چارچوبی صوری برای تغییر پیکربندی در سیستم‌های تطبیقی

جابر کریم‌پور و رباب علیاری

چکیده: امروزه از سیستم‌های پیشرفته انتظار می‌رود که بتوانند خود را با شرایط محیطی و اتفاقات غیر منتظره سازگار کنند. اولین شرط برای چنین سیستم‌هایی سازگارشدن آنها با توجه به نیازهای مشتری، داشته‌های خود و محیطی که در آن قرار دارند است و باید هنگام مواجه با مشکل یا درخواست غیر منتظره به آن جواب دهند. در راهکارهای تطابق‌سازی سعی شده با استفاده از قرارداد تطابق و تغییر پیکربندی مجدد، سیستم با شرایط وفق داده شود به طوری که این تغییر پیکربندی مجدد به دور از چشم سرویس‌گیرنده و گاهی در حین اجرای عملیات انجام گردد و طراحان سیستم را نیز از دخالت مستقیم در امور داخلی مربوط به کارگزار منع کند. شایان ذکر است گاهی اوقات همین روش‌های تطابق‌سازی نقش زیادی در استفاده مجدد مؤلفه‌ها برای ایجاد سیستم‌های جدید یا بالابردن سطح سیستم‌های قدیمی با استفاده از مؤلفه‌های موجود را نیز ایفا می‌کنند. این مقاله با روش‌های فرمال سعی در ایجاد سیستمی دارد که توان سازگاری با محیط را داشته باشد و در کنار آن، مشکلات ناشی از پیچیدگی را نیز تا حدی می‌کاهد. برای این کار ابتدا سیستم به وسیله یک مدل ریاضی نمایش داده شده و سپس از روی سیستم مدل‌شده، قرارداد تطابق و درخواست مشتری یک فرم جدید به نام آداپتور طراحی می‌شود. بعد از ایجاد آداپتور تمام پیکربندی‌ها و تراکنش‌های بین سیستم و مشتری از طریق آن انجام می‌گردد و آداپتور مسئول هماهنگ‌سازی اجزای داخلی سیستم می‌باشد. همچنین برای جلوگیری از پیچیدگی، از مفهوم شبکه و سرویس‌های سلسله مراتبی برای ایجاد شبکه‌ای از آداپتورها استفاده شده است.

کلید واژه: **آداپتور، پیکربندی پویا، سیستم‌های تطبیقی، روش‌های صوری.**

۱- مقدمه

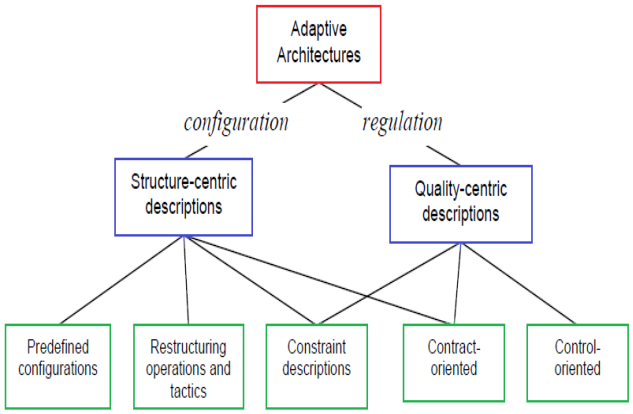
امروزه، اکثر سیستم‌های کامپیوتری از مؤلفه‌های گوناگون با رفتارهای غیر مطمئن ساخته می‌شوند. مؤلفه‌ها و همچنین رفتار آنها ممکن است با گذر زمان تغییر کند و بنابراین می‌توان گفت اگر سیستم‌های مؤلفه‌گرا توانایی غلبه بر اتفاقات غیر قابل پیش‌بینی شده و تغییرات ناگهانی را داشته باشند، می‌توانند در توسعه سیستم‌های نرم‌افزاری مفید واقع شوند. ولی این مقابله با تغییر که از آن سخن گفتیم به چه معناست؟

در اکثر سیستم‌های واقعی تغییرات و مقابله با تغییرات وجود دارد. به عنوان مثال در بافت‌های بدن موجودات، تغییرات ساختاری جهت تعامل با محیط دیده می‌شود. این تغییرات ساختاری می‌توانند به دو نوع تغییرات درونی اجزای بافت و تغییرات روابط اجزا تقسیم‌بندی شوند. جایگزینی سلول‌های مرده به وسیله سلول‌های زنده مثالی از تغییرات درونی اجزا و تغییرات ساختار سیستم عصبی نیز مثالی از تغییرات روابط اجزا است، زیرا سیستم عصبی کار بازسازی سلول‌ها را بر عهده دارد.

این مقاله در تاریخ ۲ اردیبهشت ماه ۱۳۹۲ دریافت و در تاریخ ۱ آبان ماه ۱۳۹۳ بازنگری شد.

جابر کریم‌پور، استادیار، گروه علوم کامپیوتر، دانشگاه تبریز، تبریز، (email: karimpour@tabrizu.ac.ir).

رباب علیاری، دانشجوی دکتری، گروه علوم کامپیوتر، دانشگاه تبریز، تبریز، (email: robab.alyari@gmail.com).



شکل ۱: دسته‌بندی چارچوب‌های تطبیقی [۱].

در سیستم‌های نرم‌افزاری نیز مسأله تغییرات و مقابله با آنها، یک مسأله انکارناپذیر است. همانند بافت‌های زنده در این سیستم‌ها نیز دو نوع تغییرات: الف) تغییر مؤلفه‌های سیستم و ب) تغییر روابط بین مؤلفه‌ها وجود دارد. با این دیدگاه، سیستم‌های نرم‌افزاری تطبیقی به وجود آمدند.

۱-۱ تعریف مسأله

سیستم‌های بزرگ امروزی معمولاً از چندین قسمت مختلف تشکیل می‌شوند. در مواردی ملاحظه می‌شود که سیستم کارایی لازم برای پاسخ به درخواست‌های مشتری را ندارد و یا در قسمتی از آن مشکلی پیش آمده است که قادر به ادامه حیات نیست. برای حل چنین مشکلاتی از تغییر پیکربندی استفاده می‌شود. مسأله اصلی که در این مقاله به آن پرداخته شده است، چگونگی تغییر پیکربندی در سیستم‌های تطبیقی است. برای نیل به این هدف ابتدا باید سیستمی تعریف گردد که قوی و پایدار بوده و توان ارائه رفتار تطبیقی را داشته باشد. همچنین لازم است سیستم به وسیله ابزارهای صوری مدل‌سازی شود تا کمترین خطا و حداکثر بازده را داشته باشد.

۱-۲ راهکارهای پیشین

در زمینه سیستم‌های تطبیقی چارچوب‌های بسیاری در این دو دهه اخیر معرفی گردیده است، از جمله سیستم‌های خودمدیر، سیستم‌های توکار و سیستم‌های آگاه از مفهوم. آلن کولمن [۱] در سال ۲۰۰۷ در رساله دکتری خود معماری سیستم‌های تطبیقی را به دو دسته تقسیم کرد. همان طور که در شکل ۱ مشاهده می‌شود، دسته اول با ساختار سیستم و اطمینان از خوش‌فرم بودن آن سر و کار دارد که مربوط به مدیریت پیکربندی‌ها می‌باشد. توجه دسته دوم روی اندازه‌گیری کیفیت ارتباط در یک ساختار است. کنترل کیفیت ارتباطات هدف اصلی فعالیت‌های مدیریتی این دسته می‌باشد و این توصیفات می‌توانند با توجه به ساز و کار آنها در رسیدن به پیکربندی و تنظیم مجدد، به زیردسته‌هایی توسعه یابند.

در دسته اول، توصیفات ساختار- مرکزی می‌توانند با توجه به توصیف پیکربندی‌های مثبت (توصیف آنچه هست) و منفی (توصیف آنچه نمی‌تواند باشد) به صورت: ۱) آنهایی که پیکربندی‌های مجاز تعریف

می‌کنند همانند پلاستیک [۲] و آرک جاوا [۳]، ۲) آنهایی که راهکارها و قوانین تغییر پیکربندی را تعریف می‌کنند از جمله چارچوب فورسید [۴] و ۳) آنهایی که از تعریف قیود استفاده می‌کنند (چارچوب‌های داروینی [۵] از این دسته‌اند)، مجزا شوند.

در دسته دوم، توصیفات کیفیت- مرکزی درگیر تغییرات قابل اجرا در ساختار هستند. همان گونه که در شکل ۱ نمایش داده شده است، می‌توان یک جداسازی بین راهکارهای قراردادگرا و کنترل‌گرا در توصیفات کیفیت- مرکزی قایل شد. راهکارهای کنترل‌گرا، اجزای سیستم را با نظارت بر تغییرات به وسیله متغیرهای کنترلی و سپس اجرای تغییرات با استفاده از متغیرهای پردازشی، تنظیم می‌کنند. در طرف دیگر، راهکارهای قراردادگرا، ارتباطات بین اجزا را به وسیله تعریف ارتباط مجاز و سطوح انجام آن ارتباطات، هماهنگ می‌کنند. بقیه راهکارها سطوح قابل اجرای تغییرات را تعریف کرده و شامل قسمتی برای بازبینی ساز و کار جهت حصول اطمینان از حتمی‌بودن اجرای آنها هستند و در صورت هر اتفاق غیر منتظره عکس‌العمل نشان می‌دهند. این عمل در سیستم‌های کنترلی، شامل کنترل تغییرات در ویژگی‌های اجزا می‌باشد. در این گونه سیستم‌ها اجزا باید به صورت پویا نیازمندی‌ها را برآورده سازند. البته برای تعریف روابط ساختاری در یک سیستم از قراردادها نیز استفاده می‌شود زیرا قراردادها در نقش مرتبط‌کننده، تعریف‌گر روابط بین مؤلفه‌ها هستند. در نتیجه قراردادها یا مرتبط‌ها می‌توانند برای توصیف هر گونه ساختار و کیفیت روابط مورد استفاده قرار گیرند. بنابراین توصیفات قراردادگرا به عنوان زیربخشی از هر دوی توصیفات کنترل- مرکزی و ساختار- مرکزی در نظر گرفته می‌شود.

از جمله راهکارهای کنترل‌گرا می‌توان به رین بو [۶] و آئورا [۷] اشاره کرد. آنها بر روی نظارت و کنترل مؤلفه‌ها به وسیله متغیرهای کنترلی تأکید می‌کنند. در نقطه مقابل، چارچوب‌های قراردادگرا، کنترل را به وسیله ویژگی‌های روابط مابین مؤلفه‌ها انجام می‌دهند. در بعضی چارچوب‌ها قراردادها مخلوطی از پیکربندی‌های مورد قبول ترکیب‌ها هستند و بعضی دیگر، قراردادها را برای کنترل عملیات رخ داده مابین مؤلفه‌ها به کار می‌برند. بعضی چارچوب‌ها هنگامی که دستخوش تغییراتی می‌شوند، تغییر پیکربندی داده و خود را دوباره تنظیم می‌کنند.

تغییر پیکربندی با توجه به اشتراک کاری در زمینه استفاده مجدد از مؤلفه‌ها و همچنین خود سیستم‌های تطبیقی، اساس چارچوب ارائه‌شده در این مقاله قرار گرفته است. در ادامه، چهار راهکار متشابه با چارچوب پیشنهادی این مقاله به تفصیل بررسی می‌گردند.

از جمله چارچوب‌های ارائه‌شده مشهور می‌توان به PobSAM[1] اشاره کرد که یک چارچوب تطبیقی کنترل‌گرا است. خاکپور و همکارانش [۸] در این مدل از ترکیب مجموعه‌ای از مدیرهای خودمختار و المان‌های مدیریت‌شده استفاده کرده‌اند. مدیرهای خودمختار، عناصر مدیریت‌پذیری هستند که مسئول بازبینی و انجام کارهای مربوط به رخدادها به وسیله روش‌های مناسب می‌باشند. هر مدیر مجموعه‌ای از پیکربندی‌هایی را دارد که شامل قوانین خاصی از تطابق هستند. همچنین مدیرها پیکربندی‌های خود را به صورت پویا برای جواب به تغییرات ایجادشده توسط محیط تعویض می‌نمایند. روبرتو برونی و همکارانش [۹] چارچوب کنترلی دیگری برای تطابق ارائه نموده‌اند. نکته جالب چارچوب برونی استفاده از یک زبان صوری (LTS) برای پایه ساختارش می‌باشد. آنها بر این عقیده‌اند که برای ایجاد تطابق باید رفتار یک مؤلفه را وابسته به داده‌های کنترلی خوش‌تعریف نمود. در این چارچوب تطابق به این معناست که بتوان در حین اجرای سیستم، یک داده کنترلی را به داده‌ای دیگر تبدیل کرد. یک مؤلفه هنگامی تطبیق‌پذیر است که مجموعه‌ای از داده‌های کنترلی را در حین اجرا داشته باشد.

کان‌سادو و همکارانش [۱۰] در سال ۲۰۱۱ چارچوبی را برای تغییر پیکربندی ساختاری مؤلفه‌ها تحت تطابق رفتاری با استفاده از قراردادها ارائه کردند. رفتار تطبیقی مد نظر در چارچوب کان‌سادو شامل اضافه و حذف مؤلفه‌ها در حین اجرا یا تعویض و جایگزینی آنها می‌باشد. همچنین فرض شده مؤلفه‌ها در این چارچوب، دارای رفتار قابل قبول جهت ارائه توانمندی‌های تغییر پیکربندی می‌باشند. کار اصلی این چارچوب بر پایه قرارداد تطابق بنا شده و این قرارداد به وسیله بردارهای LTS[2] تعریف شده و سپس آداپتورها را ایجاد می‌کنند. با ایجاد آداپتور، رفتار متقابل دو مؤلفه با هم وفق داده می‌شود و این کار به وسیله عملیات ارسال و دریافتی رخدادهای مابین مؤلفه‌ها انجام می‌پذیرد.

از ویژگی‌های این راهکار می‌توان به استفاده از یک زبان صوری اشاره کرد. همچنین استفاده از قرارداد تطابق باعث شده تا هیچ ناسازگاری بین واسط‌های مؤلفه‌های گوناگون رخ ندهد. دیگر ویژگی مهم این راهکار، پشتیبانی از تغییر پیکربندی در حین اجرا می‌باشد. از نقات ضعف این راهکار نیز می‌توان به پیچیدگی بیش از حد آن در ایجاد شبکه‌های تغییر پیکربندی اشاره کرد به شکلی که برای اجرای یک تغییر پیکربندی و رسیدن به مؤلفه مورد نظر، نیاز به پیمایش درخت شبکه، از برگ تا ریشه و سپس دوباره از ریشه تا برگ مورد نظر است. همچنین در مورد مقیاس‌پذیری این راهکار نیز سخنی گفته نشده، یعنی مشخص نگردیده که اگر مؤلفه‌ای با واسط کاربری ناسازگار وارد شبکه شود، چگونه این مؤلفه و آداپتور مرتبط با آن مدیریت می‌شود.

کامارا و همکارانش [۱۱] در سال ۲۰۱۲ راهکاری تعاملی برای طراحی قراردادها ارائه نمودند. طراحان برای ایجاد این راهکار در ابتدا واسط کاربری مناسبی بر اساس نماد و قراردادهای خاص معرفی کردند که از سیستم نمادین واکنشی یا همان STS[3] بهره می‌برد. آنها برای ایجاد قرارداد تطابق، از بردارهای ترکیبی که الهام‌گرفته از بردارهای موازی‌سازی است، بهره جسته‌اند. قرادادِ تطابق و واسط کاربری سرویس‌ها همراه با هم، قراردادهای تطابق را ایجاد می‌کنند و بعد، آداپتور به عنوان یک مؤلفه واسط، از قراردادها ایجاد می‌گردد که هماهنگ‌کننده سرویس‌های دخیل در یک سیستم، با توجه به قیود تعریفی در قرادادِ تطابق است. بعد از ایجاد آداپتور، تمام ارتباطات بین‌سرویسی از طریق این مؤلفه انجام می‌شود. کامارا و همکارانش بعد از معرفی آداپتور از روش سلسله مراتبی جهت ترکیب سرویس‌ها استفاده کرده‌اند.

استفاده از یک زبان صوری برای بیان واسط کاربری از نقطه قوت‌های این راهکار به شمار می‌رود و همچنین استفاده از نمایش گرافیکی کار را برای طراحان آسان نموده است. از دیگر نقاط قوت می‌توان به سلسله مراتبی بودن آن اشاره کرد که باعث می‌شود به راحتی این چارچوب را در شبکه‌های بزرگ پیاده‌سازی کرد. از نقاط ضعف این راهکار می‌توان به جدایی آداپتور از سیستم اشاره کرد که بر پیچیدگی سیستم می‌افزاید. به صورت کلی می‌توان گفت شالوده چارچوب پیشنهادی در این مقاله تشابهات فراوانی با راهکار پیشنهادی کامارا و همکارانش دارد اما در جزئیات با یکدیگر متفاوتند.

1. Policy - Based - Self - Adaptive Model

2. Labelld Transition System

3. Symbolic Transition System

۱-۳ هدف

هدف در این مقاله استفاده از راهکاری برای تغییر پیکربندی سیستم‌های تطبیق پذیر است به صورتی که این پیکربندی پویا و در حین اجرای سیستم انجام گیرد. همچنین این راهکار را برای مؤلفه‌هایی ارائه می‌کنیم که دارای توانایی تغییر پیکربندی نیستند، یعنی واسط کاربری آنها ممکن است برای سیستم مناسب نباشد و فرض می‌کنیم که مؤلفه‌ها به وسیله مشخصه و واسط کاربری مناسب ساخته می‌شوند. برای این که سیستم و مؤلفه‌های تشکیل‌دهنده آن دارای واسط کاربری قوی باشند از زبان صوری RTS[1] که توسط جین [۱۲] معرفی شده استفاده می‌کنیم و سپس رفتارهای تطبیقی را که مؤلفه‌ها باید از خود نشان دهند تحت قرارداد تطابق بیان کرده و در آخر با ایجاد ترکیبی به نام آداپتور، چارچوب خود را کامل می‌کنیم. برای این کار در ابتدا تعاریف مرتبط با چارچوب را که از کار جین برگرفته شده ارائه می‌کنیم.

۲- راهکار اصلی

در این مقاله برای تطابق بین مؤلفه‌ها از قرارداد تطابق استفاده می‌شود. درست مثل زندگی واقعی ما که در آن قراردادها برای تنظیم روابط به کار می‌روند، رفتار سیستم نرم‌افزاری نیز قراردادها را تنظیم می‌کند. در بیشتر موارد قراردادها ابزارهای مدل‌شده به اسم آداپتورها را تولید می‌کنند تا رفتار سیستم را در یک حالت انتزاعی مدل کند. پیکربندی‌های اصلی چارچوب ارائه‌شده در این مقاله به وسیله آداپتورها ساخته می‌شوند. بعد از ایجاد آداپتورها، سیستم از طریق تغییر پیکربندی خود را با شرایط محیطی و یا اتفاقات غیر منتظره تطبیق می‌دهد. در ابتدای ایجاد چارچوب، سیستم به وسیله یک زبان صوری مدل می‌شود تا دارای نحو و معنای دقیقی باشد.

۲-۱ سیستم انتقال واکنشی و قراردادها

زبان صوری انتخابی برای مدل اولیه یک سیستم قوی باید دارای ویژگی‌هایی از جمله سادگی، قابلیت فهم، استفاده آسان در ابزارهای وارسی مدل و قدرت انتقال رخداد و داده باشد. همچنین در اغلب سیستم‌های مؤلفه‌گرا، ویژگی‌های واسط مؤلفه‌ها محدود شده‌اند و تنها نمادهایی همانند نام، نوع داده و اطلاعات مسیر انتقالات درون ویژگی‌های واسط گنجانده شده است. این کمبود عمدتاً به خاطر نبود معنای صوری قوی در تعریف واسط مؤلفه‌ها به وجود می‌آید و به همین علت امکان وارسی مستقل این گونه مؤلفه‌ها وجود ندارد. راهکار این مقاله با استفاده از سیستم انتقال واکنشی یا همان RTS درصدد پوشش این نقطه ضعف می‌باشد. همچنین تعاریف به کار رفته در ادامه، روی چارچوب پیشنهادی آرنولد و نیویت [۱۳] قابل پیاده‌سازی می‌باشند.

تعریف ۱: یک سیستم انتقال واکنشی به صورت $L = (s^{\cdot}, S, \Sigma, \Delta)$ تعریف می‌گردد که در آن:

- S مجموعه حالات و $s^{\cdot} \in S$ حالت آغازی آن است.
- Σ مجموعه‌ای از رخدادهاست که شامل سه مجموعه غیر اشتراکی رخدادهای ورودی Σ^I، رخدادهای خروجی Σ^O و رخدادهای درونی Σ^H است.
- $\Delta \subseteq S \times \Sigma \times S$ مجموعه‌ای از قدم‌هاست.

RTS شبیه سیستم انتقال برچسبی (LTS) است که برای معنادهی عملگری زبان‌های مدل‌سازی به کار می‌رود. زبان RTS به دلیل جدایی رخدادهای ورودی، خروجی و داخلی از زبان LTS متمایز می‌شود. این تمایز از این نکته نشأت می‌گیرد که در یک کاربرد موازی‌سازی، سیستم روی رخدادهای داخلی و خارجی خود کنترل دارد ولی چون رخدادهای ورودی تحت نظر محیط بر سیستم وارد می‌شوند، نمی‌تواند آنها را کنترل کند. یعنی اگر محیط بخواهد ورودی به سیستم بدهد، آن گاه سیستم نیز اجازه دادن خروجی به محیط را دریافت می‌کند.

بعد از طراحی مدل اولیه سیستم، بخش اعظمی از کار این مقاله را مفهوم قراردادها تشکیل می‌دهند. در دنیای واقعی، قراردادها همواره حداقل دوطرفه هستند. آنها نوعی رابطه تعریف می‌کنند که مسئولیتی را بر عهده طرف مقابلش قرار می‌دهد. در طراحی نرم‌افزار نیز دید چندبعدی به قراردادها پیشنهاد شده است. گاهاً قرارداد را راهی برای توصیف رفتار ترکیب‌ها و تعهدی بین اشیا دانسته‌اند و این مفهوم را می‌توان به عنوان مقدماتی برای الگوهای طراحی قابل استفاده مجدد دانست.

قراردادها در این مقاله مرتبط‌کننده دو مؤلفه هستند. ممکن است مؤلفه‌ها واسط کاربری متفاوت با رفتارهای مختلف داشته و یا مؤلفه‌هایی از سیستم‌های مختلف باشند. برای استفاده از قرارداد تطابق از بردارها کمک گرفته می‌شود.

تعریف ۲: قرارداد تطابق AC یک مجموعه از بردارها همانند $V = (part۱ : event(D), part۲ : event(D))$ است که در آن:

- $part۱$ و $part۲$ رخدادهای قرارگرفته در قسمت اول و دوم سیستم هستند.
- $D \in (!, ?)$ دوتایی است که برای تعیین مسیر ارتباطات رخدادها به کار می‌رود (! برای ارسال و ? برای دریافت).

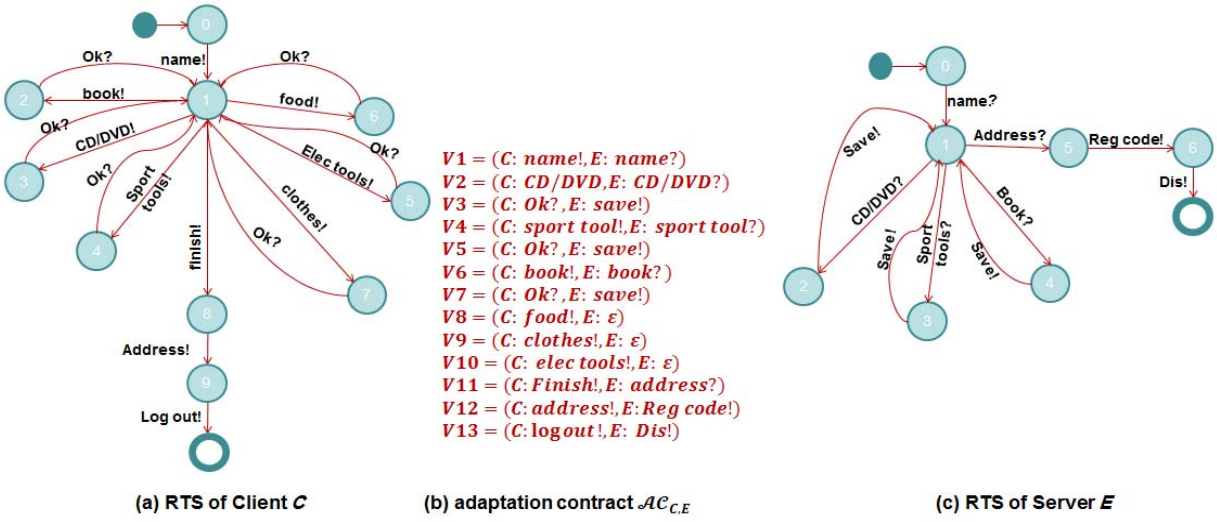
هر رخداد درون یک بردار، دقیقاً به وسیله یک مؤلفه اجرا می‌گردد و نتیجه آن روی بقیه مؤلفه‌ها تأثیر می‌گذارد. کنش‌ها زمانی اجرا می‌شوند که دقیقاً هر دوی رخدادهای قرارگرفته در قسمت‌های ۱ و ۲ در یک زمان اتفاق بیفتند.

حال از مثالی برای نشان‌دادن کار قرارداد تطابق و نحوه استفاده RTSها درون یک سیستم استفاده می‌کنیم. در این مثال از یک سیستم سرویس‌دهنده/ سرویس‌گیرنده استفاده شده تا تعویض دو سرویس‌دهنده درون سیستم را به صورت جزء به جزء بیان کنیم. این مثال یک مرکز خرید برخط را نمایش می‌دهد که شامل دو قسمت می‌باشد. واسط کاربری سرویس‌گیرنده در شکل ۲ قسمت a نمایش داده شده است. دو سرویس‌دهنده مجزا به نام‌های E و H و همچنین واسط کاربری و رفتارهای قابل ارائه آنها در شکل‌های ۲ قسمت c و ۳ قسمت c نشان داده شده‌اند.

در ابتدای شروع تراکنش‌ها، سرویس‌گیرنده به سرویس‌دهنده E متصل می‌شود که به آن پیکربندی c_E گفته می‌شود. سرویس‌گیرنده و سرویس‌دهنده هر دو به قرارداد تطابق $AC_{C,E}$ شکل ۲ قسمت b تعهد دارند. یک پیکربندی جایگزین برای این عملیات خرید، c_H نامیده می‌شود که در آن سرویس‌گیرنده به سرویس‌دهنده H متصل می‌گردد. مشابه آنچه که قبلاً گفته شد، سرویس‌گیرنده و سرویس‌دهنده روی قرارداد تطابق $AC_{C,H}$ توافق کرده‌اند. در حالت معمول تعویض دو سرویس‌دهنده باعث می‌شود سرویس‌گیرنده در سرویس‌دهنده جدید، از حالت آغازین شروع به کار کند.

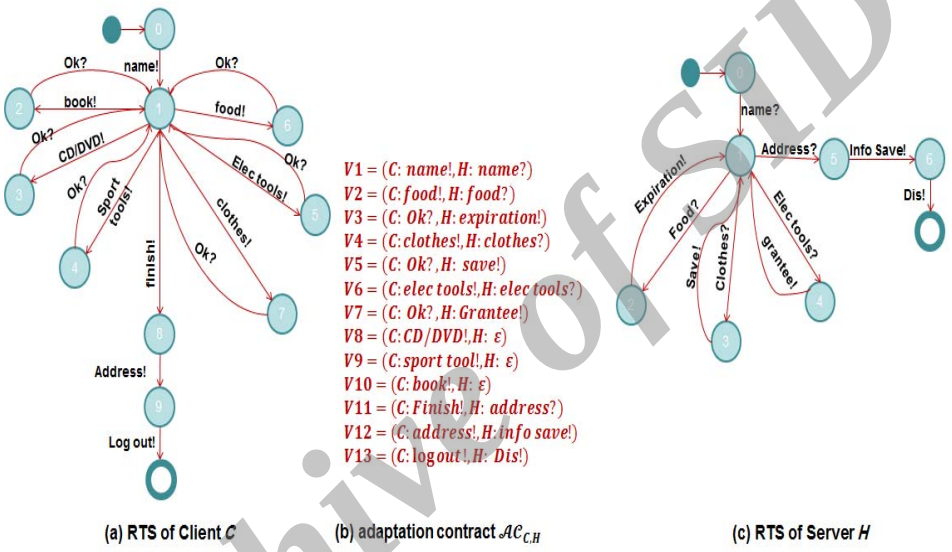
مسأله‌ای که این مقاله به آن پرداخته است، نحوه تعویض دو سرویس‌دهنده به صورت مخفی از سرویس‌گیرنده می‌باشد. به عنوان مثال پیکربندی‌های c_E و c_H به گونه‌ای باید با یکدیگر جایگزین شوند که سرویس‌گیرنده در هر زمان از عملیات خرید، بدون اطلاع از تغییر

1. Reactive Transition System



(a) RTS of Client *C* (b) adaptation contract $\mathcal{AC}_{C,E}$ (c) RTS of Server *E*

شکل ۲: مدل RTS سرویس‌گیرنده و سرویس‌دهنده E و قرارداد تطابق مابین آنها.



(a) RTS of Client *C* (b) adaptation contract $\mathcal{AC}_{C,H}$ (c) RTS of Server *H*

شکل ۳: مدل RTS سرویس‌گیرنده و سرویس‌دهنده H و قرارداد تطابق مابین آنها.

پیکربندی بتواند محصولات مورد نیاز خود را خریداری نماید. برای این کار باید سرویس‌دهنده‌های E و H با یکدیگر تعویض شوند ولی همان گونه که در شکل‌ها دیده می‌شود واسط کاربری این دو سرویس‌دهنده یکسان نمی‌باشد.

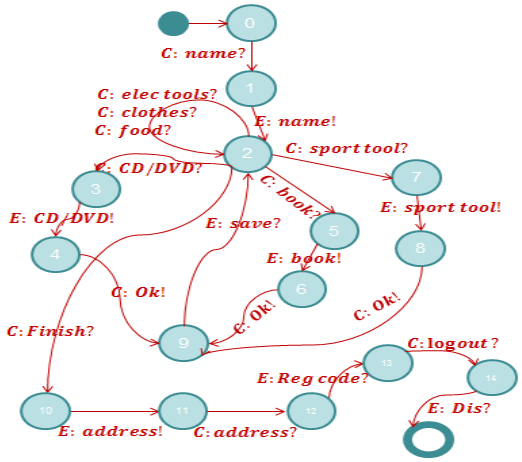
۲-۲ آداپتورها

در مهندسی نرم‌افزار، آداپتورها اکثراً به عنوان یک مؤلفه سوم، درون سیستم قرار می‌گیرند و مسئول هماهنگی سرویس‌های دخیل در سیستم با توجه به مجموعه‌ای از قیود تعریف‌شده از پیش هستند. در سیستم‌های تطبیقی، آداپتورها اکثراً مسئول دریافت، ترجمه و مرتب‌سازی مجدد پیام‌ها هستند به صورتی که برای سرویس بعدی مناسب باشند و می‌توانند به صورت خودکار بر اساس توصیفات انتزاعی تولید شوند که در آنها چگونگی حل ناسازگاری‌ها گفته شده است. همان گونه که قبلاً اشاره شد، این توصیفات انتزاعی را قرارداد تطابق می‌نامیم. در چارچوب پیشنهادی این مقاله، آداپتورها ترکیباتی به شمار می‌روند که شامل قسمت‌های مختلف سیستم و قراردادهای تطابق مابین آنها هستند. همچنین آداپتورها مسئول پیکربندی‌ها، پیکربندی‌های مجدد، بازبینی و تنظیم مجدد آن می‌باشند. آنها مستقیماً از قراردادهای تطابق ساخته می‌شوند که کار هماهنگی بین دو جزء مختلف سیستم را به عهده دارند. بعد از تعریف آداپتور، کار با سیستم به کار با آداپتورها منتهی می‌شود. همچنین برای ترکیب آداپتورها از شبکه و روش سلسله مراتبی استفاده شده است.

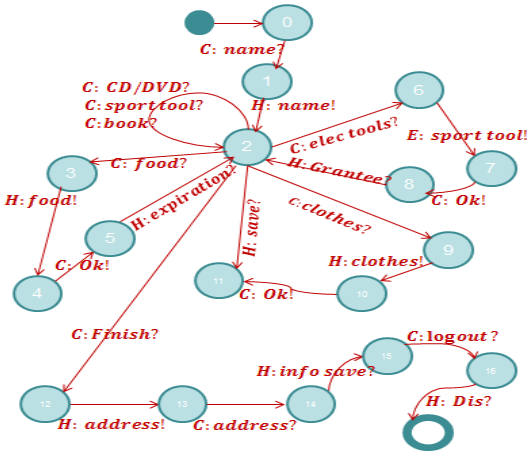
تعریف ۳: فرض کنیم P سیستمی باشد که شامل دو زیرسیستم، همراه با قرارداد تطابق مابین آنها است. جهت ساختن آداپتور برای زیرسیستم‌ها بدین شکل عمل می‌شود:

- برای هر رخداد $\alpha! \in \Sigma_{P_i}^{O}$ قدمی در آداپتور به شکل $\nu = \{P_i : \alpha!, A_p : \alpha?\}$ تعریف می‌شود به این معنی که اگر در حالت P_i یک ورودی به سیستم اتفاق بیفتد، در آداپتور آن حالت به خروجی تغییر می‌یابد.
- برای هر رخداد $\alpha? \in \Sigma_{P_i}^{I}$ قدمی در آداپتور به شکل $\nu = \{P_i : \alpha?, A_p : \alpha!\}$ تعریف می‌شود به این معنی که اگر در حالت P_i یک خروجی به سیستم اتفاق بیفتد، در آداپتور آن حالت به ورودی تغییر می‌یابد.
- در حالاتی که اجازه تغییر پیکربندی داده شده یک رخداد r_{s_i} به آداپتور اضافه می‌شود.

در حالت کلی می‌توان گفت تعریف بالا دو قسمت مختلف سیستم، (برای مثال سرویس‌دهنده و سرویس‌گیرنده) را تبدیل به یک قسمت کلی به نام آداپتور می‌کند. برای این کار از درخواست اولیه سرویس‌گیرنده از سرویس‌دهنده شروع کرده و سپس جواب سرویس‌دهنده به این درخواست



(a) Adaptor $A_{C,E}$



(b) Adaptor $A_{C,H}$

شکل ۴: آداپتورهای ساخته‌شده از سرویس‌گیرنده و سرویس‌دهنده‌ها.

را به آداپتور اضافه می‌کنیم و این کار با درخواست‌های سرویس‌گیرنده و جواب‌های سرویس‌دهنده ادامه می‌یابد. در پایان این چرخه، آداپتور ساخته می‌شود. تنها نکته اصلی، تعویض مسیر ورودی‌ها و خروجی‌ها در هنگام ساخت آداپتور می‌باشد.

آداپتورهای ساخته‌شده برای مثال بخش ۲-۱ در شکل ۴ نشان داده شده است. در اکثر راهکارهای پیشین، آداپتورها به عنوان یک مؤلفه سوم در سیستم قرار می‌گیرند و قسمت‌های مختلف سیستم از طریق عبور از آداپتورها با یکدیگر بدون هیچ ناسازگاری و خطایی ارتباط برقرار می‌کنند. در این گونه سیستم‌ها در فرایند هماهنگی و تطابق بین دو جزء مختلف سیستم، سه قسمت درگیرند. اگر مثال ارائه‌شده در قسمت قبل را در نظر بگیریم، هم سرویس‌گیرنده، هم سرویس‌دهنده و هم قسمتی به نام آداپتور در این فرایند شرکت دارند. راهکار ارائه‌شده در اینجا برای کاهش هزینه‌های کلی بازبینی، حافظه و پیچیدگی به جای استفاده از سه جزء، از یک قسمت به نام آداپتور که شامل هر سه جزء است استفاده می‌کند. همچنین مقیاس‌پذیری سیستم افزایش می‌یابد زیرا اضافه و کم‌کردن قسمت‌های یک سیستم به راحتی به وسیله اضافه یا کم‌کردن بردارهای سازنده آداپتور آن قابل انجام است.

۳-۲ مؤلفه‌های رخداد گسسته و تغییر پیکربندی

همان طور که جین در مقاله خود اشاره کرده بود، RTSها اغلب برای استفاده عملی بسیار ابتدایی هستند و به همین علت جین این سیستم را از دو طریق گسترش داد. در ابتدا برای یک رخداد، دو قسمت نوع و مقدار تعریف می‌شود. نوع‌ها برای دسته‌بندی رخدادها استفاده می‌شوند و مقادیر نمایانگر داده‌های رد و بدل شده یا پارامترهای رخدادها هستند. همچنین RTSها به گذرگاه‌های ورودی/خروجی نیز مجهز می‌شوند که هر گذرگاه یک نوع از رخدادهای خاص را معرفی می‌کند. در نتیجه یک مؤلفه (یا سیستم انتقالی) تنها به وسیله گذرگاه‌ها با دیگران ارتباط برقرار می‌کند. فرض می‌کنیم مقادیر قابل شمارش انتقالی در جهان برابر با ϑ باشد. ما با کمی تغییر روی تعریفی که جین از مؤلفه‌های رخداد گسسته (DEC)[1] در راهکار خود ارائه کرده بود، آن را برای چارچوب پیشنهادی خود مناسب می‌سازیم. تعریف ارائه‌شده به شکل زیر است:

تعریف ۴: یک مؤلفه رخداد گسسته DEC به صورت $C=(\alpha,\theta,s_\cdot,S,\Sigma,\Delta)$ است که در آن:

- α مجموعه متناهی از گذرگاه‌هاست که شامل سه مجموعه غیر اشتراکی گذرگاه‌های ورودی α^I، گذرگاه‌های خروجی α^O و گذرگاه‌های تغییر پیکربندی α^{re} می‌باشد.
- $\theta:\alpha\to 2^{\upsilon}$ یک تابع کامل است که هر گذرگاه را به زیرمجموعه مقادیری از جهان نگاشت می‌کند.
- چندتایی $(s_\cdot,S,\Sigma,\Delta)$ نمایشگر یک RTS است.

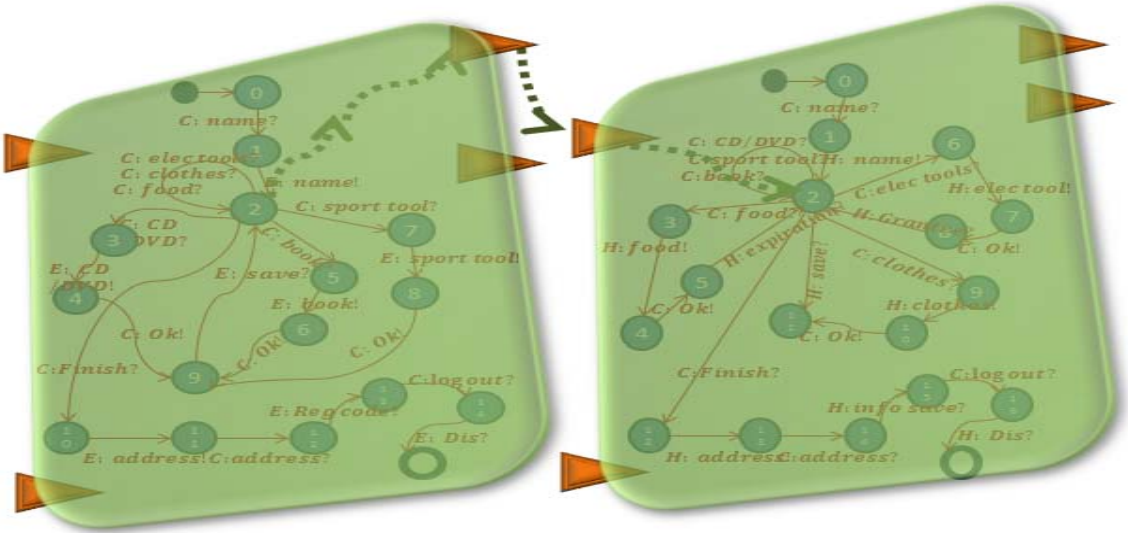
به θ تابع نگاشت گویند که هر گذرگاه را به نوع مقادیری که می‌تواند از آن منتقل شود مرتبط می‌سازد. یک رخداد ورودی/ خروجی از DEC به عنوان یک انتقال پیام از گذرگاه‌های ورودی/خروجی DEC در نظر گرفته می‌شود. همچنین رخدادهای داخلی هر DEC به عنوان رخدادهایی یکتا برای آن محسوب می‌شود. همانند رخدادهای ورودی/خروجی، رخداد تغییر پیکربندی مجدد نیز به گذرگاه مخصوص به خود منتقل می‌شود.

بعد از مدل‌کردن آداپتورها به وسیله DEC ها، ممکن است در قسمت‌هایی از سیستم نیاز به تغییر پیکربندی احساس شود. تغییر پیکربندی سیستم به معنی جایگزینی یک پیکربندی با پیکربندی دیگر است. در راهکار این مقاله، تغییر پیکربندی به جایگزینی قسمتی از سیستم با قسمت دیگر (در اینجا آداپتورها) اطلاق می‌شود که از یک حالت در جزء اول شروع شده و در یکی از حالات جزء بعدی به اتمام می‌رسد.

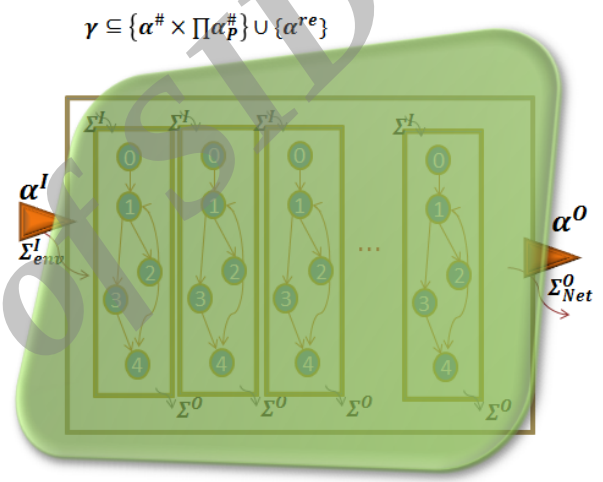
اطلاع‌دهنده شروع این نقل و انتقالات بردار r^* می‌باشد. قسمت اعظمی از کار تغییر پیکربندی، انتخاب حالاتی است که از آنها می‌توان شروع به تغییر پیکربندی کرد یا در آن حالات میزبان تغییر پیکربندی شد. همچنین این تغییر پیکربندی باید به نحوی صورت گیرد تا از دید کاربر خارجی پنهان بماند. انتخاب این حالات خارج از کار این مقاله است و فرض می‌شود که طراحان سیستم در زمان طراحی، این حالات را انتخاب کرده‌اند.

تعریف ۵: قرارداد تغییر پیکربندی به‌صورت $\Re=(A_P,\alpha_P,\Delta_\Re)$ تعریف می‌شود که در آن A_P مجموعه‌ای از آداپتورهاست و α_P پیکربندی اولیه آن است. $\Delta_\Re\subseteq A_P\times R_O\times A_P$ مجموعه‌ای از عملگرهای تغییر پیکربندی است به صورتی که $R_O\subseteq S_i^*\times S_j^*$ و $S_i^*\subseteq A_{p_i}$ حالتی است که از آن تغییر پیکربندی شروع شده و $S_j^*\subseteq A_{p_j}$ حالتی است که بعد از تغییر پیکربندی، سیستم از آن حالت شروع به کار خواهد کرد. برای هر اعلان $r\in R$ به سیستم از گذرگاه α^{re} استفاده می‌شود. انتخاب حالات S_i^* و S_j^* همان گونه که قبلاً اشاره شد، به عهده طراحان سیستم است. برای رسیدن به حالت S_j^* در جزء دوم تغییر پیکربندی از مسیرنمایی استفاده می‌گردد.

1. Discrete Event Systems



شکل ۵: تغییر پیکربندی بین آداپتورهای به فرم مؤلفه‌های رخداد گسسته.



شکل ۶: شبکه‌ای از آداپتورهای به فرم مؤلفه‌های رخداد گسسته.

تعریف ۶: فرض می‌کنیم $L=(s^{\cdot},S,\Sigma,\Delta)$ یک مؤلفه RTS باشد. مسیرنمای ξ شامل دنباله‌ای از رخدادهاست به صورتی که $\xi=\{\sigma\,|\,\sigma=e.\sigma',e\in\Sigma\wedge(s^{\cdot},e,s)\in\Delta,\sigma'\in\xi'\}$ که در آن ξ' مسیرنمایی برای $L'=(s,S,\Sigma,\Delta)$ است.

حال با مثالی تغییر پیکربندی و مسیرنمایی را توضیح می‌دهیم. آداپتورهای طراحی‌شده در بخش ۲-۲ را در نظر بگیرید. چون آداپتور $AC_{C,E}$ اجازه خرید غذا را ندارد، یک قدم تغییر پیکربندی به آن اضافه می‌شود. در آداپتور $AC_{C,H}$ نیز باید حالتی انتخاب شود که از آن حالت بعد از تغییر پیکربندی، پیکربندی جدید شروع به کار کند. در این مثال، $\Re=(A_p,\alpha_{p.},\Delta_{\Re})$ را داریم که در آن $A_p=\{AC_{C,E},AC_{C,H}\}$ و $\alpha_{p.}=AC_{C,E}$ و $\Delta_{\Re}\subseteq AC_{C,E}\times R_{.}\times AC_{C,H}$ به صورتی که $R_{.}\subseteq s_{\gamma}\times s_{\gamma}$، به این معنی که وقتی پیکربندی اولیه به حالت ۲ در آداپتور اولی می‌رسد، سیستم می‌تواند از آداپتور $AC_{C,H}$ با یک قدم تغییر پیکربندی به نام r^{*} به آداپتور $AC_{C,E}$ برود و از آنجا از حالت ۲ شروع به کار کند. همچنین برای رسیدن به حالت ۲ در آداپتور ثانوی، از مسیرنمای ξ کمک گرفته می‌شود. شکل ۵ تغییر پیکربندی بین دو آداپتور را نمایش می‌دهد. همچنین آداپتورها در این شکل به وسیله مؤلفه‌های رخداد گسسته به گذرگاه‌های ورودی خروجی و تغییر پیکربندی مجهز شده‌اند.

۲-۴ شبکه‌ای از آداپتورها

طرح‌های واقعی برای سرویس‌هایی که استفاده مجدد یا تطابق را شامل می‌شوند اکثراً چندین زیرسیستم فعل و انفعالی را دارند. این زیرسیستم‌های چندگانه باعث افزایش پیچیدگی تطابق و در نتیجه افزایش کار طراحان می‌شود. راهکار ارائه‌شده در اینجا بر اساس گزاره سرویس‌های ترکیبی بنا شده که مرتبط با سرویس‌های سلسله مراتبی متصل به هم است. با در محفظه قرار دادن فعل و انفعالات در یک سرویس ترکیبی سلسله مراتبی، طراحان می‌توانند روی ساختمان یک دسته قرارداد، برای انجام یک تطابق خاص در زیرمجموعه‌های یک مسأله توجه بیشتری نمایند. ویژگی‌های این روش در زمان طراحی، توسعه و اجرا برجسته‌تر می‌شود زیرا در مواقع خاصی ممکن است سرویس‌های ترکیبی به صورت جداگانه و مستقل از بقیه سرویس‌ها با عناصر جدید جایگزین شوند. بدین وسیله با استفاده از زیرمسأله‌ها می‌توان به سادگی نیازمندی‌های جدیدی را برای سرویس‌های دیگر یا کاربر فراهم کرد. در اینجا منظور از سرویس‌های ترکیبی همان آداپتورها می‌باشند.

قبلاً در مورد نحوه ساخت سیستم به وسیله RTS ها و همچنین DEC ها سخن گفته شد. حال به وسیله ساختن شبکه‌ای از آداپتورها یک سیستم کامل را توصیف می‌کنیم که بتواند به ابزارهای وارسی مدل اضافه شود.

تعریف ۷: $A_{net}=(\alpha,\theta,P,\gamma)$ یک شبکه از آداپتورهای به فرم مؤلفه‌های رخداد گسسته را تعریف می‌کند به صورتی که:

- α مجموعه متناهی از گذرگاه‌های بیرونی شبکه می‌باشد که شامل گذرگاه‌های ورودی α^{I}، گذرگاه‌های خروجی α^{O} و گذرگاه‌های تغییر پیکربندی α^{re} است و قرار می‌دهیم $\alpha^{\#}=\alpha^{I}\cup\alpha^{O}\cup\{\varepsilon\}$.
- $\theta:\alpha\to 2^{\upsilon}$ یک تابع نگاشت کامل است که هر گذرگاه را به مجموعه‌ای از مقادیر نگاشت می‌کند.
- P مجموعه‌ای از مؤلفه‌های DEC است.
- $\gamma\subseteq\{\alpha^{\#}\times\prod\alpha_{P}^{\#}\}\cup\{\alpha^{re}\}$ با $\alpha^{I}\cup\alpha_{p\in P}^{O}$ شاخص‌گذاری شده است.

به γ مجموعه‌ای از اتصالات گویند. γ یک بردار موازی‌سازی با الگوی موازی‌سازی خاص تعریف‌شده برای دو آداپتور است. قسمت اول γ یا همان $\{\alpha^{\#}\times\prod\alpha_{P}^{\#}\}$ برداری را نمایش می‌دهد که شامل

پیکربندی‌های همه آداپتورهای موجود در شبکه است. برای مثال پیکربندی اولیه هر آداپتور هنگام شروع به کار توسط این بردار نشان داده می‌شود. همچنین اگر در شبکه‌ای چندین آداپتور به صورت موازی در حین اجرای عملیات باشند، این بردار باید نشان‌دهنده پیکربندی هر آداپتور باشد. در هنگام نیاز به تغییر پیکربندی، هر آداپتور بردار موازی‌سازی به نام بردار γ^* را تولید می‌کند که شامل رخدادهای تغییر پیکربندی است. با این بردار سیستم از درخواست تغییر پیکربندی بخشی از سرویس‌های خود آگاه شده و همچنین از مکان آن سرویس جهت یافتن پیکربندی هدف نیز مطلع می‌گردد. شکل ۶ شبکه‌ای از آداپتورهای مولفه‌های رخداد گسسته را نشان می‌دهد.

برای این که شبکه آداپتورها خوش‌تعریف باشد، از تعاریف ارائه‌شده در رساله دکترای جین با کمی تغییر استفاده خواهد شد.

تعریف ۸: شبکه‌ای از آداپتورهای به فرم $D_{net}=(\alpha,\theta,P,\gamma)$ یک اتصال داخلی $f\in\gamma$ را در نظر بگیرید و فرض کنید $\ell\in P$ و نگاشت $\pi_\ell:\theta\to\alpha_\ell$ برای تمام ℓها برقرار باشد. آن گاه تولیدکننده f که با ρ_f نمایش داده می‌شود، یک آداپتور منحصر به فرد است به صورتی که $\pi_\ell(f)\in\alpha_\ell^O$. همچنین $\pi_{\rho_f}(f)$ را پورت تولیدکننده f می‌نامند. کنش پورت α^{re} که باعث تغییر پیکربندی شبکه می‌شود، به وسیله یک آداپتور منحصر به فرد ℓ تولید شده و r_f نامیده می‌شود و داریم $\pi_\ell(f)\in\alpha_\ell^{re}$. پورت تغییر پیکربندی f را $\pi_{r_f}(f)$ می‌نامند. همچنین مجموعه مصرف‌کننده‌های f که با نماد η_f نمایش داده می‌شود، شامل تمام آداپتورهای به فرم ℓ است به صورتی که $\pi_\ell(f)\in\alpha_\ell^I$. برای هر مصرف‌کننده ℓ، $\pi_\ell(f)$ را پورت مصرف‌کننده f می‌نامند و به علاوه، یک آداپتور ℓ را در f بی‌کار گویند اگر $\pi_\ell(f)=\varepsilon$. مجموعه بیکارها با نماد τ_f نشان داده می‌شوند.

تعریف ۹: یک شبکه از آداپتورها به نام D_{net} را خوش‌تعریف نامند هرگاه در آن $\theta_{\rho_f}(\pi_{\rho_f}(f)\subseteq\theta_\ell(\pi_\ell(f))$ برای تمام $f\in\gamma$ ، $\ell\in\tau_f$ اتفاق بیفتد.

این تعریف تضمین‌کننده ارتباط صحیح پورت‌های دارای مقادیر متناسب در یک شبکه از آداپتورها است. این کار منجر به سازگاری گونه‌های مختلف داده با یکدیگر می‌شود، بنابراین برای ساخت شبکه‌ای از آداپتورها حتماً خوش‌تعریفی آن در نظر گرفته خواهد شد.

برای این که نشان دهیم هر تغییری درون شبکه، بر آداپتورهای دخیل در آن تأثیر می‌گذارد قضیه زیر را ارائه می‌کنیم.

قضیه ۱: یک شبکه از آداپتورهای به فرم مؤلفه‌های DEC را که به شکل $A=(\alpha,\theta,P,\gamma)$ هستند، در نظر بگیرید و فرض کنید $L_N=(s_N^\cdot,S_N,\Sigma_{L_N},\Delta_N)$ محصول موازی‌شده این شبکه باشد. همچنین فرض کنید ξ مسیرنمایی از هر آداپتور منحصر به فرد ℓ با شروع از حالت اولیه $s_\ell^\cdot$ باشد. ثابت می‌کنیم که هر کنش $a\in\alpha$ باعث برداشتن قدمی توسط آداپتورها می‌شود و این قدم می‌تواند پیکربندی یا تغییر پیکربندی باشد.

برهان: این قضیه با استقرا روی کنش‌ها ثابت می‌گردد.

۱) اگر $a\in\alpha_A^I$ آن گاه به ازای هر $\ell\in P$ اگر وجود داشته باشد $s^\cdot\in S$ و $\pi_\ell(s^\cdot)$ و همچنین اگر داشته باشیم $(s^\cdot,a,s')\in\Delta_\ell$ آن گاه داریم $(s^\cdot,a,s')\in\Delta_N$.

یعنی اگر a یک کنش ورودی شبکه باشد و این کنش هر آداپتوری همانند $\ell\in P$ را وادار به برداشتن یک قدم کند، آن گاه این قدم یک قدم از محصول موازی‌شده شبکه می‌باشد.

۲) اگر در هر $\ell\in P$ خاص، داشته باشیم $a_1,a_2,\ldots,a_{m-1}\in\Sigma_I^H$ و اگر $a_m\in\Sigma_I^H$ آن گاه $\pi_\ell(s)\in a_m$ و $(s,a_m,s')\in\Delta_\ell$ بنابراین $(s,a_m,s')\in\Delta_N$.

یعنی اگر دنباله‌ای از کنش‌های داخلی شبکه را داشته باشیم و کنش بعدی دنباله نیز یک کنش داخلی باشد آن گاه این کنش آداپتور مسئول را وادار به برداشتن یک قدم داخلی می‌کند و این قدم داخلی، قدمی از محصول موازی‌شده شبکه محسوب می‌شود.

۳) اگر در هر $\ell\in P$ خاص، داشته باشیم $a_1,a_2,\ldots,a_{m-1}\in\Sigma_I^O$ و وجود داشته باشد $s_i^*\in S_\ell$ و همچنین اگر داشته باشیم $a_m\in\{(f,v)\,|\,f\in\alpha^{re},v\in\theta(f)\}$ آن گاه می‌توان نوشت $\exists\ell'\in P,\eta_{\gamma^*}=\ell'\wedge s_j^*\in S_{\ell'}$ و $\exists\gamma^*\in\gamma,rec_{r^*}=\ell\wedge a=\pi_{rec_{r^*}}(\gamma^*)$ و در نتیجه داریم $(s_i^*,a_m,s_j^*)\in\Delta_N$.

یعنی اگر دنباله‌ای از کنش‌های داخلی شبکه را داشته باشیم و کنش بعدی کنشی خارجی از L_N باشد و همچنین اگر آداپتوری همانند $\ell\in P$ را داشته که دارای یک قدم تغییر پیکربندی s_i^* است که از آن قدم، ℓ درخواست تغییر پیکربندی داده، آن گاه ℓ مجموعه‌ای از اتصالات γ^* را تولید می‌کند که این اتصالات درخواست تغییر پیکربندی را به آداپتور مصرف‌کننده ℓ' ارسال می‌کند و آن گاه بعد از تغییر پیکربندی، ℓ' از حالت s_j^* شروع به کار می‌کند. البته در نظر بگیرید که برای رسیدن به حالت s_j^* در آداپتور ℓ' از مسیرنمایی σ استفاده می‌گردد.

۴) اگر در هر $\ell\in P$، $a_1,a_2,\ldots,a_{m-1}\in\Sigma_I^O$ و اگر $a_m\in\alpha_I^O$ آن گاه وجود دارد $\ell\in P$ و $\gamma\in\Gamma,\rho_\lambda=\ell\wedge a_m=\pi_{env}(\gamma)$ و در نتیجه $(s,a_m,env)\in\Delta_N$.

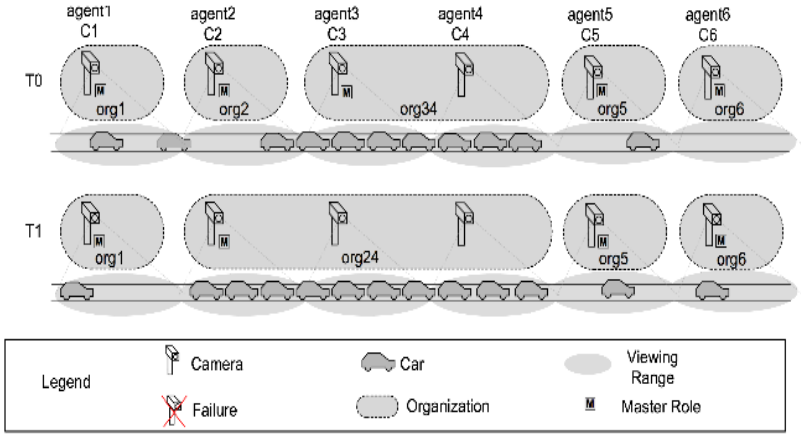
یعنی اگر دنباله‌ای از کنش‌های در شبکه را داشته باشیم و کنش بعدی، کنشی خارجی برای شبکه باشد آن گاه آداپتور $\ell\in P$ و مجموعه‌ای از اتصالات γ را داریم که آداپتور ℓ آن را تولید می‌کند و این اتصالات باعث می‌شوند که شبکه خروجی به محیط دهد و این یعنی خروجی جزء محصول شبکه می‌باشد.

هدف از اثبات قضیه ۱ نشان‌دادن تأثیرگذاری هر رخداد درون یک شبکه بر تمام آداپتورهای دخیل در آن است. با توجه به این اثبات از رسیدن سیستم به اهداف خود در شبکه‌ای با بردارهای موازی‌سازی صحیح اطمینان حاصل می‌شود. در لم بعدی جزئیات بیشتری درباره قضیه فوق توضیح داده خواهد شد. در این لم با استفاده از تصویر مسیرنما نشان داده می‌شود که در یک آداپتور خاص درون شبکه، چه اتفاقی در زمان موازی‌سازی رخ می‌دهد.

لم ۱: رخدادهای حاصل از یک مسیرنما در شبکه‌ای از آداپتورها در یک آداپتور خاص از درون همان شبکه قابل پیگیری است.

برهان: یک شبکه از آداپتورهای به فرم DEC همانند $D_{net}=(\alpha,\theta,P,\gamma)$ و یک آداپتور منحصر به فرد به نام $\ell\in P$ را در نظر بگیرید. همچنین فرض کنید L_N محصول موازی‌شده D و γ^* بردار موازی‌سازی برای تغییر پیکربندی باشد که یک رخداد خروجی هر آداپتور به فرم ℓ' که با ℓ فرق دارد را به رخداد ورودی مرتبط با ℓ نگاشت می‌کند. آداپتور ℓ' مؤلفه ℓ را وادار به برداشتن یک قدم ورودی می‌کند. همچنین فرض می‌شود i و j عضوهایی از حالات تغییر پیکربندی هستند. تصویر مسیرنمای σ درون شبکه با توجه به ℓ به صورت زیر تعریف می‌گردد

۱. $\pi_\ell(\sigma_n)=\pi_\ell(\sigma_{n-1})$
if $e\in\lambda$



شکل ۷: سیستم دیده‌بانی ترافیک.

۲. $\pi_\ell(\sigma_n) = e.\pi_\ell(\sigma_{n-1})$
if $e \in \Sigma_\ell^H$

۳. $\pi_\ell(\sigma_1) = e.\pi_\ell(\sigma_i)$
if $e \in \alpha_{\ell'}^O$ and, $\exists \gamma^*, e = \pi(\gamma^*)$ and $\rho_{\gamma^*} = \ell', \eta_{\gamma^*} = \ell$

۴. $\pi_\ell(\sigma_n) = e.\pi_\ell(\sigma_{n-1}) = \pi_{\ell'}(\sigma_j)$
if $e \in \alpha_\ell^O$ and, $\exists \gamma^*, e = \pi(\gamma^*)$ and $\eta_{\gamma^*} = \ell'$

۵. $\pi_\ell(\sigma_n) = \pi_\ell(\sigma_{n-1})$
if $e \in \alpha_\ell^O$ and, $\nexists \gamma^*, e = \pi_{\rho_{\gamma^*}}(\gamma^*)$

تصویر مسیرنمای $\pi_\ell(\sigma_n)$ روی ℓ، یک دنباله از رخدادهایی است که آداپتور خاص ℓ هنگام برداشتن قدمی از حالت اولیه s_N^i درون شبکه N ثبت می‌کند. این رخدادها به صورت بازگشتی با در نظر گرفتن هر رخداد e دخیل در مسیرنمای σ محاسبه می‌شوند. اگر e یک رخداد داخلی در ℓ باشد آن گاه به مسیرنما اضافه می‌گردد. اگر e یک رخداد خارجی از پورت هر مؤلفه DEC همانند ℓ' به غیر از ℓ باشد و یک بردار موازی‌سازی γ^* موجود باشد که ℓ مصرف‌کننده آن است، آن گاه e به عنوان اولین رخداد به مسیرنمای σ اضافه می‌شود. همچنین این مسیرنما کار خود را از حالت iام در ℓ ادامه می‌دهد. اگر e یک رخداد خروجی از ℓ باشد که برای تغییر پیکربندی آن ایجاد شده است، آن گاه با توجه به بردار موازی‌سازی γ^*، مسیرنمای σ دنباله خود را از حالت j از مؤلفه مصرف‌کننده γ^* یا همان ℓ' ادامه می‌دهد و در آخر e یک رخداد خروجی از شبکه است اگر هیچ تغییر پیکربندی را در پی نداشته باشد. در غیر این شرایط، رخداد e نادیده گرفته می‌شود.

۳- بررسی سیستم ترافیک با چارچوب پیشنهادی

در این بخش یک بررسی موردی درباره "رفتار تطبیقی در یک سیستم غیر متمرکز" [۱۴] خواهیم داشت. ایفتیخار و وینز در ۲۰۱۲ یک وارسی صوری روی این مورد داشته‌اند.

۳-۱ مشخصه‌های سیستم

سیستم دیده‌بانی ترافیک در مورد مشکلات ترافیکی اطلاعاتی را ارائه می‌کند. اهداف اصلی سیستم عبارتند از: ۱) اطلاع به کاربران در مورد تغییرات پویای ترافیک، ۲) درک این وظایف به صورت غیر متمرکز و ۳) توانمندساختن سیستم نسبت به خرابی دوربین‌ها. سیستم شامل چندین دوربین هوشمند است که در طول راه پخش شده‌اند و مثالی از بزرگ‌راه در شکل ۷ نشان داده شده است. همچنین مؤلفه‌های سیستم عبارتند از:

۱) ماشین‌ها که در محیط و تحت نظر دید هر یک از دوربین‌ها حرکت می‌کنند.

۲) دوربین‌ها که سه حالت اصلی دارند. در حالت معمول، دوربین می‌تواند رهبر بدون پیرو، رهبر با چندین پیرو و تنها پیرو باشد.

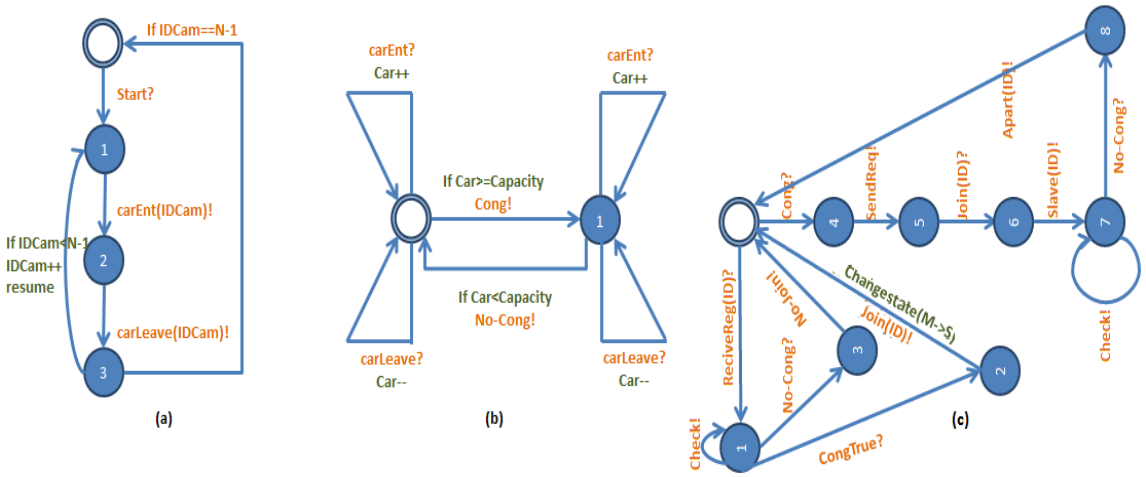
۳) دیده‌بان سیستم که شرایط واقعی ترافیک را به وسیله علایم ارسالی از دوربین‌ها و ماشین‌ها کنترل می‌کند. برای هر دوربین یک دیده‌بان سیستم درحال انجام وظیفه است.

۳-۲ عملکرد سیستم

هنگامی که سیستم به راه می‌افتد، هر دوربین مرتبط با یک سازمان تک‌دوربینی می‌باشد. البته وقتی که ترافیک سنگینی در حیطه دید یک دوربین کشف شود، سازمان این دوربین همراه با همسایه‌های آن تبدیل به یک سازمان واحد خواهند شد. برای ساده‌سازی مدیریت سازمان‌ها و ارتباط با مشتری‌ها، سازمان‌ها دارای ساختار رهبر/ پیرو می‌باشند. رهبر مسئول مدیریت پویای یک سازمان (ترکیب و جداسازی) به وسیله موازی‌سازی پیروها و همسایه‌های خود می‌باشد. همچنین رهبر شرایط ترافیکی را به مشتری‌های خود (در اینجا ماشین‌ها یا پلیس ترافیکی) اطلاع می‌دهد. بنابراین می‌توان گفت رهبر اطلاعات ارسالی توسط پیروهای خود در مورد شرایط ترافیکی خودشان را جمع‌بندی می‌کند. در T۰ از شکل ۷، چهار سازمان تک‌عضوی به نام‌های org۱ با agent۱، org۲ با agent۲، org۵ و org۶ دیده می‌شوند. همچنین یک سازمان ترکیبی به نام org۳۴ با agent۳ به عنوان رهبر و agent۴ به عنوان پیرو نیز دیده می‌شود. در T۱، ترافیک سنگین در حیطه دید دوربین‌های ۲ و ۳ دیده می‌شود و در نتیجه، سازمان org۲ و org۳۴ ترکیب شده و سازمان org۲۴ را با رهبری agent۲ تشکیل می‌دهند. هنگامی که مشکل ترافیک برطرف گردد، سازمان‌ها دوباره جدا و به شکل اولیه باز خواهند گشت.

۳-۳ سیستم در چارچوب پیشنهادی این مقاله

ایفتیخار و وینز از دو سناریو برای تطابق استفاده کرده‌اند. در اینجا ما فقط روی یکی از آنها متمرکز خواهیم شد که نیاز به ترکیب و تغییر پیکربندی پویا دارد. این سناریو شامل تطابق پویای سازمان از T۰ تا T۱ می‌باشد که در آن دوربین ۲ به سازمان دوربین‌های ۳ و ۴ وصل گردیده و ترافیک را تحت نظر می‌گیرند. برای رسیدن به این هدف، به الگوهای ماشین‌ها، دیده‌بان ترافیک و دوربین‌ها نیازمندیم. همان طور که در شکل ۸ قسمت a دیده می‌شود، ماشین‌ها به محدوده دید دوربین‌ها وارد و



شکل ۸: الگوهای ماشین‌ها، دوربین و دیده‌بان ترافیک.

Algorithm 1 *Adaptor* of traffic system
Inputs car processes, traffic monitor and camera processes, signals of help from other *Adaptors*
Outputs signals of help or information about traffic to other *Adaptors*

1: set the first state M
2: observe cameras rang of view with vectors of traffic monitor
3: **if** $\exists v_{TM} =: \langle TM : Cong! \rangle$ && the first state == M **then**
4: send signals to adaptor with ID $n-1, n+1$ and request help // by $r \in \Sigma^O$
5: **if** $\exists v_{env} =: \langle env : Join! \rangle$ **then** // by $r \in \Sigma^I$
6: compose this adaptor to your own network and reconfigure to a Master with Slave organization and set the first state to MWS
7: **end if**
8: **do** observe cameras range of view
9: **while** receive $v_{TM} =: \langle TM : No-Cong! \rangle$
10: send signal Apart to slaves and return to configuration Master of single organization and set the first state to M // by $r \in \Sigma^O$
11: **end if**
12: **goto** 1
13: **else if** $\exists v_{env} =: \langle env : Join! \rangle$ **then** // by $r \in \Sigma^I$
14: **goto** 1
15: **end if**
16: **if** receive signal $v_{env} =: \langle env : ReceiveReq(ID)! \rangle$ && the first state == M **then** // by $r \in \Sigma^I$
17: observe cameras rang of view with vectors of traffic monitor
18: **if** $\exists v_{TM} =: \langle TM : Cong! \rangle$ **then**
19: send signal $v_{cam} =: \langle C : Join! \rangle$ to adaptor (ID) and change the first state to S
20: **else if** $\exists v_{TM} =: \langle TM : No-Cong! \rangle$ **then**
21: send signal $v_{cam} =: \langle C : No-Join! \rangle$ to adaptor (ID) and **goto** 1
22: **end if**
23: **else if** receive signal $v_{env} =: \langle env : ReceiveReq(ID)! \rangle$ && the first state == MWS **then** // by $r \in \Sigma^I$
24: **goto** 6
25: **end if**

شکل ۹: نحوه کار آداپتورها.

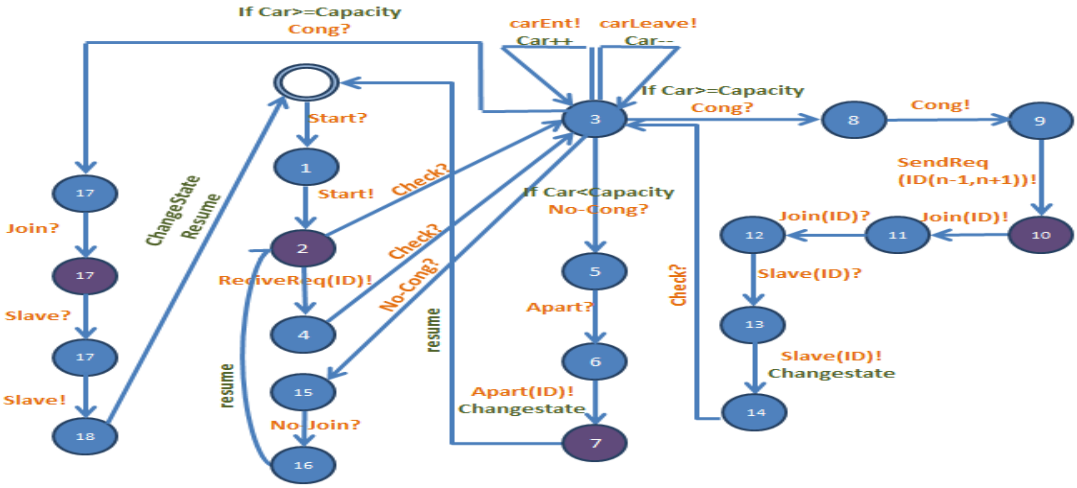
خارج می‌شوند. شکل ۸ قسمت b نمایان‌گر دیده‌بان ترافیک است به صورتی که هرگاه ماشینی داخل محدوده دوربین‌ها گردد به وسیله کانال‌های carEnt و carLeave مشاهده می‌شود. دیده‌بان ترافیک به وسیله مقایسه تعداد ماشین‌های درون محدوده دوربین‌ها با ظرفیت وجود ترافیک را تشخیص می‌دهد. شکل ۸ قسمت c نیز نمایشگر فرایند کار دوربین‌ها می‌باشد. هر دوربین به وسیله سیگنالی که از ماشین‌ها دریافت می‌کند با دیده‌بان ترافیک کار می‌کند.

برای داشتن یک ارتباط امن بین سه مؤلفه سیستم، قرارداد تطابق به سیستم اضافه می‌گردد. قراردادها تنظیم‌کننده ارتباطات بین این سه مؤلفه هستند. این قرارداد تطابق به صورت زیر می‌باشد

$$AC_{CCME} = \{\nu_{car,env} =: \langle C : start!, E : start? \rangle$$
$$\nu_{cam,TM} =: \langle C : check!, TM : Cong! \| No-Cong! \rangle$$
$$\nu_{TM,cam} =: \langle TM : Cong?, C : Cong! \rangle$$
$$\nu_{cam,env} =:$$
$$C\langle : SendReq(ID_{n-1,n+1})?, E : ReciveReq(ID_{n-1,n+1})! \rangle$$
$$\nu_{cam,env} =: \langle C : Join(ID)?, E : Join(ID)! \rangle$$
$$\nu_{cam,env} =: \langle C : Slave(ID)!, E : Slave(ID)? \rangle$$
$$//change\ state$$
$$\nu_{TM,cam} =: \langle TM : No-Cong?, C : No-Cong! \rangle$$
$$\nu_{cam,env} =: \langle C : Apart!, env : Apart? \rangle //change\ state$$
$$\nu_{cam,env} =: \langle C : \varepsilon, E : SendReq! \rangle$$
$$\nu_{cam,env} =: \langle C : ReciveReq(ID)!, E : \varepsilon \rangle$$
$$\nu_{cam,TM} =: \langle C : check!, TM : Cong! \| No-Cong! \rangle$$
$$\nu_{cam,env} =: \langle C : No-Join?, E : \varepsilon \rangle$$
$$\nu_{cam,env} =: \langle C : Join?, E : \varepsilon \rangle$$
$$\nu_{cam,env} =: \langle C : \varepsilon, E : Slave! \rangle$$
$$\nu_{cam,env} =: \langle C : Slave?, E : \varepsilon \rangle //change\ state\}$$

قرارداد تطابق شامل فرایندهای ماشین‌ها، دوربین‌ها و دیده‌بان ترافیک و محیط می‌باشد. در اینجا محیط شامل دیگر آداپتورهایی است که می‌توانند سیگنال‌های تغییر پیکربندی و سیگنال‌های ارتباطی برای تبدیل به سازمان تک‌رهبری یا رهبر همراه با پیرو و یا پیرو می‌باشد. این سیگنال‌ها را بردارهای موازی‌سازی درون یک شبکه ارسال می‌کنند. الگوریتم شکل ۹ نحوه کار آداپتورها را نشان می‌دهد.

شکل ۱۰ آداپتور سیستم ترافیک را نمایش می‌دهد. حالات تغییر پیکربندی که در آنها سیگنال‌های تغییر پیکربندی دریافت و ارسال می‌گردد، حالات ۲، ۷، ۱۰ و ۱۷ می‌باشند. قضیه ۱ تضمین‌کننده ارسال امن سیگنال‌ها درون شبکه است. بنابراین نشان دادیم که با اعمال چارچوب پیشنهادی این مقاله به مدل سیستم ترافیک، فرمول‌بندی و مدل پیشنهادی برای نیازمندی‌های این مدل کافی می‌باشد. همچنین این چارچوب قادر است آداپتورها را طوری تعویض نماید که به اجرای سیستم خللی وارد نگردد.



شکل ۱۰: آداپتور سیستم ترافیک.

۴- نتیجه‌گیری

در این مقاله، چارچوبی صوری برای توصیف پیکربندی سیستم‌های تطبیقی ارائه شد. برای این کار در ابتدا سیستم با یک زبان صوری بامعنا بر پایه مدل RTS و سپس DEC ها بنا شد که این کار به قانون‌مندی سیستم و استفاده آسان در ابزارهای وارسی مدل کمک می‌کند. برای جلوگیری از ناسازگاری‌ها قرارداد تطابق معرفی شد و بعد از آن ارتباط کلیه قسمت‌های سیستم به وسیله قرارداد تطابق با استفاده از آداپتورها میسر گشت. در انتهای کار، شبکه‌ای از آداپتورها معرفی گردید تا چارچوب پیشنهادی کامل گردد.

یکی از نقطه قوت‌های چارچوب پیشنهادی این مقاله داشتن ساختاری بازگشتی است. بیشتر چارچوب‌های پویا تنها با تغییر پیکربندی مؤلفه‌های جعبه سیاه درگیرند. بر خلاف این گونه راهکارها، چارچوب پیشنهادی ما نه تنها از تغییر پیکربندی پشتیبانی می‌کند بلکه در آن جابه‌جایی و ترکیب آداپتورها نیز قابل اجرا می‌باشند. به عبارت دیگر کل چارچوب از بالا به پایین به صورت بخش‌های جدا به یکدیگر وصل شده‌اند که نشان‌دهنده استقلال قسمت‌های گوناگون آن است. بنابراین جدا یا حذف‌شدن قسمت‌ها بدون ایجاد خللی در کار بقیه آداپتورها قابل انجام است. این ویژگی در مدل کامارا نیز دیده می‌شود. از طرف دیگر با توجه به تعریف مسیرنما، چارچوب این مقاله توانایی یافتن مؤلفه‌های درونی آداپتورها را دارد که چارچوب کامارا از این ویژگی پشتیبانی نمی‌کند.

دیگر ویژگی این چارچوب پشتیبانی از تغییر پیکربندی در حین اجرا است. قضیه یک قسمت سه تضمین‌کننده وجود این تغییر پیکربندی می‌باشد. همچنین توصیف صوری از کلیه اجزای سیستم به دلیل استفاده از زبانی ریاضی‌وار در ساخت آن موجود است. دیگر ویژگی این چارچوب داشتن مدیریتی خارجی می‌باشد. به دلیل استفاده از قراردادها برای ارتباطات بین مؤلفه‌ای که مابین مؤلفه‌های جعبه- سیاه[1] قرار می‌گیرند، مدیریت جهت نبود هیچ گونه ناسازگاری بیرونی می‌باشد یعنی به جای مدیریت مؤلفه‌ها از درون، مدیریت به وسیله قراردادها و بردارهای موازی‌سازی و تنها از طریق واسطها اعمال می‌گردد.

استفاده از مسیرنمایی و رخدادهای گذشته جهت تغییر پیکربندی در هر حالت دلخواه از سیستم، به کارهای آینده موکول می‌گردد. همچنین چون این چارچوب از تعاریف پیشنهادی جین پیروی می‌کند و راهکار جین از ترکیب صوری مؤلفه‌ها جهت جلوگیری از انفجار فضای حالت حمایت می‌کند، ممکن است راهکار پیشنهادی ما نیز دارای چنین ویژگی باشد. البته بررسی وجود یا عدم وجود این ویژگی نیز از کارهای آینده ما محسوب می‌گردد.

1. Black-Box

مراجع

[1] A. Colman, *Role - Oriented Adaptive Design*, Ph. D. Thesis, Swinburne University, Melbourn, Australia, 2007.

[2] T. Batista, A. Joolia, and G. Coulson, "Managing Dynamic Reconfiguration in Component - Based Systems," in *Proc. 2nd European WorkShop on Software Architectures*, pp. 1-17, Pisa, Italy, 13-14 Jun. 2005.

[3] J. Aldrich, C. Chambers, and D. Notkin, "ArchJava: connecting software architecture to implementation," in *Proc. of the 24th Int. Conf. on Software Engineering ICSE'04*, pp. 187-197, Orlando, US, May 2004.

[4] I. Sander and A. Jantsch, "Modelling adaptive system in ForSyDe," *Electronic Notes in Theorical Computer Science*, vol. 200, no. 2, pp. 39-54, Feb. 2008.

[5] I. Georgiadis, J. Magee, and J. Kramer, "Self - organising software architecture for distributed systems," in *Proc. of the 1st Workshop on Self - Healing Systems, WOSS'02*, pp. 33-38, New York, US, 18-19 Nov. 2002.

[6] D. Garlen, S. Cheng, A. Huang, B. Schmerl, and P. Streenkiste, "Rainbow: architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46-54, Oct. 2004.

[7] D. Garlan, V. Poladian, B. Schmerl, and J. P. Sousa, "Task - based self - adaptation," in *Proc. of the 1st ACM SIGSOFT Workshop on Self - Managed Systems*, pp. 54-57, California, 31 Oct.-1 Nov. 2004.

[8] N. Khakpour, S. Jalili, C. Talcott, M. Sirjani, and M. Mousavi, "Formal modeling of evolving self - adaptive systems," *J. of Science of Computer Programming*, vol. 78, no. 1, pp. 3-26, Nov. 2011.

[9] R. Bruni, A. Corradini, F. Gadducci, A. Lluch, and A. Vandin, "A conceptual framework for adaptation," *Lecture Notes in Computer Science*, vol. 7212, pp. 240-254, 2012.

[10] A. Cansado, J. Cubo, G. Salaun, and C. Canal, "A formal framework for structural reconfiguration of component under behavioral adaptation," *Electronic Notes in Theorical Computer Science*, vol. 263, pp. 95-110, 3 Jun. 2011.

[11] J. Camara, G. Salaun, C. Canal, and M. Ouederni, "Interactive specification and verification of behavioural adaptation contracts," *Information and Software Technology*, vol. 54, no. 7, pp. 701-723, Jul. 2012.

[12] Y. Jin, C. Lakos, and R. Esser, "Modular consistency analysis of component-based designs," *Journal of Research and Practice in Information Technolog*, vol. 36, no. 3, pp. 186-208, Aug. 2004.

[13] A. Arnold, Finite Transition Systems: Semantics of Communicating Systems, Prentice - Hall, 1994.

[14] M. U. Iftikhar and D. Weyns, "A case study on formal verification of self - adaptive behaviors in a decentralized system," in *Proc. 11th Int. Workshop on Foundations of Coordination Languages and Self Adaptation, FOCLASA'12*, pp 45-62, Aug. 2012.

جابر کریم‌پور تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد به ترتیب در سال‌های ۱۳۷۶ و ۱۳۷۹ در رشته ریاضی کاربردی با گرایش کامپیوتر در دانشگاه تبریز به پایان رسانده است. ایشان همچنین در سال ۸۷ موفق به اخذ درجه دکتری در رشته علوم کامپیوتر از همین دانشگاه شده است. در حال حاضر دکتر کریم‌پور عضو هیأت علمی گروه علوم کامپیوتر تبریز و مدیر مرکز فناوری اطلاعات این دانشگاه، از سال ۱۳۹۱ می‌باشد. زمینه تحقیقاتی ایشان شامل توصیف و وارسی سیستم، امنیت نرم‌افزار و رمزنگاری است

رباب علیاری در سال ۱۳۸۹ مدرک کارشناسی خود را در رشته ریاضی کاربردی و در سال ۱۳۹۳ مدرک کارشناسی ارشد خود را در رشته علوم کامپیوتر از دانشگاه تبریز اخذ نموده است. نام‌برده هم‌اکنون دانشجوی دکتری رشته علوم کامپیوتر همین دانشگاه از سال ۱۳۹۳ می‌باشد. زمینه‌های تحقیقاتی ایشان عبارتند از: روش‌های صوری، سیستم‌های تطبیقی، امنیت سیستم‌های تطبیقی و مهندسی نرم‌افزار.