روشی مبتنی بر فاصله برای رفع ناسازگاری مدل

رضا گرگان محمدی و احمد عبداله‌زاده بارفروش

چکیده: کاربرد رویکرد مدل‌گرا در تولید نرم‌افزار به دلیل کاهش پیچیدگی و افزایش سرعت تولید به طور جدی مورد توجه قرار گرفته است. یکی از چالش‌های مهم در کاربرد مدل وجود ناسازگاری است. یک ناسازگاری به دلیل وجود الگوهای ساختاری نامطلوب در مدل بروز می‌یابد. روش‌های فعلی ارائه‌شده برای رفع ناسازگاری مدل به میزان تغییرات مدل و فاصله بین مدل و ترمیم آن توجه ندارند. در این مقاله روشی مبتنی بر فاصله برای یافتن نزدیک‌ترین ترمیم نسبت به مدل ناسازگار ارائه می‌شود. برای این منظور مدل و فرامدل با استفاده از گراف جهت‌دار بازنمایی شده و از قواعد تبدیل گراف جهت رفع ناسازگاری بهره برده می‌شود. همچنین معیار فاصله بر اساس میزان تغییرات گراف متناظر مدل تعریف می‌شود. اعمال روش پیشنهادی به مجموعه‌ای از مدل‌های مبتنی بر BPMN بر بهبود نتایج با استفاده از معیار فاصله دلالت دارد.

کلید واژه: **تولید نرم‌افزار مبتنی بر مدل، ناسازگاری، نزدیک‌ترین ترمیم، معیار فاصله.**

۱- مقدمه

در رویکرد توسعه نرم‌افزار مبتنی بر مدل تأکید اساسی بر استفاده از مدل به عنوان مهم‌ترین فراورده و استخراج مستقیم سایر فراورده‌های مورد نیاز (نظیر کد منبع نرم‌افزار) از مدل است [۱]. ارتقای سطح انتزاع از سطح کد منبع به سطح مدل باعث بهبود قابلیت تولید نرم‌افزار می‌شود. مدل بر اساس یک زبان مدل‌سازی تعریف می‌شود که می‌تواند همه‌منظوره باشد (نظیر UML) یا برای یک دامنه خاص طراحی گردد (نظیر BPMN).

مدیریت ناسازگاری به عنوان یکی از چالش‌های مهم در کاربرد مدل مطرح است و وجود این ناسازگاری‌ها مانعی در استفاده از مدل محسوب می‌شود. یک ناسازگاری یک الگوی ساختاری نامطلوب است که باعث نقض قواعد ساختاری زبان مدل‌سازی می‌گردد که در قالب فرامدل تعریف می‌شود. فرایند مدیریت ناسازگاری شامل سه مرحله تعریف ناسازگاری، تشخیص و در نهایت رفع ناسازگاری است [۲]. تشخیص ناسازگاری از طریق بررسی وجود نواقص ساختاری در مدل صورت می‌گیرد. هدف از رفع ناسازگاری تغییر ساختار مدل به گونه‌ای است که در آن قواعد ناسازگاری برقرار نباشد. همچنین باید توجه داشت که رفع یک ناسازگاری خود می‌تواند باعث بروز ناسازگاری دیگر شود. در یک مدل ممکن است ناسازگاری‌های متفاوت و متعددی وجود داشته باشد و بنابراین برای رفع آن به روش‌های خودکار یا نیمه‌خودکار نیاز داریم.

برای رفع ناسازگاری گاهی لازم است چندین تغییر متوالی اعمال شود. مجموعه این تغییرات در نهایت باعث ایجاد یک مدل می‌شود که در این مقاله تحت عنوان یک ترمیم برای مدل اولیه شناخته می‌شود. وجود روش‌های متفاوت برای رفع ناسازگاری‌ها باعث ایجاد ترمیم‌های مختلف برای یک مدل ناسازگار می‌شود. در فرایند رفع ناسازگاری ممکن است برخی ترمیم‌ها ساختار مدل را به طور اساسی تحت تأثیر قرار دهند. در یک ترمیم بهینه مسئله رفع ناسازگاری باید با کمترین میزان تغییرات مدل اولیه صورت پذیرد. بنابراین لازم است روشی جهت مقایسه این ترمیم‌ها و تعیین نزدیک‌ترین ترمیم[1] بر مبنای معیار فاصله تعریف‌شده وجود داشته باشد. با توجه به استفاده از مدل‌های BPMN در این مقاله، در شکل ۱ مثالی در خصوص یک مدل فرایند مبتنی بر BPMN را ارائه می‌کنیم. در مدل ارائه‌شده در بخش الف این شکل سه ناسازگاری وجود دارد. در بخش ب شکل یک ترمیم برای مدل ناسازگار به همراه عمل‌های اصلاحی مورد نیاز ارائه شده است. قسمت ج از شکل ۱ یک ترمیم دیگر را نشان می‌دهد. در این مثال مدل شکل ج با تغییرات کمتری نسبت به مدل شکل ب ایجاد می‌شود و بنابراین نزدیک‌ترین ترمیم به مدل بخش الف محسوب می‌گردد.

کارهای تحقیقاتی متعددی به مسئله رفع ناسازگاری در مدل اختصاص یافته است. در [۳] روشی برای رفع ناسازگاری با استفاده از تکنیک‌های برنامه‌ریزی خودکار[2] پیشنهاد شده است. استفاده از بازنمایی گراف و کاربرد تبدیل گراف برای رفع ناسازگاری در [۴] مورد بررسی قرار گرفته است. در [۵] نویسندگان مقاله به معرفی روشی جهت پیمایش فضای حالت جهت تعیین ترمیم برای مدل ناسازگار می‌پردازند. مهم‌ترین محدودیت روش‌های موجود عدم توجه به مسئله اختلاف بین یک مدل و ترمیم آن و میزان شباهت آنها است. در فرایند رفع ناسازگاری ممکن است چندین ترمیم به دست آید و لازم است از طریق معیار مناسب شبیه‌ترین ترمیم به مدل اولیه ارائه شود. مهم‌ترین دستاورد این مقاله ارائه روشی مبتنی بر فاصله برای رفع ناسازگاری مدل است. این روش با تعریف یک معیار فاصله امکان مقایسه بین ترمیم‌های مختلف را فراهم نموده و در نهایت یک ترمیم را با کمترین میزان تغییرات (نزدیک‌ترین ترمیم) ارائه می‌دهد. رویکرد مورد استفاده در این مقاله بر اساس ساختار[3] زبان مدل‌سازی است و ناسازگاری‌های معنایی[4] در نظر گرفته نمی‌شود.

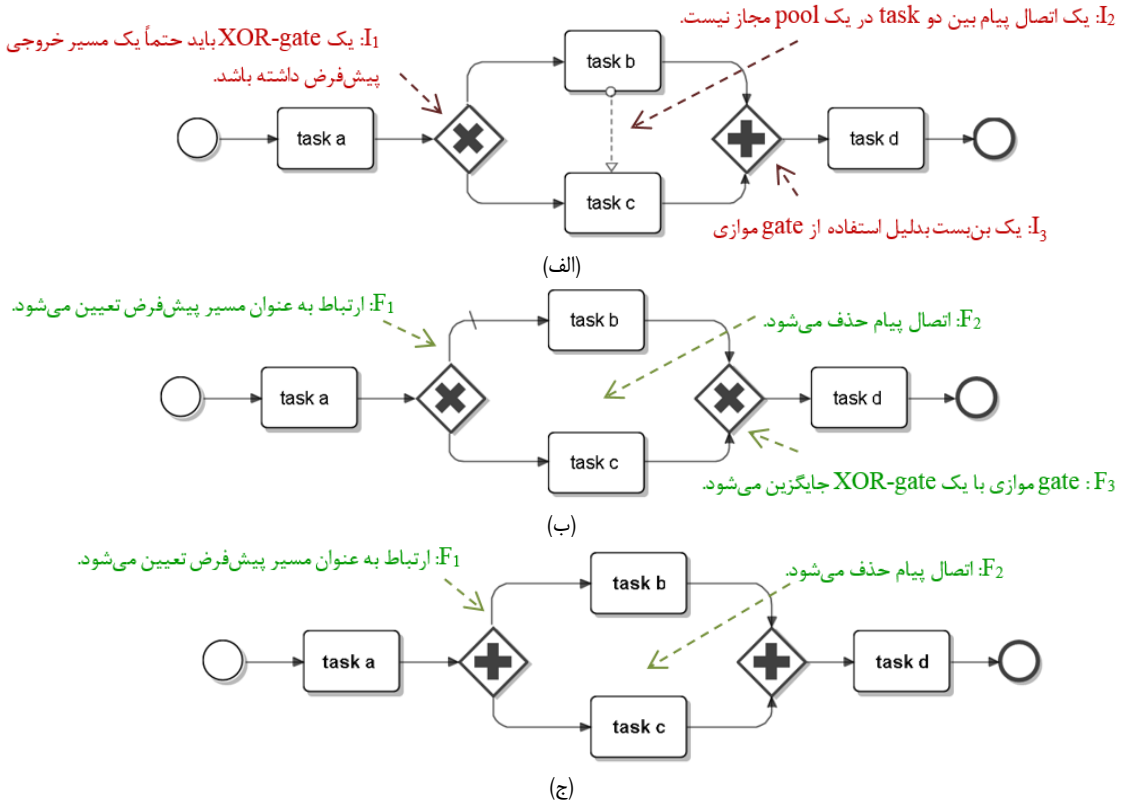
روش پیشنهادی در این مقاله بر اساس چارچوب تبدیل مدل مبتنی بر VIATRA۲[5] توسعه یافته است. این چارچوب قبلاً در مقاله [۴] برای این منظور استفاده شده است. این چارچوب امکاناتی برای تعریف الگوهای گراف و نیز موتور تبدیل گراف را فراهم می‌نماید. ویژگی مهم دیگر این چارچوب که استفاده از آن را پشتیبانی می‌کند امکان تعریف مسایل ارضای قیود به طور مستقیم بر روی مدل است [۶].

این مقاله در تاریخ ۲۱ بهمن ماه ۱۳۹۲ دریافت و در تاریخ ۱۰ آبان ماه ۱۳۹۳ بازنگری شد.

رضا گرگان محمدی، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، (email: reza_gorgan@aut.ac.ir).

احمد عبداله‌زاده بارفروش، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، (email: ahmad@aut.ac.ir).

1. Closest Repair
2. Automated Planning
3. Syntax
4. Semantic-Based
5. http://www.eclipse.org/viatra2/



شکل ۱: یک مثال از رفع ناسازگاری مدل، (الف) یک مدل BPMN ناسازگار، (ب) یک ترمیم برای مدل الف با سه تغییر و (ج) یک ترمیم برای مدل الف با دو تغییر (نزدیک‌ترین ترمیم).

ساختار این مقاله بدین شرح است: در بخش ۲ فرمول‌بندی مسئله ارائه می‌گردد. بخش ۳ به معرفی روش توصیف، تشخیص و رفع ناسازگاری با توجه به بازنمایی مدل و فرامدل مبتنی بر گراف می‌پردازد. در بخش ۴ ایده اصلی مقاله در خصوص ارائه روش یافتن نزدیک‌ترین ترمیم توصیف می‌شود. بررسی صحت و کمال روش پیشنهادی با استفاده از تکنیک‌های صوری در بخش ۵ صورت می‌پذیرد. نتایج اعمال روش پیشنهادی به مدل‌های مورد نظر در مطالعه موردی و مقایسه نتایج با کارهای مشابه در بخش ۶ ارائه می‌شود. بخش ۷ به مرور کارهای مرتبط اختصاص دارد و در نهایت جمع‌بندی و معرفی کارهای پژوهشی آینده در بخش ۸ انجام می‌گیرد.

۲- فرمول‌بندی مسئله

در این بخش تعاریف مورد استفاده برای فرمول‌بندی مسئله به همراه تعریف مفاهیم مرتبط ارائه می‌گردد. در این مقاله از گراف جهت‌دار و مفاهیم نظریه گراف [۷] و نظریه مجموعه‌ها جهت ارائه تعاریف بهره گرفته می‌شود.

تعریف ۱ (گراف جهت‌دار[1]): یک گراف جهت‌دار G به صورت $G=(V,E,src,trg)$ شامل یک مجموعه V از رئوس و یک مجموعه E شامل یال‌های گراف است. دو تابع $src,trg:E\to V$ به ترتیب مبدأ و مقصد هر یال را در گراف مشخص می‌کند.

دو گراف را می‌توان از طریق یک نگاشت گراف به هم مرتبط نمود. یک نگاشت گراف شامل نگاشت بین رئوس و یال‌ها است به گونه‌ای که مبدأ و مقصد هر یال را حفظ نماید.

تعریف ۲ (نگاشت گراف[2]): با در اختیار داشتن دو گراف G و H به صورت $G=(V_G,E_G,src_G,trg_G)$ و $H=(V_H,E_H,src_H,trg_H)$ یک نگاشت گراف به شکل $\mu:G\to H$ شامل دو تابع $\mu_V:V_G\to V_H$ و $\mu_E:E_G\to E_H$ است به طوری که مبدأ و مقصد هر یال حفظ شود، به عبارت دیگر $src_H\circ\mu_E=\mu_V\circ src_G$ و $trg_H\circ\mu_E=\mu_V\circ trg_G$.

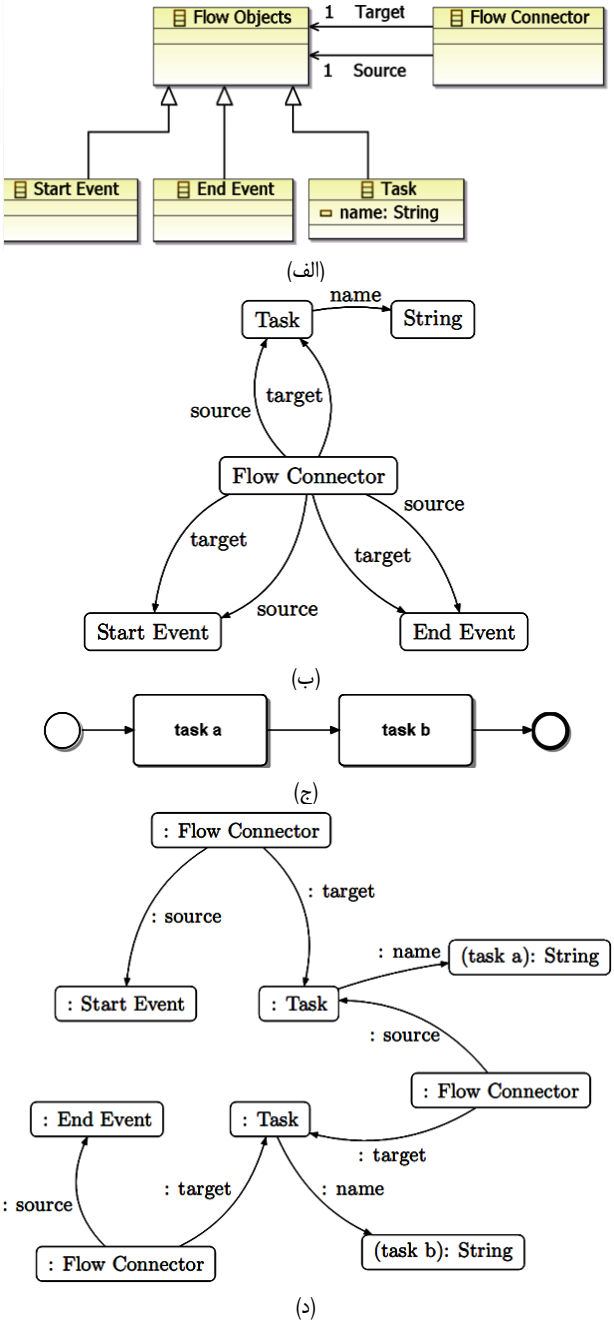
امکان اختصاص یک نوع[3] به یال‌ها و رئوس گراف وجود دارد [۷]. انتساب نوع به یک گراف از طریق تعریف گراف نوع[4] و تعریف نگاشت مربوط قابل انجام است.

تعریف ۳ (گراف نوع‌دار[5]): یک گراف نوع TG عبارت است از یک گراف جهت‌دار که در آن V_{TG} و E_{TG} شامل انواع یال‌ها و انواع رئوس است. یک گراف نوع‌دار یک زوج به صورت $(A,type)$ است که در آن A یک نمونه گراف دارای نوع بر اساس TG است که از طریق نگاشت گراف به صورت $type:A\to TG$ به TG مرتبط می‌شود.

دو مفهوم کلیدی در این مقاله مدل و فرامدل[6] است. یک فرامدل ساختار انتزاعی[7] یک مدل را تعریف می‌کند. یک توصیف فرامدل شامل تعریف فرامدل و نیز تعریف قواعد مرتبط با آن است.

تعریف ٤ (توصیف فرامدل[8]): یک توصیف فرامدل به صورت زوج $M=(TG_M,C)$ است که در آن TG_M یک گراف نوع و C مجموعه قواعد ناسازگاری مرتبط با فرامدل است.

1. Directed Graph
2. Graph Morphism
3. Type
4. Type Graph
5. Typed Graph
6. Metamodel
7. Abstract Syntax
8. Metamodel Specification



شکل ۲: بازنمایی مدل و فرامدل با استفاده از گراف نوع‌دار و گراف نوع، (الف) بخشی از فرامدل BPMN، (ب) گراف نوع معادل فرامدل مدل در بخش الف، (ج) یک نمونه مدل BPMN و (د) یک گراف نوع‌دار معادل مدل بخش ج.

یک مدل در واقع یک گراف جهت‌دار است که بر اساس گراف فرامدل نوع‌گذاری می‌شود. یک نگاشت گراف ارتباط بین مدل و فرامدل مربوط را مشخص می‌کند.

تعریف ۵ (مدل): با در اختیار داشتن یک توصیف فرامدل به صورت $M=(TG_M, C)$، مدل m یک نمونه از فرامدل M است به صورت $m=(G_m, type_m)$ که G_m یک گراف جهت‌دار و $type_m = G_m \to TG_m$ یک نگاشت گراف است.

در شکل ۲- الف بازنمایی بخشی از فرامدل BPMN با استفاده از نمودار کلاس UML نمایش داده شده است. بخش ب این شکل گراف نوع معادل این فرامدل را نشان می‌دهد. یک نمونه مدل BPMN و بازنمایی گراف معادل آن به ترتیب در بخش‌های ج و د شکل ۲ قابل مشاهده است. سازگاری یک مدل با توجه به تعریف توصیف فرامدل تعریف می‌شود.

تعریف ۶ (مدل ناسازگار[1]): با در اختیار داشتن یک توصیف فرامدل به صورت $M=(TG_M, C)$ یک نمونه مدل m منطبق با M ناسازگار است اگر m حداقل یکی از قیود ناسازگاری تعریف شده در C را ارضا نماید و به عبارت دیگر $m \models c, \exists c \in C$.

برای هر یک از قواعد ناسازگاری یک یا چند عمل اصلاحی قابل تعریف است که با اعمال آنها به مدل ناسازگاری مربوط رفع می‌شود. یک عمل اصلاحی خود شامل مجموعه‌ای از عملیات پایه نظیر اضافه/ حذف یک رأس/ یال در گراف است. نکته مهم در این زمینه وجود وابستگی بین عمل‌های اصلاحی است، یعنی با اعمال یک عمل اصلاحی ممکن است علاوه بر ناسازگاری مربوط یک یا چند ناسازگاری دیگر نیز برطرف شود و یا این که یک یا چند ناسازگاری جدید تعریف شود. به حالت اول "تأثیر جانبی مثبت" و به حالت دوم "تأثیر جانبی منفی" اطلاق می‌شود. گاهی ممکن است در یک مدل چندین ناسازگاری وجود داشته باشد و بنابراین لازم است توالی مشخصی از عمل‌های اصلاحی به مدل اعمال گردد که در نتیجه آن یک ترمیم بالقوه حاصل می‌شود.

تعریف ۷ (ترمیم بالقوه[2]): با در اختیار داشتن توصیف فرامدل $M=(TG_M, C)$ یک نمونه مدل r منطبق بر M به عنوان یک ترمیم بالقوه برای مدل ناسازگار m (منطبق بر M) در نظر گرفته می‌شود اگر r سازگار بوده و از طریق اعمال توالی مشخصی از عمل‌های اصلاحی به دست آید.

با توجه به امکان تعریف چندین عمل اصلاحی برای هر نوع ناسازگاری و همچنین وجود تأثیرات جانبی (مثبت و منفی) ممکن است برای یک مدل ناسازگار چندین ترمیم بالقوه به دست آید. هدف اصلی در این مقاله تعیین نزدیک‌ترین ترمیم نسبت به مدل اولیه است که با کمترین تغییر نسبت به مدل ارائه‌شده توسط طراح ناسازگاری‌های موجود را برطرف می‌نماید. برای یافتن نزدیک‌ترین ترمیم لازم است معیار فاصله تعریف شود.

تعریف ۸ (معیار فاصله): تابع فاصله به صورت $distance: G \times G \to R^+ \cup \{\infty\}$ تعریف شده و دو گراف معادل دو مدل را دریافت کرده و میزان شباهت آنها را به صورت یک عدد حقیقی مثبت یا صفر برمی‌گرداند و دو شرط ذیل برقرار است

$$\forall m_1, m_2 \in G : distance(m_1, m_2) = distance(m_2, m_1)$$
$$\forall m_1, m_2 \in G : distance(m_1, m_2) = 0 \Leftrightarrow m_1 = m_2$$

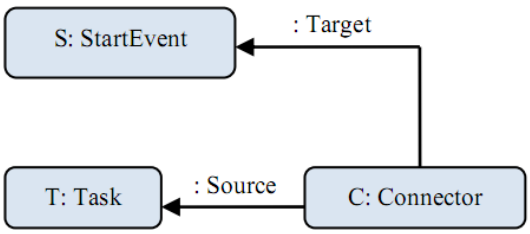
۳- مدیریت ناسازگاری مدل

مدیریت ناسازگاری مدل شامل سه فعالیت تعریف قواعد ناسازگاری، تشخیص ناسازگاری و رفع ناسازگاری است. با توجه به استفاده از گراف جهت‌دار برای بازنمایی مدل و فرامدل، سه فعالیت مورد اشاره با توجه به بازنمایی مدل با استفاده از گراف تشریح خواهد شد.

۳-۱ تعریف قواعد ناسازگاری

دو رویکرد کلی برای تعریف ساختار انتزاعی برای یک زبان مدل‌سازی عبارتند از استفاده از توصیف فرامدل و استفاده از گرامر زبان [۸]. در این مقاله از رویکرد توصیف فرامدل استفاده شده که شامل یک گراف نوع و مجموعه قواعد ناسازگاری مرتبط است. با توجه به بازنمایی مدل با

1. Inconsistent Model
2. Potential Repair



(الف)

```
Pattern StartTaskInEdge (T, S, C, E1, E2)=
  {
Task(T);
StartEvent(S);
Connector(C);
Connector.Source(E1,C,T);
Connector.Target(E2,C,S);
  }
```

(ب)

شکل ۳: تعریف قواعد ناسازگاری با استفاده از الگوی گراف، (الف) بازنمایی گرافیکی الگوی گراف و (ب) توصیف الگوی گراف با استفاده از زبان VTCL.

استفاده از گراف جهت‌دار، قواعد ناسازگاری بر اساس تعریف الگوی گراف[1] قابل توصیف است [۴].

یک الگو در گراف G در واقع یک زیرگراف از این گراف است. بنابراین با استفاده از الگوی گراف امکان توصیف الگوهای ساختاری در یک گراف وجود دارد. یک قاعده ناسازگاری به صورت یک زیرگراف نامطلوب در گراف مربوط در نظر گرفته می‌شود. در این مقاله از چارچوب VIATRA۲ برای پیاده‌سازی روش پیشنهادی استفاده می‌شود. در این چارچوب یک الگوی گراف از طریق کلید واژه Pattern در قالب زبان VTCL[2] قابل توصیف است. به عنوان مثال در شکل ۳ یک الگوی گراف نمایش داده شده است. این الگو شرایطی را توصیف می‌کند که در آن یک رأس از نوع task به یک رأس از نوع startevent از طریق یک رابط متصل می‌شود. در قسمت ب این شکل توصیف این الگو در قالب زبان VTCL نمایش داده شده است. لازم به ذکر است در ساختار یک مدل BPMN یک رأس از نوع startevent نمی‌تواند یال ورودی از سایر رئوس داشته باشد.

۳-۲ تشخیص ناسازگاری

با توجه به این که یک قاعده ناسازگاری از طریق یک الگوی گراف بیان می‌شود می‌توان مسئله تشخیص ناسازگاری را به صورت مسئله انطباق الگوی گراف[3] بیان کرد. در چارچوب VIATRA۲ یک پیاده‌سازی بهینه برای یافتن یک الگو در گراف ارائه شده است. به این ترتیب امکان شناسایی الگوهای نامطلوب (ناسازگاری‌ها) در گراف فراهم می‌شود.

۳-۳ رفع ناسازگاری

سومین فعالیت در مدیریت ناسازگاری وجود روشی جهت رفع ناسازگاری است. تشخیص یک ناسازگاری به تنهایی کفایت نمی‌کند و باید روشی برای رفع یک ناسازگاری وجود داشته باشد. برای این منظور باید متناسب با هر نوع ناسازگاری یک یا چند عمل اصلاحی وجود داشته باشد. با توجه به استفاده از بازنمایی گراف در این مقاله، مسئله رفع ناسازگاری در واقع به صورت تبدیل گراف ناسازگار اولیه به یک گراف فاقد ناسازگاری قابل بیان است. بنابراین رفع ناسازگاری از طریق تبدیل گراف[4] انجام می‌پذیرد. برای هر قاعده ناسازگاری یک یا چند عمل اصلاحی بر اساس قواعد تبدیل گراف بیان می‌شود.

یک قاعده تبدیل گراف شامل دو بخش است: بخش سمت چپ و بخش سمت راست که هر یک بر اساس الگوی گراف قابل توصیف هستند. به بیان ساده در هنگام اجرای یک قاعده تبدیل گراف الگوی توصیف‌شده در سمت چپ در گراف مورد نظر شناسایی شده و الگوی توصیف‌شده در سمت راست قاعده تبدیل جایگزین آن می‌شود. در چارچوب VIATRA۲یک قاعده تبدیل گراف با استفاده از کلمه کلیدی GTRule بیان می‌شود و شامل دو بخش پیش‌شرط و پس‌شرط است که هر یک بر اساس الگوی گراف توصیف می‌شوند. در شکل ۴ یک قاعده تبدیل گراف توصیف‌شده با استفاده از زبان VTCL ارائه شده است. بخش الف از این شکل بازنمایی گرافیکی از قاعده تبدیل است. بخش ب توصیف این قاعده در زبان VTCLرا ارائه می‌کند. با اعمال این قاعده جهت ارتباط بین یک رأس از نوع task و یک رأس از نوع startevent عکس می‌شود.

٤- نزدیک‌ترین ترمیم

در فرایند رفع ناسازگاری ممکن است چندین ترمیم برای یک مدل ناسازگار قابل ایجاد باشد که این موضوع به دلیل وجود تأثیرات جانبی و همچنین امکان تعریف چندین عمل اصلاحی برای یک ناسازگاری است. در این مقاله تأکید اصلی بر ارائه روشی جهت تعیین نزدیک‌ترین ترمیم است. یک عمل اصلاحی ممکن است خود دارای تأثیرات جانبی باشد. یک تأثیر جانبی مثبت زمانی است که با اعمال یک عمل اصلاحی نه تنها ناسازگاری مربوط بلکه یک یا چند ناسازگاری دیگر هم برطرف می‌شود و تأثیر جانبی منفی شرایطی است که اعمال یک عمل اصلاحی باعث رفع ناسازگاری مربوط می‌شود و لیکن باعث ایجاد ناسازگاری‌های جدید می‌گردد. مسئله یافتن نزدیک‌ترین ترمیم را می‌توان به صورت یک مسئله جستجو در فضای حالت مشخص تعریف کرد. با توجه به این که رفع ناسازگاری از طریق تبدیل گراف انجام می‌شود، فضای حالت در قالب یک سیستم تبدیل گراف[5] قابل بازنمایی است.

٤-١ الگوریتم پیشنهادی

الگوریتم پیشنهادی برای جستجو در فضای حالت تعریف‌شده در شکل ۵ نمایش داده شده است. این الگوریتم با استفاده از روش جستجوی عمق اول (DFS)[6] عمل می‌کند. الگوریتم شامل دو تابع است: تابع اول یک مدل ناسازگار را دریافت کرده و نزدیک‌ترین ترمیم برای آن را از طریق فراخوانی تابع دوم (سطر ششم) و کاربرد معیار فاصله (سطر هشتم) فراهم می‌کند. تابع دوم به صورت بازگشتی تعریف شده و با دریافت یک مدل ناسازگار و همچنین مجموعه وضعیت‌های نقض قبلی یک مجموعه از ترمیم‌های بالقوه را در اختیار می‌گذارد. معیار فاصله مورد استفاده در الگوریتم در بخش ۴-۲ معرفی می‌شود.

روش کار تابع دوم ($Repair_{potential}$) به این صورت است که برای مدل ورودی ابتدا تمامی ناسازگاری‌های موجود شناسایی شده (سطر ۱۹) و

1. Graph Pattern
2. Viatra Textual Command Language
3. Graph Pattern Matching
4. Graph Transformation
5. Graph Transformation System
6. Depth First Search

Algorithm 1. Closest Repair

```
1  Function Repair_closest (m)
2  Input: an inconsistent model m
3  Output: closest repair to model m
4  Begin
5      R ← ∅
6      R ← R ∪ Repair_potential(m,∅)
7      Remove the Undo Repair (if any)
8      return c ← arg min_r {distance(m,r), r ∈ R}
9  End
10 Function Repair_potential (m,Z)
11 Input: an inconsistent model m, the set Z of previous violation
   states
12 Output: a set of potential repairs for model m
13 Init
14     P ← ∅
15 Begin
16     if model m is consistent then
17         return {m}
18     end if
19     Initialize V with violations in model m
20     Generate the violation state S using V
21     if S ∈ Z then
22         return ∅
23     end if
24     Z ← Z ∪ S
25     for each v ∈ V do
26         op ← get the fixing action to remedy v
27         m ← ApplyOperator (m,op)
28         P ← P ∪ Repair_potential (m,Z)
29         m ← Undo (m,op)
30     end for
31     return P
32 End
```

شکل ۵: الگوریتم پیشنهادی برای یافتن نزدیک‌ترین ترمیم برای مدل ناسازگار.

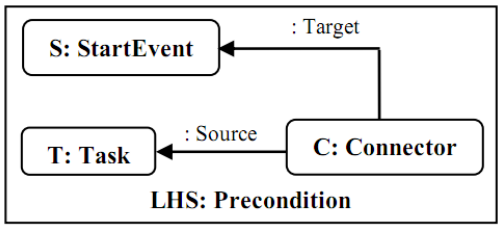
برای هر یک از این ناسازگاری‌ها در یک حلقه تکرار ابتدا عمل اصلاحی مرتبط اعمال می‌شود (سطر ۲۶ و ۲۷) و با رفع یک یا چند ناسازگاری، مدل جدید مجدداً به عنوان ورودی در فراخوانی بازگشتی تابع استفاده می‌شود (سطر ۲۸). دستور Undo (m,op) تأثیر آخرین عمل اصلاحی بر روی مدل m را خنثی می‌کند تا امکان اجرای منطق جستجو فراهم شود (سطر ۲۹). به منظور توقف مسیرهای جستجوی بی‌نتیجه و جلوگیری از ایجاد حلقه‌های بی‌پایان از مجموعه وضعیت‌های نقض قبلی استفاده می‌شود (سطر ۲۰ الی ۲۴).

تعریف ۹ (وضعیت نقض[1]): یک وضعیت نقض S یک مجموعه از دوتایی‌ها به صورت (I_i, n_i) است که در آن I_i قاعده ناسازگاری نقض شده و n_i نشان‌دهنده تعداد دفعات وجود یک ناسازگاری از نوع I_i در مدل است.

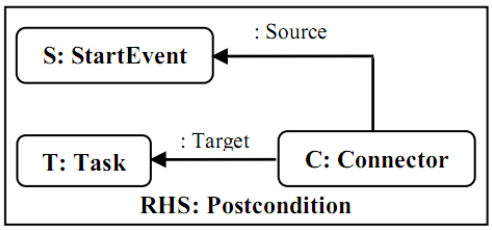
در هر مرحله از اجرای تابع دوم یک وضعیت نقض بر اساس ناسازگاری‌های تشخیص داده شده ایجاد می‌شود (سطر ۲۰). این وضعیت با مجموعه وضعیت‌های قبلی برای تشخیص تکراری‌بودن مقایسه می‌شود (سطر ۲۱). در صورتی که وضعیت تکراری باشد نتیجه تهی بازگشت داده شده (سطر ۲۲) و در غیر این صورت وضعیت جدید به مجموعه وضعیت‌های قبلی افزوده می‌شود (سطر ۲۴).

نکته مهم دیگر در خصوص این الگوریتم وجود امکان ایجاد یک ترمیم با خنثی‌کردن تغییرات کاربر روی مدل است. در واقع یک مدل اولیه ممکن است بر اثر تغییرات اعمالی یک کاربر دچار ناسازگاری شود.

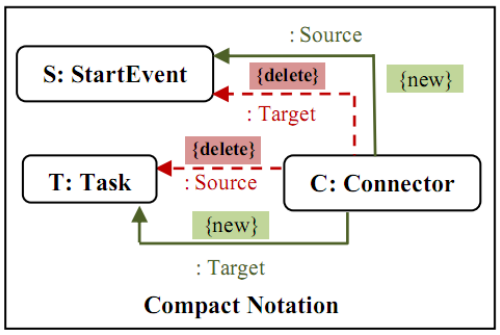
1. Violation State



(الف)



(ب)



(ج)

```
Gtrule ReverseEdgeDirection (in T, in S, in C) =
 {
precondition pattern StartTaskInEdge (T, S, C,
E1, E2)=
    {
Task(T);
StartEvent(S);
Connector(C);
Connector.Source(E1,C,T);
Connector.Target(E2,C,S);
    }
postcondition pattern StartTaskOutEdge (T, S, C,
E1, E2)=
    {
Task(T);
StartEvent(S);
Connector(C);
Connector.Target(E3,C,T);
Connector.Source(E4,C,S);
    }
}
```

(د)

شکل ۴: تعریف یک عمل اصلاحی با استفاده از قاعده تبدیل گراف، (الف) قسمت سمت چپ (پیش‌شرط) قاعده تبدیل گراف، (ب) قسمت سمت راست (پس‌شرط) قاعده تبدیل گراف، (ج) نمایش فشرده قاعده تبدیل گراف و (د) توصیف قاعده تبدیل گراف در زبان VTCL.

بدیهی‌ترین شکل برای رفع ناسازگاری‌ها خنثی‌کردن اقدامات انجام‌شده توسط کاربر در محیط مدل‌سازی است. لازم به توضیح نیست که چنین ترمیمی، مورد نظر این مقاله نیست و باید از لیست ترمیم‌های بالقوه حذف شود. چنین ترمیمی تحت عنوان "ترمیم خنثی" شناخته می‌شود و در سطر ۷ الگوریتم حذف می‌شود. چگونگی تشخیص این نوع ترمیم در بخش ۴-۲ این مقاله مورد بررسی قرار می‌گیرد.

٤-۲ معیار فاصله

تعیین نزدیک‌ترین ترمیم برای یک مدل ناسازگار بر اساس تعریف

معیار فاصله انجام می‌گیرد. با توجه به استفاده از بازنمایی گراف در این مقاله، فاصله بین دو مدل به صورت تعیین فاصله بین دو گراف متناظر قابل بیان است. با توجه به این که هدف اصلی مقایسه یک مدل و نسخه تغییر یافته آن است فاصله را می‌توان با توجه به تعداد نودها و یال‌های تغییر یافته (افزوده‌شده یا حذف‌شده) تعیین نمود.

در این مقاله از مفهوم فاصله ویرایش گراف[1] برای تخمین شباهت ساختاری دو مدل استفاده می‌شود. این معیار به عنوان هزینه تبدیل یک گراف به گراف دیگر با استفاده از عملگرهای پایه نظیر درج، حذف یا جایگزینی گره‌ها و یال‌های گراف بیان می‌شود. در این مقاله از (۱) برای تعیین فاصله بین دو گراف استفاده می‌شود. با در اختیار داشتن گراف مربوط به مدل m به صورت $G_m=(V_m,E_m)$ و گراف مرتبط با ترمیم r به صورت $G_r=(V_r,E_r)$ فاصله ویرایش دو گراف از طریق (۱) محاسبه می‌شود. در این رابطه S_v شامل مجموعه گره‌های اضافه‌شده/ حذف‌شده و S_e شامل مجموعه یال‌های اضافه‌شده/ حذف‌شده است (نماد $|\sigma|$ به معنای اندازه مجموعه σ است)

$$distance(m,r)=\frac{1}{2}\left(\frac{|S_v|}{|V_m|+|V_r|}+\frac{|S_e|}{|E_m|+|E_r|}\right) \qquad (1)$$

رابطه (۱) به گونه‌ای تعریف شده که تأثیر حذف گره/یال نسبت به افزودن آن بیشتر باشد. دلیل اصلی تأکید بر حفظ مدل طراحی‌شده توسط طراح تا حد امکان است. اضافه‌شدن یک گره به گراف ترمیم باعث افزایش هر دو عدد $|S_v|$ و $|V_r|$ می‌شود ولی حذف یک گره باعث افزایش $|S_v|$ و کاهش $|V_r|$ می‌گردد. این وضعیت در مورد افزودن و یا حذف یال‌ها نیز صدق می‌کند. به این ترتیب (۱) برای حذف گره/ یال از مدل اولیه امتیاز منفی در نظر می‌گیرد.

برای نمایش نحوه استفاده از (۱) در این قسمت به محاسبه فاصله بین مدل ناسازگار و ترمیم‌های مرتبط با آن در شکل ۱ می‌پردازیم. فاصله بین ترمیم اول (بخش ب از شکل ۱) با مدل اولیه (بخش الف از شکل ۱) طبق (۱) برابر ۰/۱۰۱ است و فاصله بین ترمیم دوم (بخش ج از شکل ۱) با مدل ناسازگار اولیه برابر ۰/۰۷۴ است. بنابراین ترمیم دوم (بخش ج از شکل ۱) به عنوان نزدیک‌ترین ترمیم انتخاب می‌شود.

همان طور که در بخش قبلی ذکر شد ممکن است در مسیر تعیین ترمیم‌های بالقوه یک ترمیم خنثی به دست آید که مسلماً مطلوب نیست. با توجه به این که یک ترمیم خنثی از طریق بی اثر کردن تغییرات اعمال‌شده توسط کاربر در مدل به دست می‌آید فاصله بین این ترمیم و مدل اولیه طبق (۱) برابر صفر خواهد بود. به این ترتیب امکان شناسایی و حذف ترمیم خنثی از مجموعه ترمیم‌های بالقوه فراهم می‌شود. لازم به ذکر است که تعریف معیار فاصله محدود به (۱) نیست و متناسب با شرایط مورد نظر امکان تعریف روابط مختلف با توجه به تغییرات گره‌ها و یال‌های گراف وجود دارد. همچنین امکان وزن‌دهی به هر دسته از تغییرات نیز میسر است.

۵- صحت و کمال روش پیشنهادی

در این بخش مسئله صحت و کمال[2] روش پیشنهادی با استفاده از تکنیک‌های صوری مورد بررسی قرار می‌گیرد. برای این منظور ابتدا تعاریف مورد نیاز ارائه خواهد شد. بازنمایی فضای حالت بر اساس مفهوم سیستم تبدیل گراف انجام می‌پذیرد. یک سیستم تبدیل در واقع یک گراف جهت‌دار است که در آن رئوس بیان‌کننده وضعیت‌ها و یال‌ها نشان‌دهنده انتقال بین وضعیت‌ها هستند.

تعریف ۱۰ (سیستم تبدیل گراف): یک سیستم تبدیل گراف GTS به صورت زوج (R,TG) است که در آن TG یک گراف نوع بوده و R شامل مجموعه قواعد تبدیل گراف مبتنی بر TG است. یک GTS در واقع یک سیستم تبدیل است که در آن رئوس خود گراف‌های نوع‌دار مبتنی بر TG بوده و تبدیل بین گراف‌ها (وضعیت‌ها) با استفاده از قواعد تبدیل گراف بیان می‌شوند.

مجموعه تمامی وضعیت‌های قابل دسترسی از وضعیت اولیه G_0 به عنوان فضای حالت یک GTS شناخته می‌شود. یک توالی تبدیل $G_0 \Rightarrow^* G_n$ در GTS به صورت $G_0 \Rightarrow G_1 \Rightarrow \ldots \Rightarrow G_n$ نمایش داده شده و شامل توالی از اعمال قواعد تبدیل گراف است که بر مبنای TG تعریف شده است. بر اساس تعریف یک سیستم تبدیل گراف، در این مقاله مفهوم "سیستم تبدیل گراف ترمیم" معرفی می‌شود.

تعریف ۱۱ (سیستم تبدیل گراف ترمیم): با در اختیار داشتن یک نمونه مدل ناسازگار m از توصیف فرامدل $M=(TG_M,C)$، یک سیستم تبدیل گراف ترمیم عبارت است از یک GTS که در آن مدل ناسازگار به عنوان وضعیت اولیه بوده و تبدیل وضعیت بر اساس اعمال قواعد تبدیل گراف (عمل‌های اصلاحی) انجام می‌پذیرد.

یک سیستم تبدیل گراف ترمیم که از این به بعد تحت عنوان RGTS مورد اشاره قرار می‌گیرد شامل وضعیت‌هایی است که هر یک نمونه مدل‌های منطبق بر فرامدل تعریف‌شده هستند. برخی از این وضعیت‌ها قواعد ناسازگاری تعریف شده در توصیف فرامدل را ارضا می‌کنند. آن دسته از وضعیت‌ها که قواعد ناسازگاری را ارضا نمی‌کنند در واقع ترمیم‌های بالقوه هستند. بنابراین مسئله مورد توجه این مقاله یافتن توالی بهینه از تبدیل‌های گراف در RGTS با در نظر گرفتن معیار فاصله است.

یک توالی تبدیل گراف $G \Rightarrow^* H$ خاتمه‌پذیر است اگر هیچ قاعده دیگری در GTS قابل اعمال به گراف H نباشد. یک GTS خاتمه‌پذیر است اگر همه توالی‌های تبدیل برای هر گراف اولیه G خاتمه‌پذیر باشند. مسئله خاتمه‌پذیری در یک سیستم تبدیل گراف از اهمیت بالایی برخوردار است. به طور کلی تصمیم‌گیری در این خصوص قابل انجام نیست [۹] اما در شرایط خاص و با فرضیات مشخص می‌توان خاتمه‌پذیری را تضمین نمود [۱۰].

هدف اصلی تضمین خاتمه توالی تبدیل گراف پس از اعمال چندین مرحله تبدیل بر روی گراف متناهی اولیه است. یک ایده کلی برای اثبات خاتمه‌پذیری در نظر گرفتن یک معیار برای خاتمه است. در مسئله این مقاله می‌توان تعداد ناسازگاری‌های موجود در مدل را به عنوان معیار خاتمه لحاظ نمود. بنابراین یک توالی تبدیل $m_0 \Rightarrow^* m_k$ در نهایت خاتمه می‌یابد اگر هیچ ناسازگاری در مدل m_k وجود نداشته باشد. به عبارت دیگر $\lim_{k\to\infty}|I_k|=0$ که در آن $|I_k|$ نشان‌دهنده تعداد ناسازگاری‌های موجود در مدل k است.

لم ۱ (خاتمه‌پذیری سیستم تبدیل گراف ترمیم): سیستم تبدیل گراف ترمیم متناظر با الگوریتم پیشنهادی این مقاله (شکل ۵) برای هر مدل متناهی و معتبر ورودی خاتمه‌پذیر است.

اثبات: برای اثبات باید نشان دهیم در سیستم تبدیل گراف ترمیم با وضعیت اولیه m_0 همه توالی‌های تبدیل به صورت $m_0 \Rightarrow^* m_p$ خاتمه‌پذیر هستند. با توجه به تعریف RGTS در هر وضعیت متناسب با ناسازگاری‌های موجود تنها امکان اعمال قواعد تبدیل گراف مرتبط با

1. Graph Edit Distance
2. Soundness and Completeness

جدول ۱: لیست برخی قواعد ناسازگاری برای مدل BPMN به همراه قواعد ترمیم مرتبط.

قواعد ناسازگاری	عمل‌های اصلاحی ممکن
در صورت بازشدن مسیرها توسط XOR-gate و بسته‌شدن آنها توسط parallel gate امکان بروز بن‌بست وجود دارد.	جایگزینی parallel gate با یک XOR-gate حذف هر دو gateway و افزودن یک رابط جایگزینی XOR-gate با یک parallel gate
بین دو task در یک pool مشابه امکان وجود اتصال پیام وجود ندارد.	حذف اتصال پیام جایگزینی اتصال پیام با یک اتصال معمولی
یک XOR-gate باید حتماً حداقل یک مسیر پیش‌فرض داشته باشد. یک startevent نمی‌تواند رابط ورودی داشته باشد.	انتخاب یکی از مسیرهای خروجی gate به عنوان پیش‌فرض حذف رابط ورودی به startevent

عمل‌های اصلاحی متناظر وجود دارد. دو حالت قابل تصور است: حالت اول، ناسازگاری‌های موجود در m_p یک وضعیت نقض تکراری ایجاد می‌کند که در این صورت توالی تبدیل خاتمه می‌یابد. حالت دوم، اگر m_p ناسازگار باشد و وضعیت‌های نقض متناظر تکراری نباشد همواره یک توالی تبدیل به صورت $m_p \Rightarrow^* m_q$ وجود دارد که $\lim_{q\to\infty} |I_q| = 0$ و در نتیجه $m_0 \Rightarrow^* m_q$ خاتمه‌پذیر است.

در ادامه صحت روش پیشنهادی در این مقاله جهت رفع ناسازگاری مورد بررسی قرار می‌گیرد.

قضیه ۱ (صحت): با در اختیار داشتن یک مدل ناسازگار m که یک نمونه ساختاری از توصیف فرامدل $M=(TG_M, C)$ است الگوریتم پیشنهادی همواره یک ترمیم (در صورت وجود) را فراهم می‌کند.

اثبات: در روش پیشنهادی، مدل ناسازگار به عنوان وضعیت اولیه در سیستم تبدیل گراف ترمیم در نظر گرفته می‌شود. الگوریتم پیشنهادی متناسب با ناسازگاری‌های موجود در مدل اولیه، قواعد تبدیل گراف متناظر را اعمال می‌کند و به این ترتیب مدل اولیه به مدل جدید تبدیل می‌شود $m_0 \Rightarrow m_1$. در صورتی که مدل جدید همچنان ناسازگار باشد فرایند رفع ناسازگاری با اعمال قواعد تبدیل گراف دیگر ادامه می‌یابد. با توجه به اثبات خاتمه‌پذیر بودن RGTS (لم ۱) در نهایت توالی تبدیل گراف به صورت $m_0 \Rightarrow^* m_n$ وجود خواهد داشت که در آن m_n یک ترمیم است.

در ادامه باید مسئله کمال برای روش پیشنهادی بررسی شود، به این معنا که روش پیشنهادی برای همه نمونه مدل‌های ممکن بر اساس تعریف ۵ این مقاله قابل اعمال است.

قضیه ۲ (کمال): الگوریتم پیشنهادی برای همه نمونه‌های ساختاری از توصیف فرامدل به صورت $M=(TG_M, C)$ معتبر است.

اثبات: مراحل اثبات این قضیه ساده است. بر اساس تعاریف ۴ و ۵ هر نمونه ساختاری از $M=(TG_M, C)$ قابل بازنمایی به وسیله گراف جهت‌دار است. الگوریتم پیشنهادی هیچ فرض محدودکننده‌ای برای مدل ورودی به جز مطابقت با توصیف فرامدل مربوط در نظر نمی‌گیرد و بنابراین مسئله کمال روش پیشنهادی برقرار است.

۶- ارزیابی

روش پیشنهادی در این مقاله در مطالعه موردی بر اساس مدل‌های BPMN مورد ارزیابی قرار گرفته است. در این بخش به توضیح مطالعه موردی و بیان نتایج آزمایشات می‌پردازیم.

۶-۱ پیاده‌سازی

پیاده‌سازی روش ارائه‌شده بر مبنای چارچوب VIATRA۲ انجام گرفته است. این چارچوب مبتنی بر گراف بوده و امکانات یک‌پارچه‌ای جهت تبدیل مدل بر اساس تبدیل گراف ارائه می‌کند. این چارچوب همچنین امکاناتی برای یافتن الگو در گراف به صورت افزایشی دارد که باعث بهبود کارایی می‌شود. مزیت دیگر چارچوب استفاده‌شده وجود قابلیت اعمال مسایل بهینه‌سازی به طور مستقیم بر روی مدل است. برای این منظور از امکانات چارچوب DSE مبتنی بر VIATRA۲ بهره گرفته شده است [۶]. این چارچوب امکانات لازم برای تعریف فضای جستجو و اعمال استراتژی جستجوی مورد نظر را در اختیار می‌گذارد.

امکان دیگر چارچوب مورد اشاره که در پیاده‌سازی تأثیر بسیاری دارد وجود قابلیت تراکنش در سطح تبدیل مدل است که امکان بی‌اثر کردن نتایج اعمال یک قاعده تبدیل را میسر می‌سازد. این قابلیت برای بازگشت به وضعیت قبلی در پیاده‌سازی الگوریتم به کار رفته است. هر تراکنش در این چارچوب دارای یک شماره شناسه یکتا است که مورد استفاده قرار می‌گیرد. به منظور مقایسه بین یک مدل و ترمیم آن یک مقایسه‌گر توسعه یافته که بر مبنای معیار تعریف‌شده به محاسبه فاصله بین مدل و ترمیم آن می‌پردازد.

۶-۲ مطالعه موردی

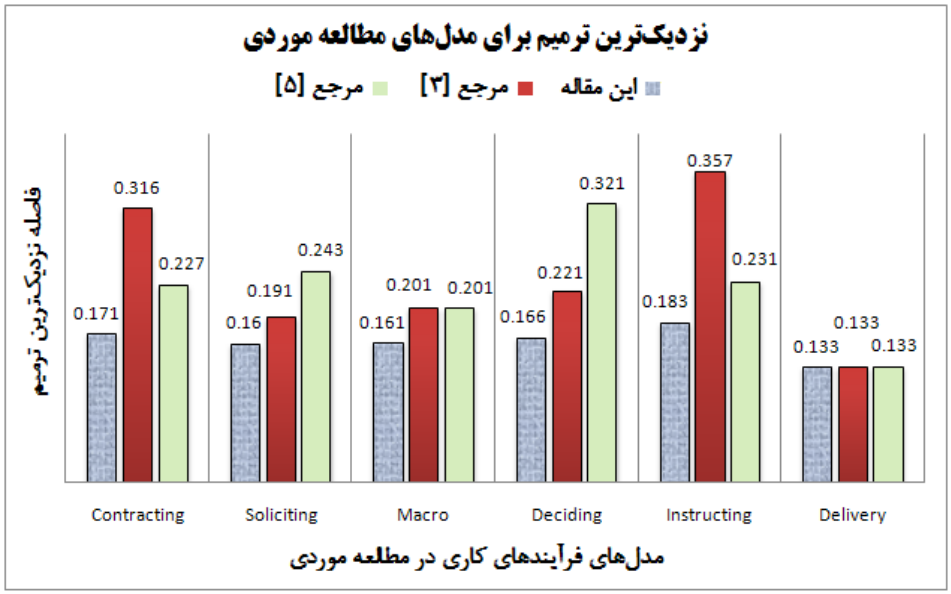
در سال‌های اخیر استفاده از زبان مدل‌سازی BPMN برای توصیف فرایندهای کاری گسترش زیادی یافته است. در این مقاله روش پیشنهادی به مجموعه‌ای از مدل‌های مبتنی بر BPMN اعمال شده است. مجموعه مدل‌های مورد استفاده در این مقاله بر اساس مدل‌های مورد استفاده در [۴] است که شامل تعریف قواعد ناسازگاری و همچنین قواعد اصلاحی است[۱]. جدول ۱ به برخی از نمونه قواعد ناسازگاری و همچنین قواعد اصلاحی مرتبط اشاره دارد.

روش ارائه‌شده در این مقاله بر روی مجموعه مدل‌های فرایند جدول ۲ اعمال گردید. در این جدول مشخصات هر مدل شامل تعداد رئوس و یال‌های گراف معادل و نیز تعداد ناسازگاری‌های مدل ارائه شده است.

۶-۳ نتایج و بحث

نتایج اعمال روش پیشنهادی این مقاله به مدل‌های مورد استفاده در مطالعه موردی در جدول ۲ ارائه شده است. برای هر مدل BPMN تعداد گره‌ها، یال‌ها و همچنین تعداد ناسازگاری‌ها نمایش داده شده است. همچنین فاصله نزدیک‌ترین ترمیم هم در جدول وجود دارد. شکل ۶ مقایسه بین نتایج روش پیشنهادی این مقاله با دو کار مشابه قبلی را نمایش می‌دهد. بر اساس شکل مشخص است که روش این مقاله نتایج بهتری ارائه می‌کند که به دلیل تأکید بر یافتن نزدیک‌ترین ترمیم است.

۱. دسترسی به مجموعه مدل‌های مورد استفاده در مطالعه موردی به همراه تعریف قواعد تشخیص و رفع ناسازگاری و همچنین کدهای پیاده‌سازی در مسیر زیر امکان‌پذیر است:
http://ceit.aut.ac.ir/islab/researches/modeling/distance.rar



شکل ۶: مقایسه بین نتایج این مقاله و نتایج دو کار مشابه.

جدول ۲: نتایج اعمال روش پیشنهادی به مدل‌های مورد استفاده در مطالعه موردی.

عنوان مدل فرایند	تعداد رئوس گراف	تعداد یال‌های گراف	تعداد ناسازگاری‌ها در مدل	فاصله نزدیک‌ترین ترمیم نسبت به مدل اولیه
Contracting	۲۹	۳۸	۱۲	۰٫۱۷۱
Soliciting	۱۷	۲۲	۶	۰٫۱۶۰
Macro	۱۲	۱۴	۴	۰٫۱۶۱
Deciding	۹	۱۰	۳	۰٫۱۶۶
Instructing	۳۲	۴۴	۱۴	۰٫۱۸۳
Delivery	۷	۷	۲	۰٫۱۳۳

روش پیشنهادی در این مقاله امکان تشخیص و رفع ناسازگاری در یک مدل و یافتن نزدیک‌ترین ترمیم را به صورت خودکار فراهم می‌کند. برای این منظور لازم است قواعد ناسازگاری و نیز قواعد اصلاحی مرتبط با هر نوع ناسازگاری تعریف شود.

یافتن نزدیک‌ترین ترمیم و رفع ناسازگاری مدل در روش پیشنهادی مقاله نیازمند تعیین ترمیم‌های بالقوه بر اساس اعمال قواعد اصلاحی بر مدل ناسازگار است. اندازه مدل ورودی و نیز تعداد ناسازگاری‌های مدل و تعداد قواعد اصلاحی تعریف‌شده برای هر نوع ناسازگاری دو عامل تأثیرگذار بر کارایی و مقیاس‌پذیری راهکار پیشنهادی هستند. با توجه به دامنه کاربرد راهکار پیشنهادی در حوزه مهندسی نرم‌افزار در خصوص مسئله مقیاس‌پذیری می‌توان به این نکات توجه داشت: اولاً در حوزه مهندسی نرم‌افزار تأکید بر استفاده از مدل‌های مختلف جهت مدل‌سازی یک سیستم نرم‌افزاری از نماهای[1] مختلف است و با توجه به تأکید بر تجزیه سیستم به مؤلفه‌های با کمترین وابستگی[2]، مدل‌های بزرگ قابل تجزیه به مدل‌های کوچک‌تر هستند. ثانیاً اگر تعداد ناسازگاری‌های یک مدل از یک حد مشخص بیشتر باشد عملاً ارزش چندانی برای ترمیم نخواهد داشت و بازطراحی مدل راهکار مناسب‌تری خواهد بود.

مسئله دیگر مرتبط با موضوع مقیاس‌پذیری بحث کارایی روش انطباق الگوی گراف است که می‌تواند به صورت بالقوه گلوگاه کارایی روش پیشنهادی باشد. برای رفع این مسئله از قابلیت‌های چارچوب VIATRA۲ بهره گرفته شده است. این چارچوب با استفاده از روش انطباق الگو به صورت افزایشی باعث بهبود کارایی در این زمینه می‌شود [۱۱].

۷- کارهای مرتبط

مسئله تشخیص ناسازگاری مدل در کارهای تحقیقاتی متعددی مورد توجه قرار گرفته است. در [۱۲] یک چارچوب برای تشخیص سریع ناسازگاری‌ها پیشنهاد شده که با تعریف محدوده برای هر یک از قواعد سازگاری امکان تشخیص سریع ناسازگاری‌ها در صورت تغییر مدل را می‌دهد. در [۱۳] تشخیص ناسازگاری‌ها از طریق بازنمایی مدل و فرامدل در قالب منطق توصیفی و استفاده از پرس و جو در این زبان انجام می‌شود. در [۴] استفاده از الگوی گراف برای توصیف و تشخیص ناسازگاری‌ها در یک مدل پیشنهاد شده است. کاربرد پرس و جو مبتنی بر پرولوگ[3] در [۵] برای تشخیص ناسازگاری مورد بحث قرار می‌گیرد.

کارهای تحقیقاتی در زمینه رفع ناسازگاری مدل با رویکردهای مختلفی به ارائه راهکار می‌پردازند. در [۱۴] روشی برای رفع ناسازگاری در مدل‌های UML ارائه شده است. در این روش گزینه‌های مناسب جهت رفع یک ناسازگاری از طریق یک تابع مولد ایجاد می‌شوند. استفاده از تبدیل گراف برای رفع ناسازگاری مدل در [۱۵] و [۱۶] گزارش شده است. در [۳] روشی مبتنی بر منطق توصیفی جهت رفع ناسازگاری پیشنهاد شده است. در [۱۷] روشی برای تعریف اصلاح محدوده برای رفع نقض قیود در پیکربندی نرم‌افزار ارائه شده است. این روش بر مبنای تعیین محدوده مقادیر مجاز برای تغییرات اجرا می‌شود.

زمینه تحقیقاتی مرتبط دیگر در این خصوص در فضای پایگاه داده

1. View
2. Coupling
3. Prolog

است. مسئله تشخیص و رفع ناسازگاری در این فضا هم به طور جدی مورد توجه بوده است [۱۸]. مسئله ترمیم قیود با هدف پیداکردن یک ترمیم بهینه برای تبدیل یک پایگاه داده ناسازگار به وضعیت سازگار در این فضای تحقیقاتی مطرح است ([۱۹] و [۲۰]).

روش پیشنهادی این مقاله رویکرد جدیدی در مقایسه با کارهای مشابه در بحث تشخیص و رفع ناسازگاری مدل ارائه می‌کند. در رویکرد پیشنهادی نتایج حاصل از اعمال روش رفع ناسازگاری از نظر میزان شباهت به مدل ناسازگار اولیه بر اساس معیار فاصله مقایسه می‌شوند و نزدیک‌ترین ترمیم برای کاربر ارائه می‌شود. تعریف معیار فاصله متناسب با شرایط مسئله و با در نظر گرفتن فاصله ویرایش بین دو گراف انجام می‌پذیرد.

۸- نتیجه‌گیری

در این مقاله روشی برای تشخیص و رفع ناسازگاری در یک مدل ناسازگار ارائه شده است. مهم‌ترین تفاوت این کار با کارهای مشابه تأکید بر تعریف و استفاده از معیار فاصله برای یافتن نزدیک‌ترین ترمیم برای مدل ناسازگار است. در روش پیشنهادی با انجام جستجو در بین ترمیم‌های ممکن و از طریق مقایسه بر مبنای معیار فاصله امکان تعیین نزدیک‌ترین ترمیم فراهم می‌شود. به این ترتیب ناسازگاری‌های موجود در مدل اولیه با کمترین تغییرات برطرف می‌شود.

در این مقاله از گراف جهت‌دار برای بازنمایی مدل و فرامدل استفاده شده است. تعریف ناسازگاری در قالب الگوی گراف انجام پذیرفته و مسئله تشخیص ناسازگاری به صورت مسئله یافتن یک الگو در گراف در نظر گرفته می‌شود. همچنین رفع ناسازگاری با استفاده از تبدیل گراف اولیه به گراف مطلوب از طریق اعمال قواعد تبدیل گراف صورت می‌پذیرد.

در کارهای پژوهشی آینده گسترش این روش به ساختارهای چندمدله مورد توجه است. یک ساختار چندمدله از مجموعه چندین مدل مرتبط به هم تشکیل می‌شود. مفهوم ناسازگاری در این ساختارها پیچیدگی بیشتر داشته و شامل ناسازگاری‌های محلی و سراسری است.

مراجع

[1] J. Bezivin, "In search of a basic principle for model driven engineering," *Special Novatica Issue UML and Model Engineering*, vol. 2, no. 2, pp. 21-24, 21-24. 2004.

[2] G. Spanoudakis and A. Zisman, "Inconsistency management in software engineering: Survey and open research issues," In S. K. Chang, ed., *Handbook of Software Engineering and Knowledge Engeering*, pp. 24-29, 2001.

[3] P. J. Puissant, T. Mens, and R. Straeten, "Resolving model inconsistencies with automated planning," in *Proc. of the 3rd Workshop on Living with Inconsistencies in Software Development, CEUR Workshop Proc.*, pp. 8-14, 2010.

[4] A. Hegedus, A. Horvath, I. Rath, M. C. Branco, and D. Varro, "Quick fix generation for DSMLs," in *Proc. IEEE Symp. on Visual Languages and Human-Centric Computing, VL/HCC,* pp. 17-24, Pittsbourgh, PA, USA, 18-22 Sep. 2011.

[5] M. Silva, A. Mougenot, X. Blanc, and R. Bendraou, "Towards automated inconsistency handling in design models," in *Proc. CAiSE 2010*, vol. LNCS 6051, pp. 348-362, 2010.

[6] A. Horvath and D. Varro, "CSP(M): constraint satisfaction programming over models," in *Proc. of ACM/IEEE 12th Int. Conf. on Model Driven Engineering Languages and Systems, Models'09,* vol. LNCS 5795, pp. 107-121, Oct. 2009.

[7] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer, "Fundamental theory for typed attributed graphs and graph transformation based on adhesive HLR categories," *Fundamenta Informaticae*, vol. 74, no. 1, pp. 31-61, 2006.

[8] A. Kleppe, Software Language Engineering: Creating Domain-Specific Languages Using Metamodels, Addison-Wesley, 2008.

[9] D. Plump, "Termination of graph rewriting is undecideable," *Fundamental Informatica*, vol. 33, no. 2, pp. 201-209, 2008.

[10] D. Varro, S. Varro-Gyapay, H. Ehrig, U. Prange, and G. Taentzer, "Termination analysis of model transformationsby petri nets," in *Proc. ICGT06*, vol. LNCS 4178, pp. 260-274, 2006.

[11] G. Bergmann, A. Okros, I. Rath, D. Varro, and G. Varro, "Incremental pattern matching in the viatra model transformation system," in *Proc. of the 3rd Int. Workshop on Graph and Model Transformations*, pp. 25-32, May 2008.

[12] A. Egyed, "Instant consistency checking for UML," in *Proc. Int'l Conf. Software Engineering, ICSE'06,* pp. 381-390, May 2006.

[13] R. V. D. Straeten, Inconsistency Management in Model Driven Engineering an Approach Using Description Logics, Ph. D Thesis, Vrije Universiteit Brussel, Brussels, Belgium, 2005.

[14] A. Egyed, E. Letier, and A. Finkelstein, "Generating and evaluating choices for fixing inconsistencies in UML design models," in *Proc. 23rd IEEE/ACM Int. Conf. on Automated Software Engineering*, pp. 99-108, 2008.

[15] C. Amelunxen, E. Legros, A. Schurr, and I. Sturmer, "Checking and enforcement of modeling guidelines with graph transformations," in *Applications of Graph Transformations with Industrial Relevance,* vol. 5088, Lecture Notes in Computer Science, Springer, pp. 313-328, 2008.

[16] E. Guerra and J. Lara, "Event-driven grammars: towards the integration of meta-modelling and graph transformation," in *Graph Transformations,* vol. 3256, Lecture Notes in Computer Science, Springer, pp. 215-218, 2004.

[17] Y. Xiong, A. Hubaux, S. She, and K. Czarnecki, "Generating range fixes for software configuration," in *Proc. 34th Int. Conf. on Software Engineering, ICSE'12*, pp. 58-68, Zurich, Switzerland, 2-9 Jun. 2012.

[18] J. Wijsen, "Database repairing using updates," *ACM Trans. on Database Systems*, vol. 30, no. 3, pp. 722-768, Sep. 2005.

[19] F. N. Afrati and P. G. Kolaitis, "Repair checking in inconsistent databases: algorithms and complexity," in *Proc. of 12th Int. Conf. on Database Theory*, pp. 31-41, Saint-Petersburg, Russia, 23-26 Mar. 2009.

[20] E. Santos and J. P. Martins, "An Argumentation framework for optimal repair checking," in *Proc IEEE 5th Int. Conf. on. Intelligent Computer Communication and Processing, ICCP'09,* pp. 19-26, 27-29 Aug. 2009.

رضا گرگان محمدی در حال حاضر دانشجوی دکتری نرم‌افزار در دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر است. نامبرده مقطع کارشناسی ارشد و کارشناسی خود را به ترتیب در سال‌های ۱۳۸۷ و ۱۳۸۵ در رشته مهندسی کامپیوتر گرایش نرم‌افزار همان دانشگاه به پایان رسانده است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: مهندسی نرم افزار مبتنی بر مدل، مدل‌های قابل اجرا، زبان های مدل‌سازی خاص دامنه، روش‌های صوری در مهندسی نرم افزار، مهندسی نیازمندی‌ها.

احمد عبداله‌زاده بارفروش در سال ۱۳۵۴ مدرک کارشناسی خود در رشته حسابداری (مهاد ریاضی) را از دانشگاه تهران دریافت کرد و مدرک کارشناسی ارشد خود را در سال ۱۳۵۹ در رشته مهندسی کامپیوتر با گرایش برنامه‌نویسی برنامه‌های کاربردی از دانشگاه وست‌کاست در لس‌آنجلس واقع در آمریکا دریافت نمود. ایشان در سال ۱۳۶۹ مدرک دکتری خود را در رشته سیستم‌های هوشمند از دانشگاه بریستول انگلستان اخذ نمود و در همان سال عضو هیأت علمی دانشگاه صنعتی امیرکبیر شد. وی در بیست سال گذشته به عنوان استاد مدعو در دانشگاه های مریلند (آمریکا)، اوراسی (فرانسه)، ترنتو (ایتالیا)، و لوفبرو (انگلستان) حضور داشته است. ایشان مولف کتاب‌های «مقدمه ای بر هوش مصنوعی توزیع شده» و «متدلوژی تضمین کیفیت نرم- افزار» است. زمینه‌های تخصصی ایشان عبارتند از: هوش مصنوعی، سیستم های مبتنی بر عامل، سیستم‌های خبره، پردازش زبان طبیعی، سیستم‌های تصمیم یار، هوش تجاری، داده کاوی، و مهندسی نرم‌افزار.