده در اپسیلون بدون هیچ اثر جانبی در فرامدل‌ها امکان‌پذیر است:

model's name.resource.useXmiIds =true

در ادامه، سه عمل حذف، تغییر و اضافه کردن روی مدل‌های محک Families2Persons مورد بررسی قرار می‌گیرد.

۳-۲-۱- حذف

درصورتی‌که، یک شیء از نوع Family توسط کاربر حذف شود، یعنی حالت ۶ یا ۷ از جدول ۱ اتفاق بیفتد، آن‌گاه EVL+trace برای بازیابی سازگاری همه عناصر (از نوع Person) متناظر از مدل مقصد را حذف می‌کند. در این حالت، از نظر فنی وقتی یک شیء حذف می‌شود، اشاره‌گر مربوط به آن در پیوند ردیابی خالی می‌شود. ممکن است کاربر یک شیء از نوع Member را حذف کند، در این حالت، کافی است فرد متناظر با آن از مدل مقصد حذف شود. توجه کنید که ممکن است کاربر از مدل مقصد یک شیء از نوع Person را حذف کند؛ بنابراین، EVL+trace باید شیء متناظر از نوع Person را حذف کند. همچنین، اگر خانواده (شیء Family) متناظر با فرد حذف شده دارای هیچ عضوی نباشد، آن خانواده نیز حذف می‌گردد. توجه کنید که



شکل ۸: نمونه‌ای از مدل‌های محک Families2Persons

صوری‌سازی روابط یک‌به‌یک (همانند رابطه Member با Person) در [۸] به‌طور کامل توضیح داده شده است.

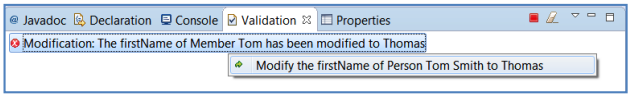
۳-۲-۲- تغییر

پس از بررسی حذف نوبت به بررسی تغییر می‌رسد. اولین حالت ممکن، حالتی است که مقدار صفت firstName یک شیء Member توسط کاربر تغییر یابد. برای تشخیص این تغییر، باید مقدار eqField پیوند ردیابی با صفت firstName عنصر مبدأ مقایسه شود (عبارت مقابل check در لیست ۱). درصورتی‌که، این مقادیر مساوی نباشند، آنگاه کنترل به قسمت fix منتقل می‌گردد. سپس، شیء متناظر با عضو تغییر یافته پیدا می‌شود و مقدار fullName آن تنظیم می‌شود. همچنین، مقدار eqField پیوند ردیابی نیز به‌روز می‌شود.

```
constraint MemberModification
{
  check: self.eqField = self.source.firstName
  message: "Modification: The firstName of Member "+self.eqField
           +" has been modified to "+ self.source.firstName
  fix
  {
    title:"Modify the firstName of Person "+self.eqField+" "+
          self.target.getlastName()+" to "+self.source.firstName
    do
    {
      var person = Persons!Person.all.select
                   (p| p.id() = self.target.id()).first;
      person.fullName = self.source.firstName + ' '
                        +person.getlastName();
      self.eqField = self.source.firstName;
    }
  }
}
```

لیست ۱: قطعه کد تغییر در firstName از شیء Member

نمونه‌ای از اجرای قطعه کد لیست ۱ در شکل ۹ نمایش داده شده است. طبق پیغام نمایش‌داده‌شده، نام کوچک شیء Tom از مدل مبدأ به Thomas توسط کاربر تغییر یافته، راه‌حل پیشنهادی EVL+trace به کاربر در پنجره پایینی بیان می‌کند که نام Tom Smith مربوط به شیء (Person) متناظر از مدل مقصد نیز عوض شود.



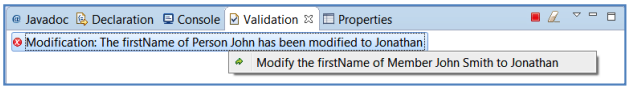
شکل ۹: نمونه اجرای قطعه کد لیست ۱

دومین حالت مربوط است به تغییر در بخش اول fullName یک شیء از نوع Person در مدل مقصد که در مقابل، EVL+trace باید مقدار firstName مربوط به شیء (از نوع Member) متناظر با آن در مدل مبدأ و eqField پیوند ردیابی متناظر را به‌روز کند. قطعه کد مربوط به این نوع تغییر در لیست ۲ آمده است.

```
constraint PersonfirstNameModification
{
  check: self.eqField = self.target.getfirstName()
  message: "Modification: The firstName of Person "+
                 self.eqField+" has been modified to "
                 +self.target.getfirstName()
  fix
  {
    title:"Modify the firstName of Member "+self.eqField+" "+
          self.source.getlastName()+" to "+
          self.target.getfirstName()
    do
    {
      var member = Families!Member.all.select
                   (m| m.id() = self.source.id()).first;
      member.firstName = self.target.getfirstName();
      self.eqField = self.target.getfirstName();
    }
  }
}
```

لیست ۲: قطعه کد تغییر در بخش اول fullName از شیء Person

نمونه‌ای از اجرای لیست ۲ در شکل ۱۰ آورده شده است.



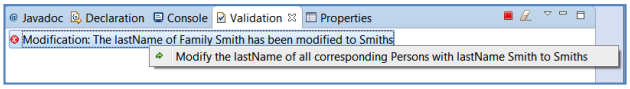
شکل ۱۰: نمونه اجرای قطعه کد لیست ۲

حالت سوم، مربوط به تغییر در نام‌خانوادگی یک شیء از نوع Family است (حالت ۳ از جدول ۱). در این حالت، برای بازیابی سازگاری بخش دوم fullName همه اشیاء (Person) متناظر با شیء تغییر یافته و eqField پیوند ردیابی مربوطه به‌روز می‌شوند. قطعه کد مربوط به بررسی این تغییر و بازیابی سازگاری بین مدل‌ها در لیست ۳ آمده است.

```
constraint FamilyModification
{
  check: self.eqField = self.source.lastName
  message: "Modification: The lastName of Family "+self.eqField+
           " has been modified to "+self.source.lastName
  fix
  {
    title:"Modify all corresponding Persons with lastName "+
          self.eqField+" to "+self.source.lastName
    do
    {
        for(target in self.target)
        {
        var person = Persons!Person.all.select
                     (p|p.id() = target.id()).first;
        person.fullName = person.getfirstName()+
                          ' '+ self.source.lastName;
        }
        self.eqField = self.source.lastName;
    }
  }
}
```

لیست ۳: قطعه کد تغییر در lastName شیء Family

نمونه‌ای از اجرای قطعه کد لیست ۳ در شکل ۱۱ نمایش داده شده است.



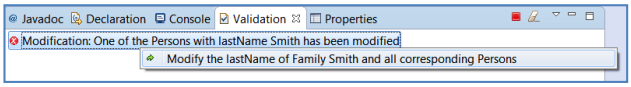
شکل ۱۱: نمونه اجرای قطعه کد لیست ۳

آخرین حالت مربوط به تغییر در بخش دوم fullName یک شیء از نوع Person در مدل مقصد است (حالت ۲ از جدول ۱). در این حالت، EVL+trace باید نام‌خانوادگی خانواده متناظر از مدل مبدأ، بخش دوم fullName بقیه افراد خانواده در مدل مقصد و eqField پیوند ردیابی را به روز کند. قطعه کد لیست ۴، پیاده‌سازی این حالت را نشان می‌دهد.

```
constraint PersonlastNameModification
{
  check: self.target.select(p|
           not (p.getlastName()= self.eqField)).size()=0
  message: "Modification: One of the Persons with lastName "+
                self.eqField+ " has been modified"
  fix
  {
    title: "Modify the lastName of Family "+self.eqField+
          " and all corresponding Persons"
    do
    {
      var target = self.target.select(p|
                     not (p.getlastName()= self.eqField)).first;
      var family = Families!Family.all.select(f|
                     f.id() = self.source.id()).first;
      family.lastName = target.getlastName();
      for(target2 in self.target)
      {
        var person = Persons!Person.all.select(p|
                       p.id() = target2.id()).first;
        person.fullName = person.getfirstName()+
                          ' '+ target.getlastName();
      }
        self.eqField = target.getlastName();
    }
  }
}
```

لیست ۴: قطعه کد تغییر در بخش دوم fullName از شیء Person

نمونه اجرای قطعه کد لیست ۴ در شکل ۱۲ نشان داده شده است.



شکل ۱۲: نمونه اجرای قطعه کد لیست ۴

۳-۲-۳- اضافه کردن

اگر کاربر یک شیء جدید از نوع Member را به یک خانواده اضافه کند، آن‌گاه تبدیل دوسویه باید عنصری در مدل مقصد از نوع Person با جنسیت متناسب و نام هم‌ارز با شیء جدید ایجاد کند. برای تشخیص اشیا جدید، کافی است بررسی کنیم که شناسه شیء توسط هیچ یک از پیوندهای ردیابی مورد ارجاع قرار نگرفته است؛ این بررسی توسط تابع isNew() (لیست ۵) انجام می‌گیرد. لیست ۶، قطعه کد مربوط به بررسی وجود عضوی جدید و اعمال سازگاری (در صورت تشخیص آن) را نشان می‌دهد.

```
operation Families!Member isNew() : Boolean
{
        var seq = EVLTrace!OneToOne.all.select(t|
                    t.source.isKindOf(Families!Member));
        for (s in seq)
              if(s.source.id()= self.id()) return false;
        return true;

}
```

لیست ۵: قطعه کد مربوط تابع isNew() برای نوع Member

توجه کنید که قطعه کد لیست ۶ مربوط به جنسیت Female است و برای Male نیز باید قید جداگانه‌ای نوشته شود. همچنین در این حالت بررسی می‌شود که کاربر شیء متناظر با عضو جدید را خودش از قبل به مدل مقصد اضافه نکرده باشد، زیرا در آن حالت، کافی‌ست فقط یک پیوند ردیابی بین اشیاء جدید در مدل ردیابی ایجاد شود؛ به عبارت دیگر، لازم نیست مانند کد بلوک fix در لیست ۶، یک شیء از نوع Female ساخته شود.

```
constraint FemaleAddition
{
  guard : self.isFemale()
  check: self.hasEquivalentFemalePerson()
  message:"Addition: Member "+self.getfullName()+
          " has no equivalent Person"
  fix
  {
    title:"Create an equivalent Female for "+self.getfullName()
    do
    {
      var female: new Persons!Female;
      female.fullName = self.getfullName();
      Persons!Female.all.add(female);

      self.createMemberPersonTrace(female);
        female.addToFamilyPersonTrace();
    }
  }
}
```

لیست ۶: قطعه کد بررسی نوع جدید Member و ایجاد اشیاء متناظر

از آن‌جایی‌که هر خانواده جدید اضافه شده حداقل دارای یک عضو می‌باشد، نیازی نیست که بررسی شود آیا یک شیء جدید از نوع Family در مدل مبدأ توسط کاربر اضافه شده است یا خیر.

حالت دیگر از عملیات اضافه کردن مربوط به بررسی اشیاء جدید در مدل مقصد می‌باشد. در این حالت، ابتدا بررسی می‌شود که اگر یک شیء جدید از نوع Person در مدل مقصد اضافه شده، حتماً عنصر متناظر با آن از نوع Family در مدل مبدأ وجود داشته باشد و اگر وجود نداشت، آن خانواده اضافه شود (حالت ۱۲ از جدول ۱). قطعه کد مربوط به بررسی و اجرای این امر در لیست ۷ آمده است.

حال برای اضافه کردن اعضای خانواده، همان‌طور که در خط ۳۰ از لیست ۷ می‌بینیم باید قید MemberAddition (لیست ۸) اجرا شود.

```
context Persons!Person
{
  guard: self.isNew()
  constraint FamilyAddition {
    check {
      if(not self.hasEquivalentFamily()) return false;
      else {
          self.satisfies("MemberAddition");
          return true;
        }}
    message: "Addition: New Person "+self.fullName +
                " has no equivalent families"
    fix{
    title:"Create new family for "+self.fullName+
           " if has not been created in before fixes"
    do{
      if(not self.hasEquivalentFamily()){
        var collection = self.getNewlastNames();
        for(name in collection){
            var family : new Families!Family;
            family.lastName = name;
            Families!Family.all.add(family);
            var trace : new EVLTrace!OneToMany;
            trace.name = name;
            trace.category = "FamilyPersonTrace";
            trace.eqField = name;
            trace.source = family;
            EVLTrace!OneToMany.all.add(trace);
          }}
      self.satisfies("MemberAddition");
    }}}}
```

لیست ۷: قطعه کد مربوط به ایجاد خانواده جدید در مدل مبدأ

لیست ۸ برای نوع Female نوشته شده است؛ همان‌طور که مشاهده می‌شود دو راه‌حل (دو بلوک fix) برای بازیابی سازگاری

پیشنهاد می‌شود. راه اول پیشنهاد می‌دهد که عضو جدیدی که قرار است متناظر با این فرد در مدل مبدأ ایجاد شود، نقش مادر خانواده را داشته باشد. راه دوم بیان می‌کند که عضو جدید در نقش دختر به خانواده اضافه گردد. راه‌حل‌ها به‌صورت خودکار به کاربر نشان داده می‌شوند و او تصمیم می‌گیرد که کدام راه را انتخاب کند تا به‌طور خودکار اجرا شود.

```
context Persons!Female{
  guard: self.isNew()
  @lazy
  constraint MemberAddition{
    check: self.hasEquivalentMember()
    message: "Addition: A new Female "+self.fullName+
        " has been added"
    fix{
      title:"Add a corresponding Member for "+ self.fullName+
          " as Mother"
      do{
        var family =self.getEquivalentFamily();
        if (family.mother.isUndefined()){
          var member: new Families!Member;
          member.familyMother = family;
          member.firstName = self.getfirstName();
          Families!Member.all.add(member);
          member.createMemberPersonTrace(self);
          self.addToFamilyPersonTrace();}
        else{
          System.user.inform("Cannot add because the family "
          +self.getlastName()+" has another mother");
          delete self;}
      }}
    fix{
      title:"Add a corresponding Member for "+ self.fullName+
          " as Daughter"
      do{
          var family =self.getEquivalentFamily();
          var member: new Families!Member;
          member.familyDaughter = family;
          member.firstName = self.getfirstName();
          Families!Member.all.add(member);
          member.createMemberPersonTrace(self);
          self.addToFamilyPersonTrace();
      }}}}
```

لیست ۸: قطعه کد مربوط به ایجاد عضوی جدید در مدل مبدأ

۴- مقایسه EVL+trace با روش‌های تبدیل دوسویه

در این بخش قصد داریم قابلیت‌های روش EVL+trace را در مقایسه با روش‌های شناخته‌شده تبدیل مدل دوسویه بیان کنیم. به همین منظور ابتدا، برخی روش‌های تبدیل دوسویه معرفی می‌شوند. سپس، معیارهای مقایسه روش‌های تبدیل مطرح شده و درنهایت، روش پیشنهادی بر اساس آن معیارها با سه روش تبدیل شناخته‌شده مقایسه می‌شود.

۴-۱- روش‌های تبدیل دوسویه

زبان پرس‌وجو/دید/تبدیل رابطه‌ای[۹] زبان رابطه‌ای تعریف‌شده توسط انجمن مدیریت شیء است که نحو آن می‌تواند متنی یا گرافیکی باشد و از تبدیل دوسویه به‌صورت اعلانی پشتیبانی می‌کند [۱۶]. این زبان هنوز دارای ابهامات زیادی است [۶]. یکی از ابزارهایی که این زبان را پیاده‌سازی می‌کند، medini[۱۰] نام دارد. پروژه QVTd پیاده‌سازی در محیط اکلیپس از این زبان است که کامپایلر آن تحت توسعه است.

گرامرهای سه تایی گراف[۱۱] زبانی قدرتمند بر پایه مفاهیم ریاضی برای تعریف تناظر میان دو نوع مدل به روش دوسویه و اعلانی می‌باشد [۱۷]. با بهره‌مندی از شباهت میان گراف و فرامدل، سه گراف برای فرامدل مبدأ، فرامدل مقصد و فرامدل تناظر در نظر گرفته می‌شود. فرامدل تناظر، زبانی را برای تعریف ارتباط میان مدل مبدأ و مقصد فراهم می‌کند. گرامرهای سه‌تایی گراف برای تعریف ارتباط بین دو مدل، تبدیل مدلی به مدل از نوع دیگر، محاسبه تناظر بین دو مدل یا نگه‌داری سازگاری دو مدل استفاده می‌شوند. این زبان، علاوه بر تبدیل مدل، در تجمیع و همگام‌سازی مدل‌ها نیز کاربرد دارد [۵]. ابزارهای TGG ازجمله eMoflon[۱۲]، EMorF[۱۳] و Henshin[۱۴] به‌طور مستقل توسعه یافته‌اند.

زبان تبدیل ژنوس[۱۵] زبانی مبتنی بر قید است که به طور ویژه برای پشتیبانی از مفهوم دوسویگی بنا شده است [۱۸]. پیاده‌سازی آن بر پایه برنامه‌نویسی منطقی است و برای حل مسائل ان پی-سخت مناسب است. تبدیل در این زبان، همانند تبدیل‌های QVT-R می‌تواند از نوع تحمیل یا بررسی باشد. اگر از نوع تحمیل باشد، در جهتی خاص با انتخاب یکی از مدل‌ها به‌عنوان مقصد اجرا می‌شود. با اجرای فرآیند تبدیل، ابتدا برقراری سازگاری روابط بررسی می‌شود و برای آن‌هایی که سازگار نیستند، با ایجاد، حذف و تغییر مدل مقصد، سازگاری دو مدل برقرار می‌شود. این زبان از سازوکار پیشرفته ردیابی حمایت می‌کند که درآن همه عناصر چه آن‌هایی که درگیر تبدیل می‌شوند و چه آن هایی که از فرآیند تبدیل حذف می‌شوند، ذخیره می‌گردند. برتری این زبان در تبدیل دوسویه با مقایسه آن با گراف‌های سه‌تایی، برای برخی از کاربردها ثابت شده است [۱۹].

۴-۲- معیارهای مقایسه

در این بخش، معیارهایی را برای مقایسه روش EVL+trace با سه روش QVT-R [۱۶]، TGG [۱۷] و JTL [۱۸] معرفی می‌کنیم. برخی از این معیارها مرتبط با ساختار زبان تبدیل و برخی دیگر متناسب با ویژگی‌های دوسویگی دارد.

۱. **سبک زبان [۳]**: زبان می‌تواند به سه شکل اعلانی، دستوری یا ترکیبی از این دو باشد. در حالت اعلانی، فقط مشخص می‌گردد که چه عنصری از مبدأ به چه عناصری از مقصد تبدیل شود. در حالت دستوری، چگونگی تبدیل کاملاً مشخص می‌گردد. حالت ترکیبی از مزایای هر دو روش در جهت تعریف کد تبدیل استفاده می‌کند.

۲. **شبه جاوا**: ازآنجایی‌که، اکثر توسعه‌دهندگان مدل رانده نیازمند آشنایی با زبان برنامه‌نویسی جاوا هستند، شبیه بودن نحو زبان تبدیل به این زبان قابلیت یادگیری را افزایش می‌دهد.

۳. **شبه زبان قید شیء**: اکثر طراحان فرامدل، برای تکمیل معنای فرامدل‌ها از زبان قید شیء بهره می‌برند. درنتیجه، با نحو این زبان آشنایی کامل دارند. بنابراین، اگر زبان تبدیل دارای نحو عینی مشابه زبان قید شیء باشد، سادگی و قابلیت یادگیری در آن افزایش می‌یابد.

۴. **پشتیبانی فنی [۲۰]**: درصورتی‌که زبانی دارای وب‌سایت مخصوص برای بیان نمونه‌های کاربردی و پرسش و پاسخ داشته باشد، آن زبان از پشتیبانی فنی بالایی برخوردار است.

۵. **پشتیبانی ابزاری [۲۰]**: داشتن کامپایلر (مترجم) و اشکال‌زدا، توسعه سریع و نصب آسان ازجمله زیرمعیارهای پشتیبانی ابزاری هستند.

۶. **قابلیت حمل [۲۰]**: این معیار از سه دیدگاه سخت‌افزاری، سیستم‌عامل و سازگاری با نسخه‌های اکلیپس موردتوجه قرار می‌گیرد. چون نرم‌افزار اکلیپس پایه ابزارهای تبدیل است و سازگار با هر دو پردازنده ۳۲ بیتی و ۶۴ بیتی است و بر روی سیستم‌عامل‌های ویندوز و لینوکس قابل‌نصب است، صرفاً این معیار را از دیدگاه نسخه‌های متعدد هلیوس، ژونو، ایندیگو، کپلر، لونو و مارس بررسی می‌کنیم.

۷. **پشتیبانی عملگری [۷]**: درصورتی‌که یک زبان از سه عمل حذف، تغییر و اضافه حمایت کند، به‌طور کامل می‌تواند این معیار را ارضا کند.

۸. **قابلیت نگه‌داری [۷]**: درصورتی‌که تغییری در یکی از مدل‌ها داده شود و این تغییر قابلیت انتشار نداشته باشد، باید پس از اجرای تبدیل آن تغییر در مدل موردنظر بدون دست‌خوردگی حفظ شود.

۹. **بقراط‌گونگی[۱۶] [۶]**: این معیار یعنی تبدیل حق ندارد مدل‌ها را تغییر دهد، درحالی‌که، قبل از اجرای تبدیل سازگار باشند.

۱۰. **قابلیت مصالحه[۱۷] [۷]**: درصورتی‌که قبل از اجرای تبدیل نیازمند تعیین جهت اجرا (از مبدأ به مقصد یا بالعکس) باشیم، این ویژگی ارضا نمی‌شود. اما اگر روش دوسویه بتواند هر دو جهت تبدیل روبه‌جلو و روبه‌عقب را همزمان اجرا کند و راه‌حلی برای مواجهه با تداخل‌های ناسازگار داشته باشد، این ویژگی ارضا می‌شود.

۴-۳- مقایسه روش‌های تبدیل دوسویه

در این بخش، بر اساس ۱۰ معیار بیان‌شده در بخش ۴-۲ روش‌های تبدیل دوسویه EVL+trace، QVT-R، TGG و JTL مقایسه می‌شوند.

۴-۳-۱- روش EVL+trace

برنامه‌های روش EVL+trace به زبان EVL نوشته می‌شوند که دارای سبک ترکیبی اعلانی- دستوری است. قسمت دستوری این زبان بر مبنای زبان شیء اپسیلون که شبه جاوا است، می‌باشد. همچنین، قیود و شروط این زبان همانند زبان قید شیء نوشته می‌شوند. بنابراین، قابلیت یادگیری روش پیشنهادی بسیار بالاست. پشتیبانی فنی از زبان‌های اپسیلون بسیار بالا بوده، زیرا مثال‌ها و نمونه‌های کاربردی زیادی در وب‌سایت[۱۸] این چارچوب وجود دارد و وب‌سایتی[۱۹] برای پرسش و پاسخ راجع به ابزارها و زبان‌های این چارچوب در دسترس است که به سرعت به‌روز می‌شود. همه زیرمعیارهای پشتیبانی ابزاری به‌غیراز اشکال‌زدا توسط چارچوب اپسیلون حمایت می‌شود. قابلیت حمل EVL+trace به دلیل بالا بودن قابلیت حمل اپسیلون قابل‌توجه است زیرا این چارچوب با همه نسخه‌های اکلیپس سازگار است. درنتیجه، همه برنامه‌های EVL+trace بدون هیچ تغییری می‌توانند روی همه نسخه‌ها اجرا گردند. همان‌طورکه در بخش ۳ دیدیم، این روش از سه عمل حذف، تغییر و اضافه پشتیبانی می‌کند. چون قیود مربوط به تبدیل روبه‌جلو و روبه‌عقب در برنامه EVL تعریف می‌شوند، اجرای دو جهت تبدیل در یک زمان انجام می‌گیرد (قابلیت مصالحه) و تداخل‌ها به‌صورت تعاملی رفع می‌گردند. قیود فقط روی عناصر درگیر در تبدیل تعریف و اجرا می‌شوند و درنتیجه، روش پیشنهادی دخالتی در تغییرات غیرقابل‌انتشار ندارد و قابلیت نگه‌داری را پشتیبانی می‌کند. ازآن‌جایی‌که، هر قید قبل از اعمال تبدیل به بررسی سازگاری می‌پردازد، درصورتی‌که مدل‌ها سازگار باشند، هیچ عملی انجام نمی‌دهد و بقراط‌گونگی ارضا می‌شود.

۴-۳-۲- روش QVT-R

سبک زبان QVT-R اعلانی بوده و نحو آن شباهت زیادی به زبان قید شیء دارد. پشتیبانی فنی این زبان به دلیل وجود ابهاماتی در آن [۶] بالا نیست، اما وب‌سایتی[۲۰] برای پرسش و پاسخ در اختیار است که به اندازه اپسیلون فعال نیست. ابزار medini، همه زیرمعیارهای پشتیبانی ابزاری را به‌غیراز توسعه سریع فراهم می‌کند. ابزارهای QVT-R یا مانند medini منسوخ شده‌اند و یا مانند QVTd هنوز به‌طور کامل پیاده‌سازی نشده‌اند؛ بنابراین، این روش دارای قابلیت حمل پایینی است.

این زبان، پشتیبانی عملگری را برای هر سه عمل برآورده می‌کند و چون قبل از تحمیل تبدیل به بررسی سازگاری مدل‌ها می‌پردازد، قابلیت بقراط‌گونگی را نیز ارضا می‌کند. برای برنامه‌های این زبان باید جهت تبدیل قبل از اجرا مشخص گردد، درنتیجه، قابلیت مصالحه را ندارد. ازآن‌جایی‌که، QVT-R مدل مقصد را از ابتدا ایجاد می‌کند، تغییرات غیرقابل‌انتشار را از بین برده و قابلیت نگه‌داری را برآورده نمی‌کند.

۴-۳-۳- روش TGG

سبک زبان TGG اعلانی و نحو آن گرافیکی است که در برخی از پیاده‌سازی‌ها از قیود زبان قید شیء برای تکمیل معنای تبدیل بهره می‌برد. ابزارهای TGG به‌طور مستقل توسعه داده شده‌اند و با هم همخوانی ندارند؛ به‌علاوه، پشتیبانی فنی خاصی از آن‌ها صورت نمی‌گیرد و توسعه‌دهندگان صرفاً از طریق پست الکترونیکی به حل چالش‌های این زبان می‌پردازند.

اگرچه، ابزارهای متعدد این زبان به خوبی قابلیت‌های آن را پیاده‌سازی کرده‌اند اما پیشرفتی در توسعه ابزاری این زبان وجود ندارد و درنتیجه، با نسخه‌های جدیدتر اکلیپس سازگار نیست. این زبان پشتیبانی عملگری را برآورده می‌کند، اما قابلیت نگه‌داری، مصالحه و بقراط‌گونگی را پشتیبانی نمی‌کند.

۴-۳-۴- روش JTL

نحو زبان JTL کاملاً مشابه با QVT-R بوده، اگرچه زبان کوچک‌تر و ساده‌تری است و همه قابلیت‌های دستوری QVT-R را حمایت نمی‌کند. تبدیل نوشته‌شده به این زبان به نمونه‌ای از یک مسئله

منطقی ترجمه می‌شود و توسط الگوریتم‌های حل آن مسئله پاسخ داده می‌شود. سپس، جواب آن به شکل قابل‌فهمی از فرمت‌های مدل رانده ترجمه می‌گردد. اگرچه این زبان پیاده‌سازی شده، اما به‌صورت ابزاری جامع در اختیار کاربران قرار ندارد و پشتیبانی فنی از آن نمی‌شود. این زبان از هر سه عمل حذف، اضافه و تغییر پشتیبانی کرده و قابلیت بقراط‌گونگی را به‌تبع از QVT-R حمایت می‌کند. چون جهت تبدیل قبل از اجرا مشخص می‌شود، قابلیت مصالحه را نداشته و ازآن‌جایی‌که، از ۲۰۱۱ به‌روز نشده، قابلیت حمل را نیز برآورده نمی‌کند. روش JTL، مدل مقصد را از اول ایجاد می‌کند، به همین دلیل، نمی‌تواند از قابلیت نگه‌داری حمایت کند.

۴-۳-۵- نتایج مقایسه

بر اساس ویژگی‌هایی که برای چهار روش تبدیل دوسویه بیان شد، خلاصه‌ای از مقایسه در جدول ۲ نشان داده شده است.

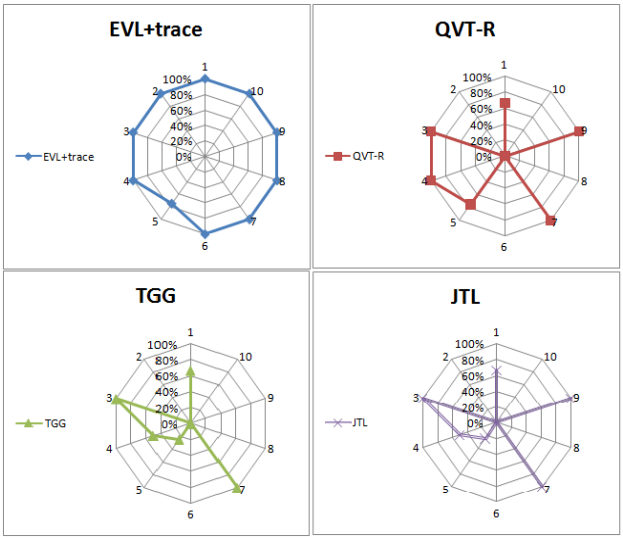
جدول ۲: نتایج مقایسه روش‌های تبدیل دوسویه

✓ = قابل‌پشتیبانی، ✗ = غیرقابل‌پشتیبانی

معیارهای مقایسه	EVL+trace	QVT-R	TGG	JTL
سبک زبان	ترکیبی	اعلانی	اعلانی	اعلانی
شبه جاوا	✓	✗	✗	✗
شبه زبان قید شیء	✓	✓	✓	✓
پشتیبانی فنی (نمونه کاربردی)	✓	✓	✓	✓
پشتیبانی فنی (پرسش و پاسخ)	✓	✓	✗	✗
پشتیبانی ابزاری (کامپایلر)	✓	✓	✓	✓
پشتیبانی ابزاری (اشکال‌زدا)	✗	✓	✗	✗
پشتیبانی ابزاری (توسعه سریع)	✓	✗	✗	✗
پشتیبانی ابزاری (نصب آسان)	✓	✓	✗	✗
قابلیت حمل	✓	✗	✗	✗
پشتیبانی‌عملگری	✓	✓	✓	✓
قابلیت نگه‌داری	✓	✗	✗	✗
بقراط‌گونگی	✓	✓	✗	✓
قابلیت مصالحه	✓	✗	✗	✗

با توجه به نتایج مقایسه، نمودار رادار هر یک از روش‌ها به صورت شکل ۱۳ می‌باشد. اعداد ۱ تا ۱۰ در این نمودار نمایش‌دهنده معیارهای بخش ۴-۲ هستند. معیارهای شبه جاوا، شبه زبان قید شیء، قابلیت حمل، نگهداری، مصالحه، بقراط‌گونگی و پشتیبانی عملگری می‌توانند یکی از دو مقدار ۰٪ یا ۱۰۰٪ را بگیرند.

سبک زبان برای حالت ترکیبی ۱۰۰٪، برای حالت اعلانی ۶۶/۷٪ و برای حالت دستوری ۳۳/۳٪ را می‌گیرد (زیرا حالت ترکیبی بهتر از اعلانی و اعلانی بهتر از دستوری است). درصورتی‌که، هر دو زیرمعیار پشتیبانی فنی ارضا شود مقدار ۱۰۰٪ و درصورتی‌که یکی ارضا گردد ۵۰٪ برای آن در نظر گرفته می‌شود. برای پشتیبانی ابزاری درصورتی‌که هر یک از زیرمعیارها ارضا گردند، ۲۵٪ به مقدار ارزیابی اضافه می‌شود.



شکل ۱۳: نمودار رادار چهار روش تبدیل دوسویه

همان‌طور که در شکل ۱۳ مشاهده می‌شود، روش پیشنهادی در مقایسه با سایر روش‌ها قابلیت‌های بیش‌تری را در اختیار توسعه‌دهندگان تبدیل در اختیار قرار می‌دهد.

۵- نتیجه‌گیری

در این مقاله، روش EVL+trace بر مبنای زبان اعتبارسنجی چارچوب اپسیلون و تکنیک‌های ردیابی‌پذیری به‌صورت مشروح ارائه و پیاده‌سازی شد. این روش می‌تواند روابط یک‌به‌چند و چندبه‌یک برای تبدیل را با توجه به صوری‌سازی ارائه‌شده پشتیبانی کند. برای نمایش چگونگی استفاده از این صوری‌سازی، روش EVL+trace روی محک Families2Persons پیاده‌سازی شد. همچنین، معیارهایی برای زبان‌ها و روش‌های تبدیل دوسویه معرفی شد که بر اساس آن‌ها، روش پیشنهادی با روش‌های شناخته‌شده تبدیل دوسویه مقایسه شد. نتایج ارزیابی روش نشان می‌دهد که EVL+trace قابلیت‌هایی ازجمله نگه‌داری، بقراط‌گونگی، مصالحه، قابلیت حمل را به‌طور ویژه پشتیبانی می‌کند؛ درحالی‌که، سایر روش‌های تبدیل مطرح چنین قابلیتی ندارند.

ازآن‌جایی‌که، نحو زبان اعتبارسنجی اپسیلون بسیار شبیه به زبان قید شیء و جاوا می‌باشد، کار با EVL+trace برای توسعه‌دهندگان تبدیل راحت‌تر از روش‌هایی همانند TGG [۵، ۱۷] - که دارای یک نمادسازی خاص هستند- می‌باشد. چون چارچوب اپسیلون با همه نسخه‌های اکلیپس حتی جدیدترین آن سازگار است، قابلیت حمل EVL+trace بسیار بالاست؛ درحالی‌که ابزارهای QVT-R و TGG [۶، ۱۷] یا منسوخ شده‌اند و یا پشتیبانی کاملی از ویژگی‌های دوسویگی ندارند و به‌طور کامل پیاده‌سازی نشده‌اند. برای تحقق تبدیل دوسویه و همگام‌سازی بین مدل‌ها سه عملیات حذف، تغییر و اضافه کردن مربوط به هر دو سوی تبدیل، در یک برنامه EVL تعریف می‌شوند، تا در حین پشتیبانی ویژگی انتشار تغییر، بتوان در یک زمان تداخل

و ناسازگاری‌های بین دو مدل را مرتفع ساخت. این روش، تصمیمات کاربر را برای رفع تداخل در نظر می‌گیرد و ازاین‌رو به‌عنوان اولین روش تعاملی دوسویه شناخته شده است. برای بالا بردن سطح تجرید روش EVL+trace، به‌عنوان کارهای آینده پیشنهاد می‌شود که از تکنیک‌های مدل رانده جهت خودکارسازی تولید کد برنامه EVL استفاده شود. در این حالت، توسعه‌دهنده می‌تواند با ایجاد مدل تبدیل در سطح انتزاع بالاتر، برنامه تبدیل دوسویه را با استفاده از یک تولیدکننده کد ایجاد کند و آن را روی مدل‌های متعدد اجرا نماید.

مراجع

[۱] زهرا اسلامی مشکنانی، اشکان سامی، «تأثیر اندازه‌های طراحی نسبت به اندازه‌های کد در بهبود کارایی سامانه‌های آزمون خودکار»، *مجله مهندسی برق تبریز*، دوره۴۲، شماره۱، صفحه۱۳-۲۵، ۱۳۹۱.

[۲] زینب اسمعیل‌پور، اشکان سامی، «گسترش ابزارهای شناسایی الگوهای طراحی با عملگر تصحیح برچسب»، *مجله مهندسی برق تبریز*، دوره ۴۵، شماره ۲، صفحه ۱۱-۲۶، ۱۳۹۴.

[3] M. Brambilla, J. Cabot and M. Wimmer, *Model-Driven Software Engineering in Practice*. Synthesis Lectures on Software Engineering, Morgan & Claypool Publishers, 2012.

[4] B. Zamani, *On verifying the use of a pattern language in model driven design*, PhD thesis, Concordia University, July 2009.

[5] H. Giese and R. Wagner, "From model transformation to incremental bidirectional model synchronization," *Software & Systems Modeling*, vol. 8, no. 1, pp. 21–43, 2009.

[6] P. Stevens, "Bidirectional Model Transformations in QVT: Semantic Issues and Open Questions," *in Model Driven Engineering Languages and Systems*, vol. 4735 of Lecture Notes in Computer Science, pp. 1–15, 2007.

[7] S. Hidaka, M. Tisi, J. Cabot and Z. Hu, "Feature-based classification of bidirectional transformation approaches," *Software & Systems Modeling*, pp. 1–22, 2015.

[8] L. Samimi-Dehkordi, B. Zamani and S. Kolahdouz-Rahimi, "From trace-based inter-model validation to bidirectional model synchronization with reconciliation," *in the 5th International eConference on Computer and Knowledge Engineering (ICCKE 2015)*, pp. 123-130, 2015.

[9] J. Bézivin, "In Search of a Basic Principle for Model Driven Engineering," *UPGRADE – The European Journal for the Informatics Professional*, vol. 5, no. 2, pp. 21–24, 2004.

[10] S. Balsamo, A. di Marco, P. Inverardi and M. Simeoni, "Model-based performance prediction in software development: a survey," *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 295–310, 2004.

[11] S. Sendall and W. Kozaczynski, "Model Transformation: The Heart and Soul of Model-Driven Software Development," *Software*, vol. 20, no. 5, pp. 42–45, 2003.

[12] K. Czarnecki, J. Foster, Z. Hu, R. Lämmel, A. Schürr and J. Terwilliger, "Bidirectional Transformations: A Cross-Discipline Perspective," *in Theory and Practice of Model Transformations (R. Paige, ed.)*, vol. 5563 of Lecture Notes in Computer Science, pp. 260–283, 2009.

[13] P. Stevens, "A Landscape of Bidirectional Model Transformations," *in Generative and Transformational Techniques in Software Engineering II*, vol. 5235 of Lecture Notes in Computer Science, pp. 408–424, 2008.

[14] D. Kolovos, R. Paige and F. Polack, "On the evolution of ocl for capturing structural constraints in modelling languages," *in Rigorous Methods for Software Construction and Analysis*, vol. 5115 of Lecture Notes in Computer Science, pp. 204–218, 2009.

[15] C. Poskitt, M. Dodds, R. F. Paige and A. Rensink, "Towards rigorously faking bidirectional model transformations," *in Proceedings of the Workshop on Analysis of Model Transformations*, pp. 70–75, 2014.

[16] I. Kurtev, "State of the Art of QVT: A Model Transformation Language Standard," *in Applications of Graph Transformations with Industrial Relevance*, vol. 5088 of Lecture Notes in Computer Science, pp. 377–393, 2008.

[17] A. Schürr and F. Klar, "15 Years of Triple Graph Grammars," *in Graph Transformations*, vol. 5214 of Lecture Notes in Computer Science, pp. 411–425, 2008.

[18] A. Cicchetti, D. Ruscio, R. Eramo and A. Pierantonio, "JTL: A Bidirectional and Change Propagating Transformation Language," *in Software Language Engineering*, vol. 6563 of Lecture Notes in Computer Science, pp. 183–202, 2011.

[19] R. Eramo and A. Bucaioni, "Understanding bidirectional transformations with TGGs and JTL," *Electronic Communications of the EASST*, vol. 57, 2013.

[20] R. W. Sebesta, *Concepts of Programming Languages*, 10th Ed., Pearson, 2012.

زیرنویس‌ها

[1] Model-driven development

[2] Epsilon Validation Language (EVL)

[3] Traceability

[4] www.eclipse.org/atl/atlTransformations/#Families2Persons

[5] Metamodel

[6] Meta-metamodel

[7] Bidirectional Transformation (Bx)

[8] Object Constraint Language (OCL)

[9] Query/View/Transformation Relational (QVT-R)

[10] http://projects.ikv.de/qvt

[11] Triple Graph Grammars (TGG)

[12] www.emoflon.org/emoflon/

[13] emorf.org/

[14] de-tu-berlin-tfs.github.io/Henshin-Editor/

[15] Janus Transformation Language (JTL)

[16] Hippocracticness

[17] Reconciliation

[18] http://www.eclipse.org/epsilon/

[19] https://www.eclipse.org/forums/index.php/f/22/

[20] https://www.eclipse.org/forums/index.php/f/243/