تاریخ دریافت مقاله: ۹۶/۰۴/۱۲
تاریخ پذیرش مقاله: ۹۶/۰۵/۳۰

خود–ترمیمی در سازوکار کنترلی سامانه‌های تطبیق‌پذیر

علی طریحی
دانشکده مهندسی و علوم کامپیوتر- دانشگاه شهید بهشتی - تهران - ایران
پست الکترونیکی: a_tarihi@sbu.ac.ir

حسن حقیقی*
دانشکده مهندسی و علوم کامپیوتر- دانشگاه شهید بهشتی - تهران - ایران
پست الکترونیکی: ir.ac.sbu@haghighi_h

فریدون شمس علیئی
دانشکده مهندسی و علوم کامپیوتر- دانشگاه شهید بهشتی - تهران - ایران
پست الکترونیکی: f_shams@sbu.ac.ir

چکیده

امروزه تطبیق‌پذیری‌های پیچیده، واکنش به وضعیت‌های پیچیده و ویژگی‌های خود-* سامانه‌های رایانه‌ای جدید بر عهده سازوکارهای کنترلی گذاشته می‌شود. این موضوع باعث شده است که مأموریت سازوکارهای کنترلی حیاتی بوده و در نتیجه، درصورتی‌که منطق تطبیق یا مؤلفه‌های آن دچار شکست شوند، رفتار سامانه غیرقابل‌پیش‌بینی گردد. این مقاله سعی دارد تا راهی برای رفع این مشکل با الهام از زیست‌شناسی بیابد. این هدف با معرفی یک الگو برای پشتیبانی از ویژگی خود-ترمیمی در یک سازوکار کنترلی شناخته‌شده صورت می‌پذیرد. ویژگی افزوده‌شده به سازوکار کنترلی انتخابی، این امکان را می‌دهد که این سازوکار عملکرد خود را در صورت روی دادن تغییرات ناخواسته در محیط یا خرابی‌های داخلی بازیابی کند. بدین شکل، از بروز مشکل جدی ناشی از فقدان سازوکار کنترلی جلوگیری به عمل می‌آید. الگوی پیشنهادی توصیف و درستی‌یابی صوری شده و کاربردپذیری آن از طریق یک مطالعه موردی نشان داده‌شده است.

واژه‌های کلیدی: سازوکارهای کنترلی، الگوی ناظر/کنترل‌کننده، ویژگی‌های خود-*، خود-ترمیمی، الهام از زیست‌شناسی.

۱- مقدمه

افزایش پیچیدگی در سامانه‌های رایانه‌ای جدید، به ایجاد نوع جدیدی از سازوکارهای کنترلی[1] منجر شده است [۱]؛ سازوکارهایی که نیاز به تطبیق دارند تا از دخالت عنصر انسانی بکاهند [۲] و امکان پاسخ به تغییرات را ایجاد نمایند [۳]. اما خود سازوکار یکی از عوامل ایجادکننده پیچیدگی به شمارمی‌رود، چون سامانه‌های تطبیق‌پذیر از نوع حلقه بسته[2] بوده و بر اساس بازخورد از محیط و سامانه عمل می‌نمایند [۲]. سازوکار بازخوردی[3] [۴, ۵] یکی از قدیمی‌ترین سازوکارهای کنترلی به شمار

* نویسندهٔ مسئول

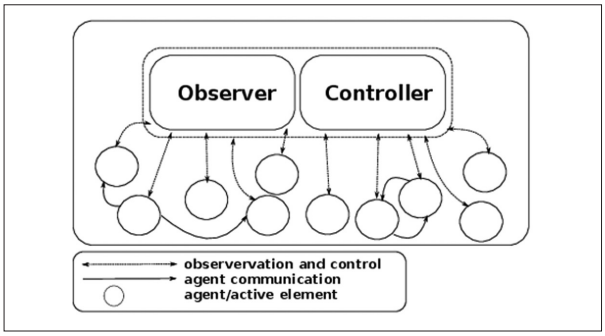
1- Control mechanisms
2- Closed-loop
3- Feed-back mechanism

مـی‌رود که در بسـیاری از شـاخه‌ها به‌کار رفته اسـت [۲, ۵-۱۰]. در ایـن سـازوکار کنترلی، از رفتار سـامانه به‌عنوان منبع اطلاعات اثرگذار یا تصحیح‌کننده رفتار آتی سـامانه بهره برده می‌شود [۵]. این سـازوکار از دولایه مشخص تشکیل می‌شود: یک لایه پایین که عملکرد سامانه را محقق می‌سـازد و یک لایه بالاتر که سـامانه را هدایت می‌نماید [۳].

درحالی‌که نمونه سـازوکارهای بازخوردی سـاخت دست بشـر تنها به چند قرن پیش بازمی‌گردد، در طبیعت از دیرباز چنین سـازوکارهایی را می‌توان مشاهده نمود [۱۱, ۱۲]. به‌عنـوان نمونه می‌توان از حلقه‌های بازخوردی مولکولی[4] [۱۳] و دیگر مسـیرهای حیاتی مثل سیگنال‌های کنترلـی پروتئولیز[5] [۱۴] در موجودات ابتدایی نام برد. این سازوکارها بیشتر مبتنی بر واکنش‌های شیمیایی هستند تا این‌که هوشمندی در آن‌ها دخیل باشد [۱۵]. سازوکارهای پیچیده‌تـر کنترلـی را می‌توان در موجـودات زنده دارای دسـتگاه عصبی مرکزی پیدا کرد. این دسـتگاه مسئولیت رفتار، تعامل و پاسـخ موجود زنـده را در قبال محیط بر عهـده دارد [۱۶-۱۸]. دسـتگاه عصبی مرکـزی اطلاعات جمع‌آوری شـده از طریـق حواس را پـردازش نموده و پاسخ‌های مناسب به محیط را تولید می‌کند [۱۹].

با الهام گرفتن از چنین سـازوکارهایی است که الگوی ناظر/کنترل‌کننده[6] جهت تحقق سـازوکار کنترلی پیشنهاد شد [۶, ۲۰]. این الگو به شکل موفقی در بسیاری از حوزه‌ها اسـتفاده شده است [۶]. شـکل ۱: شمای کلی الگوی ناظر/کنترل‌کننـده یک نما از الگوی ناظر/کنترل‌کننده را نشـان می‌دهد [۲۰]. لایه زیرین یا سـامانه تحت نظارت و کنترل[7]، توسط لایه بالاتر یعنی لایه ناظر/کنترل‌کننده هدایت می‌شود [۲۰]. عامل‌ها یا عناصر فعال نیز هم با ناظر/کنترل‌کننده و هم با یکدیگر تعامل دارند.

در الگوی ناظر/کنترل‌کننـده، ناظر مؤلفه‌ها را در قالب



شکل ۱: شمای کلی الگوی ناظر/کنترل‌کننده

ویژگی‌های تعریف‌شـده (همانند در دسـترس بودن یک عملکرد)، مورد پایش قرار می‌دهد. سـپس این اطلاعات را به شکلی تجمیع می‌نماید که قابل‌استفاده کنترل‌کننده باشد. این الگو همانند بسـیاری از سازوکارهای کنترلی، سامانه را از طریـق بازخورد مثبت و منفی هدایـت می‌کند [۲۱]. بـا وجود فواید الگوی ناظر/کنترل‌کننـده به‌عنوان یک نوع سازوکار بازخوردی، مولفه‌های ناظر/کنترل‌کننده می‌تواند به نقطه شکست سامانه مبدل گردد. شکست در مؤلفه‌های سـازنده ناظر/کنترل‌کننـده بـه دلیل خطاهـای داخلی یا تغییرات پیش‌بینی‌نشده محیطی، به عدم کنترل کل سامانه یا عدم امکان فعالیت آن می‌انجامد. تاکنون در خصوص این مشکل در این الگو مطالعه‌ای صورت نگرفته است.

از این رو، یک راه‌حل مؤثر، ایجاد ویژگی خود-ترمیمی[8] در سـازوکار کنترلی است. یک سـامانه زمانی در برابر کنش‌های خارجی، خود-ترمیم خوانده می‌شـود که بتواند در قبال آن‌ها ویژگی امن[9] بودن را حفظ نموده و یا فقط به شکل موقتی آن را از دست داده و مجددا بازیابی نماید [۹]. این ویژگی، سامانه‌های حسـاس به ایمنی[10] را از خطرات حفظ می‌کند. برای ایجاد این ویژگی در سـازوکار کنترلی، این مقاله با الهام از زیست‌شناسی، از موجودات زنده دارای قابلیت ترمیم دسـتگاه عصبی مرکزی ایده گرفته اسـت. دلیل این تصمیم شباهت میان عملکرد سازوکار کنترلی و دستگاه عصبی مرکزی در جمع‌آوری اطلاعات و پردازش آن‌ها در پاسخ به محیط است. در طبیعت موجوداتی وجود

4- Molecular feedback loops
5- Proteolysis control signal
6- Observer/Controller
7- System under Observation and Control (SuOC)
8- Self-healing
9- Safe
10- Safety-Critical

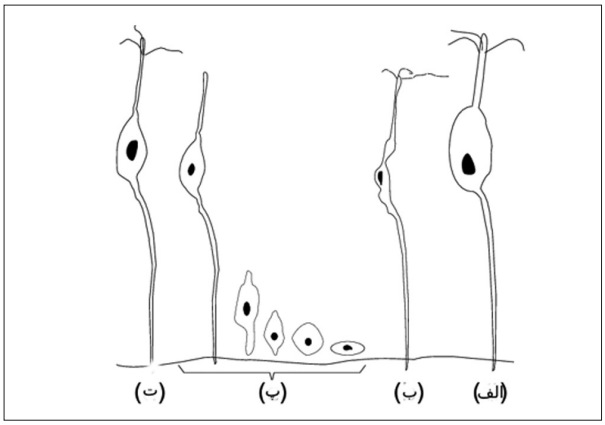
دارند که می‌توانند دستگاه عصبی مرکزی خود را تا حد قابل‌توجهی ترمیم نمایند. به‌بیان‌دیگر، آن‌ها ویژگی خود-ترمیم را در سطح سازوکار کنترلی دارا هستند.

در بخش بعدی، پیش‌زمینه کار ارائه داده‌شده است تا یک فهم مشترک از مفاهیم مورد نیاز از مقاله ایجاد گردد. بخش سوم به کارهای مرتبط می‌پردازد. بخش چهارم، ایده‌های قابل استفاده از پدیده زیستی را در قالب اصولی بیان می‌دارد. الگوی پیشنهادی در بخش پنجم بابیان مفاهیم اصلی مطرح می‌گردد. جزئیات الگوی پیشنهادی، تغییرناپذیر[11]های سامانه و توصیف و درستی‌یابی آن در بخش ششم آمده است. کاربردپذیری الگو با کمک مطالعه موردی در بخش هفتم آمده است. بخش آخر نتیجه‌گیری و کارهای آتی را در بر می‌گیرد.

۲- پیش‌زمینه

این بخش برخی اطلاعات پیش‌زمینه‌ای در خصوص پدیده زیستی مورد الهام را برای فهم بیشتر مفاهیم الگوی پیشنهادی بیان می‌دارد. پدیده مورداستفاده «ترمیم دستگاه عصبی مرکزی» در گورخرماهی[12] است. این نوع ماهی، یک مدل شناخته‌شده زیست‌شناسی به شمار می‌رود. برای این‌که از جزئیات زیاد زیستی پرهیز شود، این بخش فقط به روشن کردن جنبه‌های کلیدی این پدیده می‌پردازد.

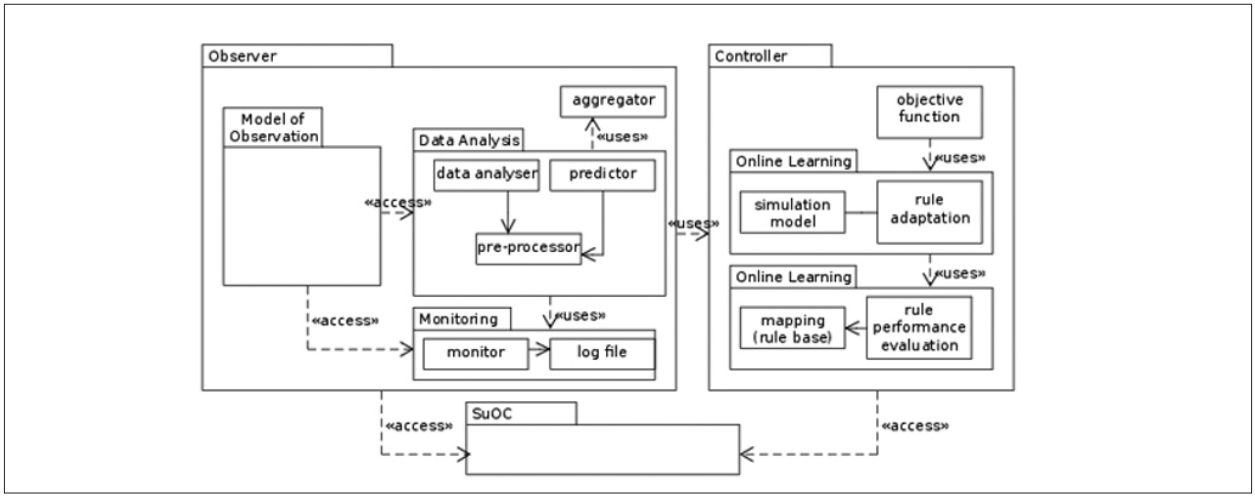
تولیدمثل عالی، ساختار جنینی ساده، گورخرماهی را در مطالعات مربوط به تکامل جنینی مهره‌داران بر موش برتری داده است [۲۲]. گورخرماهی ویژگی‌های ترمیمی جالبی در باله‌ها، عضلات قلب و دستگاه عصبی از خود نشان می‌دهد [۲۳]. فرایند ترمیم در دستگاه عصبی، شامل جایگزینی نورون‌های صدمه‌دیده توسط نورون‌های ایجادشده از نوع خاصی از سلول‌ها به نام سلول‌های بنیادین می‌شود. یکی از ابتدایی‌ترین نمونه‌های ترمیم دستگاه عصبی در روپوشه بویایی[13] رخ می‌دهد که یک



شکل ۲: ترمیم نورون در روپوشه بویایی [۲۶]. از راست به چپ: (الف) سالم (ب) صدمه‌دیده (پ) در حال ترمیم (ت) سالم

مدل برای از بین رفتن و ترمیم نورون‌ها ایجاد می‌کند [۲۴]. در این نوع از ترمیم وقتی یک نورون می‌میرد، با یک سلول بنیادی از سلول‌های پایه‌ای در منطقه روپوشه بویایی جایگزین می‌گردد [۲۵]. بدین صورت سلول بنیادی تمایزیافته و جایگزین نورون می‌گردد (شکل ۲: ترمیم نورون در روپوشه بویایی [۲۶]. از راست به چپ: (الف) سالم (ب) صدمه‌دیده (پ) در حال ترمیم (ت) سالمشکل ۲). فرایند بدین شکل رخ می‌دهد: وقتی‌که نورون سالم است شکل ۲: ترمیم نورون در روپوشه بویایی [۲۶]. از راست به چپ: (الف) سالم (ب) صدمه‌دیده (پ) در حال ترمیم (ت) سالم(شکل ۲-الف) و صدمه می‌بیندشکل ۲: ترمیم نورون در روپوشه بویایی [۲۶]. از راست به چپ: (الف) سالم (ب) صدمه‌دیده (پ) در حال ترمیم (ت) سالم (شکل ۲-ب)، یک سلول بنیادی شروع به جایگزینی آن می‌کنشکل ۲: ترمیم نورون در روپوشه بویایی [۲۶]. از راست به چپ: (الف) سالم (ب) صدمه‌دیده (پ) در حال ترمیم (ت) سالمد (شکل ۲-پ). هنگامی‌که فرایند تمام شود، نورون صدمه‌دیده به شکل کامل جایگزین می‌گرشکل ۲: ترمیم نورون در روپوشه بویایی [۲۶]. از راست به چپ: (الف) سالم (ب) صدمه‌دیده (پ) در حال ترمیم (ت) سالمدد (شکل ۲-ت). درزمانی که تعداد نورون‌های صدمه‌دیده زیاد می‌شود، فرایند به شکل محسوسی سرعت بیشتری به خود می‌گیرد.

11- Invariant
12- Zebrafish
13- Olfactory epithelium



شکل ۳: نمایش ساده‌شده معماری عمومی ناظر/کنترل‌کننده

۳- کارهای مرتبط

ایده اضافه کردن خود-ترمیمی در سطح الگوی ناظر/کنترل‌کننده جدید بوده و در جستجوی سامانمند هیچ کار مرتبط مستقیمی یافت نشد. ازاین‌رو، پژوهش‌های مرتبط با این الگو و گونه‌های مختلف آن از نظر خواهد گذشت.

ریختر[۱۴] و دیگران [۲۰] معماری عمومی ناظر/کنترل‌کننده را به‌عنوان شناخته‌شده‌ترین پژوهش در خصوص این الگو مطرح کرده‌اند. پژوهشگران دیگر این معماری را برای نیازهای خود پالایش نموده‌اند (همانند [۲۷, ۲۸شکل ۳: نمایش ساده‌شده معماری عمومی ناظر/کنترل‌کنندهشکل ۳ نمایش ساده‌شده معماری عمومی ناظر/کنترل‌کننده را نشان می‌دهد. مؤلفه «پایشگر[۱۵]» وضعیت عامل‌ها، منابع و خطاها را پایش می‌کند و آن را در «فایل سابقه[۱۶]» ثبت می‌نماید. مؤلفه «پیش‌پردازنده[۱۷]» داده‌ها را برای سایر زیرمؤلفه‌های ناظر آماده می‌سازد. مؤلفه «پیش‌بینی‌کننده[۱۸]» وضعیت بعدی منابع یا سامانه را تخمین می‌زند. «تحلیل‌گر داده[۱۹]» داده‌های جمع‌آوری‌شده توسط پایشگر را تحلیل می‌نماید. در نهایت، مؤلفه «تجمیع کننده[۲۰]»، داده‌ها را از مؤلفه‌های پایشگر، تحلیل‌گر داده و پیش‌بینی کننده تجمیع کرده و به کنترل‌کننده می‌فرستد.

بخش کنترل‌کننده مسئولیت تصمیم‌گیری و اجرای تصمیم‌ها را برعهده دارد. تصمیم‌های مزبور توسط زیرمؤلفه‌های «یادگیری برخط[۲۱]» و «یادگیری برون‌خط[۲۲]» انجام می‌گیرد. اطلاعات مورد نیاز توسط «تجمیع کننده[۲۳]» به کنترل‌کننده رفته و به وسیله زیرمؤلفه «نگاشت[۲۴]» (یا پایگاه قوانین) و «ارزیابی کارایی قانون[۲۵]» در کنترل‌کننده استفاده می‌گردد. زیرمؤلفه «ارزیابی کارایی قوانین» برای یادگیری برخط است که قوانین تصمیم‌گیری را بروز می‌سازد. از سوی دیگر، مؤلفه «تطبیق قانون[۲۶]» و «مدل شبیه‌سازی[۲۷]» زیرمؤلفه‌هایی هستند که قوانین جدید را ایجاد و قوانین قدیمی را حذف می‌کنند و عهده‌دار یادگیری برون‌خط هستند. «تابع هدف[۲۸]» مؤلفه‌ای است که تعامل با کاربر/راهبر سامانه را بر عهده دارد و سامانه را پس از دریافت هدف ارزیابی می‌نماید. در نهایت «مدل نظارت[۲۹]» توسط کنترل‌کننده انتخاب می‌شود تا روش نظارت مناسب و پارامترهای پایش (همانند نرخ نمونه‌برداری) را تعیین نماید. تمام پایش‌های سامانه بر اساس این مدل انجام می‌گیرد.

14- Richter
15- Monitor
16- Log file
17- Pre-processor
18- Predictor
19- Data analyzer
20- Aggregator

21- Online learning
22- Offline learning
23- Aggregator
24- Mapping
25- Rule performance evaluation
26- Rule adaptation
27- Simulation model
28- Objective function
29- Observation model

نفز[30] و دیگران [۲۷] معماری عمومی ناظر/کنترل‌کننده را با روشـــی به نام «روش تغییرناپذیر بازیابی[31]» پالایش می‌کنند. در این روش، اهداف ســـامانه از طریق حفاظت از تغییرناپذیرها و تصحیح رفتار ســـامانه در صورت عدول از تغییرناپذیـــر انجام می‌گیرد. در این روش بخش ناظر از مؤلفه «پایشگر تغییرناپذیر[32]» تشکیل‌شده که وظیفه پایش ســـامانه را برای تشـــخیص نقض تغییرناپذیرها بر عهده دارد. بخش کنترل‌کننده مشتمل بر دو مؤلفه «الگوریتم‌های بازپیکربندی[33]» و «کنترل‌کننده نتیجه[34]» است که به ترتیب وظیفـــه پیدا کردن راه‌حل‌های ممکـــن و اطمینان از صحت راه‌حـــل در تأمیـــن تغییرناپذیر را بر عهـــده دارند. در یک تطبیق راه‌حل‌ها تاجایی مورد امتحان قرار می‌گیرند که در نهایت یک راه‌حل موفق حاصل گردد.

راث[35] و دیگـــران [۲۸] از معمـــاری عمومـــی ناظـــر/کنترل‌کننده برای یک میان‌افزار اســـتفاده برده‌اند. معماری این میان‌افزار مشـــابه معماری چرخـــه MAPE [۲۹] بوده و ویژگی‌هـــای خود-ترمیمـــی، خود-حفاظتـــی[36]، خود-پیکربندی[37] و خود-بهینگی[38] را برای ســـرویس‌های سوار بر میان‌افزار ایجاد می‌نماید. هر سرویس سوار شده بر این میان‌افزار، باید یک واسط را جهت استفاده از این ویژگی‌ها ایجاد کرده باشـــد. مؤلفه‌های ایجادکننـــده خود-* در این معماری به شـــکل مســـتقل عمل کرده و لذا ممکن است که نتایج متناقض گردند.

در کار قبلـــی نگارنـــدگان ایـــن مقاله [۳۰]، بـــا تکیه‌بر معماری عمومی ناظر/کنترل‌کننده و تمرکز بر گونه خاصی از سامانه‌های اســـتفاده‌کننده از این الگوی معماری، با نام ســـامانه‌های پردازش کننده جریان منابع[39]، یک ســـامانه به‌عنوان مجموعه‌ای از عامل‌ها در نظر گرفته شد. بر اساس این تعریف، یک معماری پشـــتیبانی‌کننده از دو نوع قابلیت ارائه شد. اولین نوع این قابلیت‌ها، «قابلیت کاری[40]» نام دارد که کارهای بخش تحت نظـــارت و کنترل (همانند پردازش داده) را انجام می‌دهد. نـــوع دیگر قابلیت‌ها، «قابلیت ناظر/کنترل‌کننده» نام گرفت که این نوع از قابلیت‌ها وظیفه انجام دادن عملیات نظارت و کنترل را عهده‌دار بودند.

30- Nafz
31- Restore invariant approach
32- Invariant monitor
33- Reconfiguration algorithms
34- Result checker
35- Roth
36- self-protection
37- self-configuration
38- self-optimization
39- resource-flow systems
40- task capabilities

۴- ایده‌های الهام گرفته‌شده از زیست‌شناسی

در این بخش ایده‌هایی را که از پدیده زیستی (بخش ۲) الهام گرفته شـــده در قالب اصولـــی بیان می‌گردد. این کار باعث عدم نیاز رجوع به جزئیات زیست‌شناسی می‌شود.

اصل یک. ســـازوکار کنترلی یک ســـامانه باید با کمک مؤلفه‌های خود آن ســـامانه ایجاد گردد. به‌بیان‌دیگر، هیچ عنصری دیگـــری نباید بر ایجاد ســـازوکار کنترل دخالت داشته باشد.

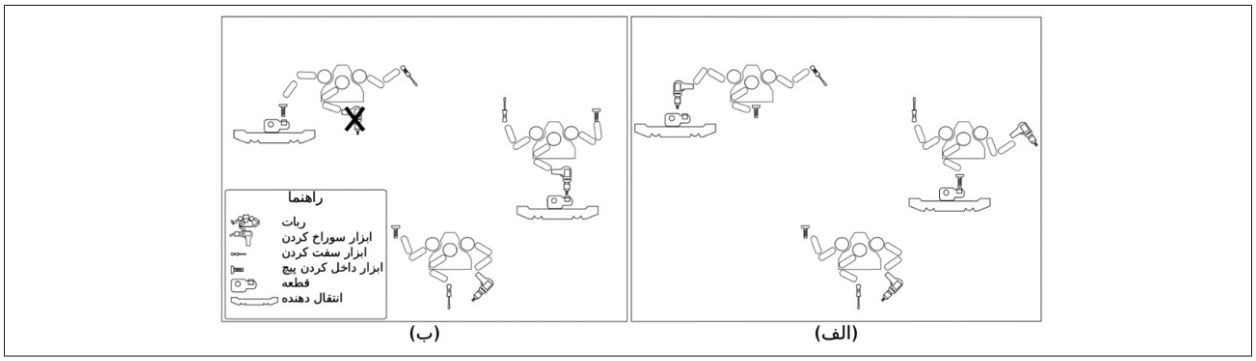
عناصر ســـامانه باید قابلیت کنترلی مناســـب را جهت تشکیل ســـازوکار کنترلی داشته باشند. مثلاً یک عنصر با قابلیت جمـــع‌آوری اطلاعات می‌تواند در بخش ناظر به‌کار گرفته شود.

اصل دو. مؤلفه‌های سازوکار کنترلی به هم وابستگی‌هایی دارند. یعنی برای فعال شدن یک عملکرد سازوکار کنترلی، تعدادی از مؤلفه‌ها باید موجود و فعال باشند.

اصل دو بر وابســـتگی بین مؤلفه‌های سازوکار کنترلی تأکید دارد. وابســـتگی معادل با نیـــاز برای یک نوع خاص ســـلول یا سلول بنیادی جهت انجام ترمیم است. سازوکار کنترلی از مؤلفه‌های متعددی تشکیل‌شـــده که برای ایجاد سازوکار کنترلی مورد استفاده قرار می‌گیرند. به‌بیان‌دیگر، سازوکار کنترلی بدون کلیه مؤلفه‌های وابسته کار نخواهد کرد. درصورتی‌که مؤلفه‌ای دچار مشـــکل شـــده باشـــد، ســـازوکار کنترلی باید متوقف شـــود و به خود-ترمیمی بپردازد.

اصل ســـه. عناصر ســـامانه باید ویژگی‌های مشترکی جهت امکان نظارت و کنترل داشته باشند.

در گورخرماهی و دیگر موجودات زنده، دستگاه عصبی



شکل ۴: نمای شماتیک مطالعه موردی در عملیات عادی (الف). وقتی یکی از ابزارها (ابزار سوراخ کردن در روبات سمت چپ) شکسته شده، و بازپیکربندی انجام شده است (ب).

مرکزی می‌تواند بر سلول‌های بافت‌ها، نظارت و کنترل داشته باشد. این اصل نشان می‌دهد که مؤلفه‌ها باید از منظر نظارت و کنترل همگن باشند. مثلاً همه عناصر واسط‌های یکسانی جهت امکان نظارت و کنترل داشته باشند.

اصل چهـــار. اگر یک مؤلفـــه ســـازوکار کنترلی دچار شکست گردد، مؤلفه‌های دیگری باید جایگزین آن گردند.

در ارتبـــاط با گورخرماهـــی، باید یک ســـلول بنیادی جایگزین نورون از بین رفته شود. لذا، عناصری از سامانه باید بتوانند هرکدام از قابلیت‌های موردنیاز کنترل را اجرا نمایند.

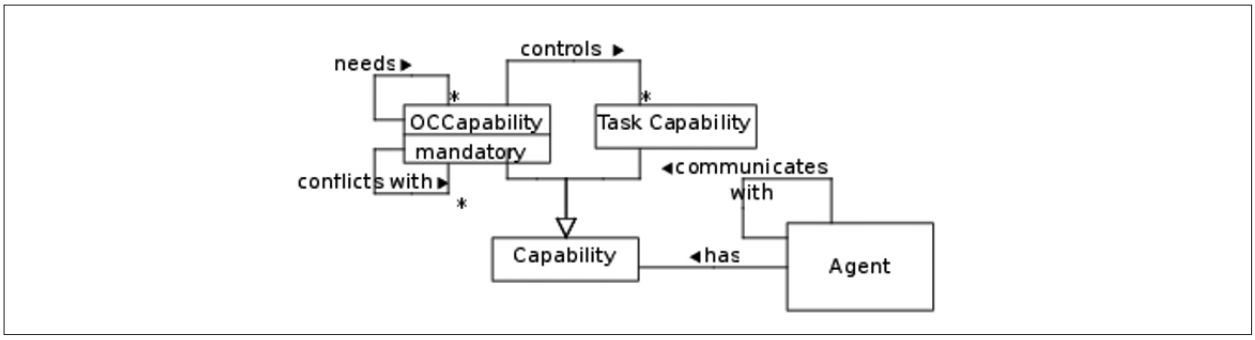
۵- الگوی ناظر/کنترل‌کننده خود-ترمیم

بر اساس اصول گفته‌شده در بخش چهار، در این بخش الگوی ناظر/کنترل‌کننده خود-ترمیم ارائه می‌گردد. این الگو به مشکل خود-ترمیمی در سازوکار کنترلی می‌پردازد. در این جا سعی می‌شود منطق تصمیم‌های اخذشده بر اساس اصول بخش چهار تبیین گردد. این بخش به چند زیر بخش تقسیم شده است. به جهت توضیح بیشتر، اولین زیر بخش به یک مثال اختصاص دارد که در ادامه مقاله مورد استفاده قـــرار خواهد گرفـــت. در زیر بخـــش دوم، فرا-مدل الگوی پیشنهادی ارائه می‌گردد. سومین و چهارمین زیر بخش به توضیح جنبه‌های اساسی این الگو می‌پردازند.

۱.۵ مثال

هدف این است که با استفاده از یک مطالعه موردی، که در بسیاری از کارهای مرتبط استفاده شده است، اثر پدیده ترمیم دستگاه عصبی مرکزی روی الگوی ناظر/کنترل‌کننده نشـــان داده شـــود. این مثال، که در سراسر مقاله استفاده خواهد شد، یک نمونه از سامانه‌ها موسوم به سامانه‌های خود-ســـازمانده جریان منابع[41] است [۲۷, ۳۱, ۳۲]. در این سامانه‌ها، منابعی (قطعات) توســـط عامل‌هایی (روبات‌ها) پـــردازش می‌شـــوند. ایـــن قطعات توســـط دســـتگاه‌های خودکاری بین روبات‌ها جابجا می‌گردند. طبیعت پردازش فیزیکی است [۲۷, ۳۱, ۳۲] که شامل سه کار سوراخ کردن قطعه، واردکردن پیچ و ســـفت کردن پیچ در قطعه می‌شکل ٤: نمای شـــماتیک مطالعه موردی در عملیات عادی (الف). وقتی یکی از ابزارها (ابزار سوراخ کردن در روبات سمت چپ) شکسته شـــده، و بازپیکربندی انجام شده است (ب). شود (شکل ٤ الف). ابزارهای مورد استفاده ممکن است کار نکنند و در نتیجه، روبات بخشی یا تمام عملکرد خود را از دست بدهد. همچنین ابزارها قابلیت خاموش/روشن کردن دارنـــد و در هنگام خرابی می‌توان ابـــزار فعال هر روبات را تغییـــر داد. هدف، پردازش تمام قطعـــات حتی با وجود شکست در ابزارها اســـت. این کار با تغییر ابزارهای فعال روبات‌ها انجام می‌پشکل ٤: نمای شماتیک مطالعه موردی در عملیات عادی (الف). وقتی یکی از ابزارها (ابزار سوراخ کردن در روبات سمت چپ) شکسته شده، و بازپیکربندی انجام شده است (ب). شکل ٤-ب یک نمونه از بازپیکربندی را در هنگام خرابی ابزار سوراخ کردن نشان می‌دهد. ممکن

41- Self-organizing resource-flow systems



شکل ۵: فرا-مدل الگوی ناظر/کنترل‌کننده خود-ترمیم

است در حالت‌هایی هیچ تغییری نتواند به ادامه پردازش بینجامد.

۲.۵ فرا-مدل

اکثر کارهای مرتبط با الگوی ناظر/کنترل‌کننده، معماری عمومی ناظر/کنترل‌کننده را پالایش کرده‌اند. به دلیل تنوع کارهای موجود، یک فرا-مدل برای الگوی پیشنهادی ایجادشده، تا معماری عمومی ناظر/کنترل‌کننده و گونه‌های مختلف آن را پشتیبانی کند. چنین فرا-مدلی الگوی پیشنهادی را برای استفاده در کارهای مرتبط کارا می‌نماید. از همین‌رو در این‌جا یک فرا-مدل بر اساس پژوهش [۳۰] پیشنهاد میشکل ۵: فرا-مدل الگوی ناظر/کنترل‌کننده خود-ترمیم‌شود (شکل ۵). در فرا-مدل پیشنهادی، هر سامانه مجموعه‌ای از عامل‌ها است که دو نوع قابلیت «قابلیت ناظر/کنترل‌کننده» و «قابلیت کاری» را دارا هستند. این امر اصل اول را محقق می‌سازد که در آن قابلیت‌های ناظر/کنترل‌کننده، به سازوکار کنترلی مرتبط هستند.

فرا-مدل داده‌شده دارای دو نوع قابلیت است که توسط رابطه عمومی‌سازی[42] در زبان UML از یک مفهوم ارث‌بری نموده تا بیانگر نگاه واحد به قابلیت‌ها باشد. قابلیت‌های ناظر/کنترل‌کننده مسئولیت هدایت قابلیت‌های کاری را دارند (همانند تغییر ابزار در مثال). رابطه‌های نیاز[43] و تضاد[44] نیز نشان‌دهنده رابطه میان قابلیت‌های ناظر/کنترل‌کننده و توسعه‌یافته اصل دو از بخش ٤ است.

رابطه نیاز برای نشان دادن وابستگی یک قابلیت ناظر/کنترل‌کننده به قابلیت ناظر/کنترل‌کننده دیگر استفاده می‌شود (همانند وابستگی «الگوریتم‌های بازپیکربندی» به «پایش تغییرناپذیر» در بخش کنترل‌کننده «روش بازیابی تغییرناپذیر» در بخش ۳). رابطه تضاد نشان‌دهنده رابطه‌ای است که طبق آن، دو قابلیت هم‌زمان نمی‌توانند در سامانه وجود داشته باشند. رابطه سومی به نام اجباری وجود دارد که به الزامی بودن یک قابلیت در سامانه اشاره دارد که در ردهٔ Capability شکل ٥ آمده است. فرا-مدل این توانایی را دارد که از گونه‌های متمرکز، نامتمرکز و چند-لایه پشتیبانی شکل ٦: سه گونه معماری عمومی ناظر/کنترل‌کننده با استفاده از فرا-مدل پیشنهادی: (الف) متمرکز، (ب) چند-لایه، (پ) نامتمرکزنماید (شکل ٦-الف.، ب. و پ.).

۳.۵ روابط میان قابلیت‌ها

معماری عمومی ناظر/کنترل‌کننده را می‌توان از ویژگی‌ها و روابط میان آن‌ها، آن‌گونه که در جدول ۱ دیده می‌شود، محقق ساخت. ستون قابلیت موردنیاز در این جدول نشان‌دهنده قابلیت یا قابلیت‌هایی است که قابلیت مزبور به آن‌ها نیاز دارد. لازم به ذکر است که رابطه تضاد در این معماری یافت نشد.جدول ۱: خلاصه قابلیت‌های ناظر/کنترل‌کننده استفاده‌شده در الگوی پیشنهادی برای پشتیبانی از معماری عمومی ناظر/کنترل‌کننده (ا = اجباری؛ ن = ناظر؛ ک = کنترل‌کننده). از آن‌جایی‌که همه قابلیت‌ها اجباری هستند، ستونی برای آن در نظر گرفته نشده است.

42- generalization
43- Needs
44- Conflicts with

جدول ۱: خلاصه قابلیت‌های ناظر/کنترل‌کننده استفاده‌شده در الگوی پیشنهادی برای پشتیبانی از معماری عمومی ناظر/کنترل‌کننده (ا = اجباری؛ ن = ناظر؛ ک = کنترل‌کننده). از آن‌جایی‌که همه قابلیت‌ها اجباری هستند، ستونی برای آن در نظر گرفته نشده است.

قابلیت	معادل در جدول ۱	نوع	توضیح	قابلیت موردنیاز
Monitoring (پایش)	پایشگر	ن-ا	وضعیت عامل‌ها، منابع و خطاها را تشخیص می‌دهد.	ندارد
Prediction (پیش‌بینی)	پیش‌بینی کننده	ن	وضعیت بعدی سامانه یا منابع را پیش‌بینی می‌نماید.	Monitoring
Simulation (شبیه‌سازی)	مدل شبیه‌سازی	ک	نتایج را شبیه‌سازی نموده تا در یادگیری برون‌خط به‌کار رود.	ندارد
Analyzing (تحلیل)	تحلیل‌گر داده	ن	داده‌های جمع‌آوری‌شده را تحلیل می‌کند.	Monitoring
Aggregation (تجمیع)	تجمیع کننده	ن-ا	داده‌ها را از منابع مختلف در ناظر تجمیع می‌کند.	Monitoring, Analyzing, Prediction
Mapping (نگاشت)	نگاشت	ک-ا	قوانین را با توجه به شرایط اعمال می‌کند.	Aggregation
User Control (کنترل کاربر)	تابع هدف	ک	نمایش‌دهنده اهداف کاربر و ارزیابی اهداف است.	Rule Adaptation
Rule Performance Evaluation (ارزیابی کارایی قانون)	ارزیابی کارایی قانون	ک	عهده‌دار انجام آموزش بر خط.	Mapping
Rule Adaptation (تطبیق قانون)	تطبیق قوانین	ک	تغییر قوانین در کنترل‌کننده را عهده‌دار است.	Mapping
OC Model Management (مدیر مدل)	ندارد	ک-ا	یک مدل کلی از سامانه را نگهداری می‌کند و تطابق وضعیت فعلی سازوکار کنترل در حال اجرا را بر عهده دارد.	ندارد

به کمک جدول ۱، الگوی پیشنهادی قابلیت ترمیم شکست را در سازوکار کنترل برعهده دارد.

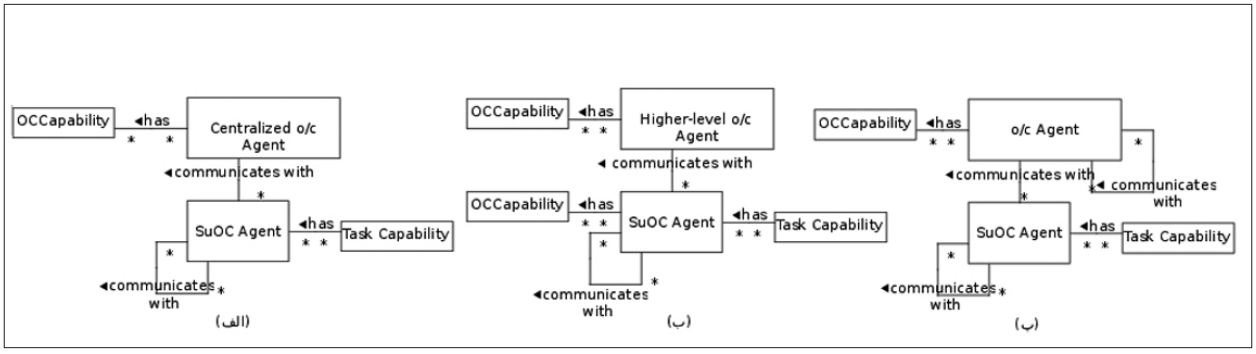
تفاوت اصلی میان معماری عمومی ناظر/کنترل‌کننده و قابلیت‌های الگوی پیشنهادی در قابلیت مدیر مدل است. این قابلیت، عملکرد مؤلفه «مدل نظارت» را در معماری عمومی توسعه می‌دهد. این کار از طریق نگهداری یک مدل به نام مدل ناظر/کنترل‌کننده از کل سامانه و مشخص کردن لیست قابلیت‌های ناظر/کنترل‌کننده‌ای که باید فعال باشند صورت می‌پذیرد. از همین رو، این قابلیت برای ایجاد تطبیق‌پذیری در سطح سازوکار کنترلی حیاتی است و از همین رو جزء قابلیت‌های اجباری در نظر گرفته شده است. جدول ۲ رابطه بین قابلیت‌های ناظر/کنترل‌کننده در مثال استفاده شده در این مقاله را با سازوکار کنترلی روش بازنشانی تغییرناپذیر، نشان می‌دهد.

با ترکیب فرا-مدل پیشنهادی و روابط میان قابلیت‌های ناظر/کنترل‌کننده در جدول ۲، شکل ۷ یک نمودار شیء

جدول ۲: روابط میان قابلیت‌های ناظر/کنترل‌کننده در مثال

قابلیت	نوع رابطه	قابلیت دیگر
الگوریتم‌های بازپیکربندی	نیاز	پایش تغییرناپذیر
امتحان نتیجه	نیاز	الگوریتم‌های بازپیکربندی

در خصوص نسخه متمرکز سازوکار کنترلی «روش بازنشانی تغییرناپذیر» را برای مثال استفاده شده نشان می‌دهد. همچنین، به دلیل استفاده از نوع متمرکز معماری، تمام قابلیت‌های ناظر/کنترل‌کننده روی یک عامل (robot2) قرار دارند که همان ناظر/کنترل‌کنشکل ٦: سه گونه معماری عمومی ناظر/کنترل‌کننده با استفاده از فرا-مدل پیشنهادی: (الف) متمرکز، (ب) چند-لایه، (پ) نامتمرکزنده مرکزی (شکل ٦) است. رابطه نیاز بر اساس جدول ۲ تخصیص یافته و چون همه مؤلفه‌های ناظر/کنترل‌کننده الزامی هستند، ویژگی اجباری برای همه آن‌ها درست در



شکل ۶: سه گونه معماری عمومی ناظر/کنترل‌کننده با استفاده از فرا-مدل پیشنهادی: (الف) متمرکز، (ب) چند-لایه، (پ) نامتمرکز

نظر گرفته شده‌است. همچنین یک قابلیت کاری فقط وقتی فعال باشد، ویژگی active آن مقداردهی شده است.

٤.٥ نگاه کلی به عملیات سامانه

عملیات سامانه‌ای که از الگوی پیشنهادی استفاده می‌کند، قابل تجرید به یک ماشین حالت است (شکل ۸). در این ماشین حالت، دو حالت سکون[45] وجود دارد [۲۷]. یکی از این حالت‌ها برای تطبیق در سامانه تحت نظارت و کنترل و دیگری برای تطبیق‌پذیری ناظر/کنترل‌کننده در نظر گرفته‌شده است. در حالت‌های سکون، سامانه فقط عملیاتی را انجام می‌دهد که وضعیت سامانه را پایدار کند. این موضوع با اصل دو از بخش ٤ تطابق دارد.

حالت ابتدایی سامانه. بر اساس اصل یک از بخش ٤، عامل‌ها باید سازوکار کنترلی را تشکیل دهند. لذا وقتی عملیات آغاز می‌شود، مدل ناظر/کنترل‌کننده برای تمام عامل‌ها ارسال می‌شود. لذا هر عامل از قابلیت‌های ناظر/کنترل موردنیاز اطلاع پیدا می‌کند. در این وضعیت، یک الگوریتم انتخاب، اجرا و در هر عامل مناسب‌ترین قابلیت(های) ممکن ناظر/کنترل‌کننده مشخص می‌گردد. هدف این مرحله رسیدن به سامانه‌ای است که با مدل تطابق داشته باشد. اگر این مرحله با شکست مواجه شود، باید سامانه غیرفعال بماند.

عملیات عادی. در عملیات عادی، عامل‌ها کارهای محوله را انجام می‌دهند. ممکن است که این کار، انجام عملیاتی روی منابع (مثل سوراخ کردن در مثال این مقاله) یا این‌که کار ساده‌ای مثل جمع‌آوری اطلاعات در یک شبکه حسگر بی‌سیم باشد [۳۳].

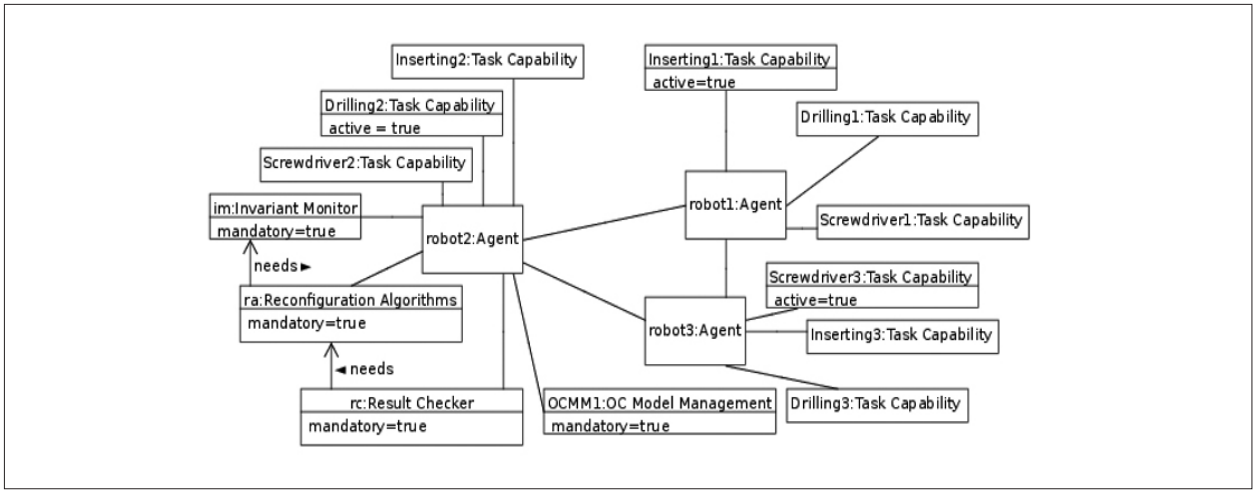
تطبیق‌پذیری. در دو سطح تطبیق‌پذیری رخ می‌دهد: ۱) در سطح سامانه تحت نظارت و کنترل، از طریق سازوکارهایی مانند بازپیکربندی، تطبیق‌پذیری انجام می‌گیرد (شکل ٤: نمای شماتیک مطالعه موردی در عملیات عادی (الف). وقتی یکی از ابزارها (ابزار سوراخ کردن در روبات سمت چپ) شکسته شده، و بازپیکربندی انجام شده است (ب).شکل ٤-ب). ۲) در سطح ناظر/کنترل‌کننده (یعنی در مؤلفه‌های ناظر/کنترل‌کننده)، بر اساس اصل چهار از بخش ٤، روش الهام‌یافته از طریق ترمیم دستگاه عصبی اعمال می‌شود.

الگوی پیشنهادی، تطبیق‌پذیری در سطح ناظر/کنترل‌کننده را با استفاده از خود-ترمیمی ایجاد می‌نماید. لازم به ذکر است که در برخی از مراحل هیچ تطبیقی ممکن است یافت نشود که بتواند سازگاری با مدل را ایجاد نماید (یعنی مجموعه پیکربندی‌ها تهی گردد) [۳٤]. برای نمونه، ممکن است که کلیه مصداق‌های یک مؤلفه ناظر/کنترل‌کننده در دسترس نبوده یا دچار شکست شده باشند (شامل کلیه مؤلفه‌هایی که آن قابلیت ناظر/کنترل‌کننده را، حتی اگر غیرفعال است، دارند).

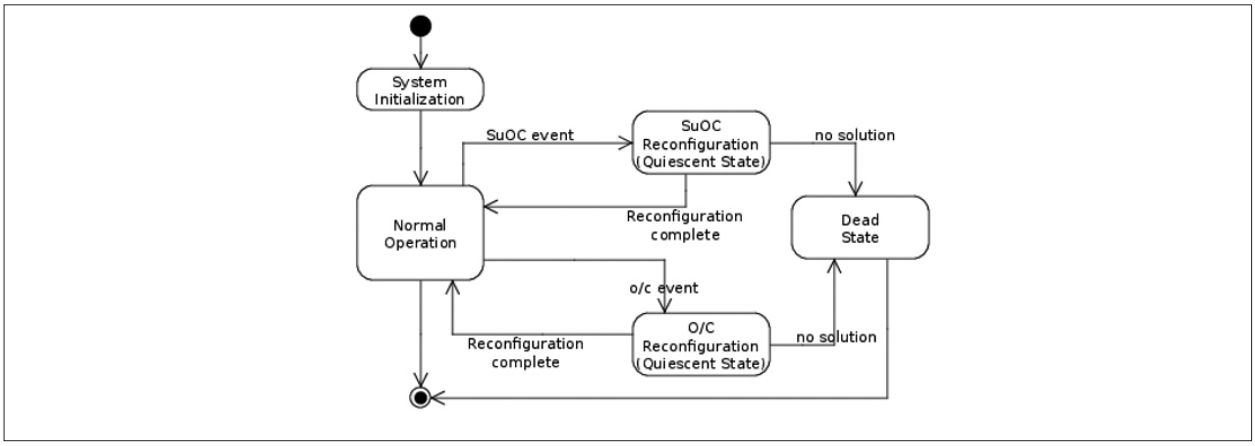
خود-ترمیمی در الگوی پیشنهادی را می‌توان از طریق تاکتیک معماری اسپیر[46] محقق کرد [۳۵] که بر وجود مؤلفه‌های اضافی و فعال شدن آن‌ها در هنگام

45- Quiescent

46- Spare



شکل ۷: مدل مشتق شده برای مثال با استفاده از فرا-مدل پیشنهادی



شکل ۸: عملیات سامانه با استفاده از الگوی پیشنهادی

نیاز تکیه دارد. سازوکار خود-ترمیمی تنها وقتی لازم است که یک قابلیت ناظر/کنترل‌کننده جایگزین گردد. برای مثال، اگر شکست در یک قابلیت پایش رخ دهد، یک قابلیت پایش دیگر باید فعال شود. حالت دیگر آن است که مدل ناظر/کنترل‌کننده در مدیر مدل تغییر کند که در این صورت لازم خواهد بود یک ناظر/کنترل‌کننده جدید فعال گردد تا با مدل جدید سازگار شود. برای نمونه اگر گونه معماری از متمرکز به غیرمتمرکز تغییر یابد، سامانه باید از گونه غیرمتمرکز فرا-مدل پیروی نماید شکل ٦: سه گونه معماری عمومی ناظر/کنترل‌کننده با استفاده از فرا-مدل پیشنهادی: (الف) متمرکز، (ب) چند-لایه، (پ) نامتمرکز(شکل ٦-پ).

۶- جزئیات توصیف و درستی‌یابی

برای مشخص کردن جزئیات الگوی پیشنهادی و درستی‌یابی[47] آن، الگوی پیشنهادی توسط زبان ماد[48] توصیف می‌شود. ماد یک زبان سطح بالا است که یک محیط با کارایی بالا برای پشتیبانی از توصیف قابل‌اجرا ارائه می‌دهد و قابلیت برنامه‌ریزی توصیفی توسط منطق بازنویسی را دارد [٣٦]. از ماد به جهت سادگی، کارایی و پشتیبانی از منطق زمانی خطی[49][٣٧] استفاده شد که برای درستی‌یابی الگوی پیشنهادی لازم است.

۱.۶ توصیف صوری

هرکدام از دو نوع مجموعه قابلیت‌ها به دو زیرمجموعه

47- Verification
48- Maude
49- Linear Temporal Logic

افراز می‌گردنـد: فعال و غیرفعال. مجموعه فعال شـامل قابلیت‌هـای کاری و ناظر/کنترل‌کننده اسـت که در حال انجام عملیات هستند. مجموعه غیرفعال شامل قابلیت‌های اضافی و غیرفعال اسـت. یک موضوع مهم در این راستا، سـازگاری قابلیت‌های فعال و غیرفعـال با عملیات عادی و تطبیق‌پذیری اسـت؛ مثـلاً اگر یک قابلیـت کاری دچار شکسـت شودشکل ٤: نمای شـماتیک مطالعه موردی در عملیات عادی (الف). وقتی یکی از ابزارها (ابزار سـوراخ کردن در روبات سمت چپ) شکسته شده، و بازپیکربندی انجام شده اسـت (ب). (شکل ٤–الفشکل ٤: نمای شماتیک مطالعـه موردی در عملیـات عادی (الـف). وقتی یکی از ابزارهـا (ابزار سـوراخ کـردن در روبات سـمت چپ) شکسـته شـده، و بازپیکربندی انجام شـده است (ب). و شـکل ٤–ب)، نیاز اسـت که قابلیت دیگری با فعال شدن جایگزیـن آن گردد. توصیـف ۱ توضیح بالا را با تعریف Agent به‌عنوان عامل به شـکل صـوری بیان می‌دارد. P نشـان‌دهنده مجموعـه توانـی اسـت. TaskCapability و OCCapability بـه ترتیب مجموعه‌هـای قابلیت‌های کاری و ناظر/کنترل‌کننده را نشـان می‌دهند. id مجموعه یکتای شناسه برای عامل‌ها اسـت. اولین دو P برای قابلیت‌های فعال/غیرفعـال کاری و دو مورد بعـدی برای قابلیت‌های فعال و غیرفعال ناظر/کنترل‌کننده اسـتفاده شـده است. Resource منابع تحت پـردازش Agent را توصیف می‌کند و بـرای نگهداری ارتبـاط بین عامل‌ها و کارها اسـتفاده می‌گردد.

توصیـف ۱. توصیـف هرکـدام از عامل‌هـا از مجموعـه فعال/غیرفعـال کاری و ناظر/کنتـرل کننـده تشـکیل شـده اسـت. Resource وضعیـت پردازشـی منبـع تحت پـردازش عامـل را نشـان می‌دهـد. از این رو، به‌صـورت دنبالـه‌ای از کارهای باقی‌مانـده تعریف می‌شود (رابطه ۱).

$$Agent = id \times (\mathrm{P}(TaskCapability) \times \mathrm{P}(TaskCapability)) \times (\mathrm{P}(OCCapability) \times \mathrm{P}(OCCapability)) \times Resource$$
$$Resource = seq(TaskCapability)$$

تعریـف اصلی دیگر برای الگوی پیشـنهادی، تعریف مـدل برای مدیر مدل جـدول ۱: خلاصه قابلیت‌های ناظر/کنترل‌کننـده استفاده‌شـده در الگـوی پیشـنهادی برای پشتیبانی از معماری عمومی ناظر/کنترل‌کننده (ا = اجباری؛ ن = ناظر؛ ک = کنترل‌کننـده). از آن‌جایی‌که همه قابلیت‌ها اجباری هستند، ستونی برای آن در نظر گرفته نشده است. است (جدول ۱) که با OCModel اینجا تعریف شده است.

توصیـف ۲. OCModel مجموعه کلیـه مدل‌های ناظر/کنترل‌کننده برای سامانه است. هر مدل ناظر/کنترل‌کننده از مجموعه‌ای از قابلیت‌های ناظر/کنترل‌کننده و روابط میان آن‌ها تشکیل شده است (روابط ۲ و ۳).

$$OCModel \in \mathrm{P}(OCCapability) \times \mathrm{P}(OCMElement)$$
$$OCMElement = reltype \times OCCapability \times OCCapability$$
$$\text{where } reltype = \{needs, conflictswith, mandatory\}$$

OCMElement روابط میان قابلیت‌های ناظر/کنترل‌کننده را توصیـف می‌نمایـد (روابـط conflicts with، needs و mandatory به ترتیب برای رابطه‌های نیاز، تضاد و اجباری در توصیف). هرکدام از OCMElementها یک سـه‌تایی از reltype و دو OCCapability اسـت. reltype نـوع رابطه را مشخص می‌نماید. در خصوص رابطه اجباری که به شکل دو زوج قابلیت ناظر/کنترل‌کننده تعریف‌شده است، باید این نکتـه را افزود که این رابطه یک‌تایی فقط به دلیل تسـهیل توصیف و درستی‌یابی و همگن شدن شکل روابط به شکل دوتایی توصیف شده است. لذا برای رابطه‌ای به شکل rel-type = mandatory، هـر دو قابلیت ناظر/کنترل‌کننده یکی خواهند بود.

توصیف ۳. یک سـامانه بر اسـاس الگـوی معماری پیشـنهادی ترکیـب یـک مـدل ناظر/کنترل‌کننـده، مجموعـه‌ای از عامل‌هـا و مجموعـه‌ای از منابـع خواهد بود (رابطه).

$$OCSystem = OCModel \times \mathrm{P}(Agent) \times \mathrm{P}(Resource)$$

توصیف بالا ساختار الگوی پیشنهادی را نشان می‌دهد. برای توصیف رفتار، از ساختاری مشابه قوانین استفاده شده

است. برخی قوانین اینجا آمده است، ولی برای عدم وابستگی به زبان صوری خاص، قوانین به شـــکل شبه کد و نه زبان ماد آمده‌اند. هر قانون از یک بخش اگر برای تشخیص رخداد و یک بخش آنگاه به منزله بدنه رفتار تشکیل شده است. اگر و آنگاه با if و then در توصیف آمده‌اند.

توصیـــف ۴. قوانین تا رفتار ســـامانه تحت نظارت و کنترل را در قالب کلی اخذ[۵۰]، پردازش[۵۱] و رهاسازی[۵۲] منبع

$$(۵)\quad \begin{array}{l} if\ head(r) \in allTaskCapabilities(a) \wedge r \in Available\mathrm{Re}source\ then \\ setResource(a,r) \wedge AvailableResource = AvailableResource - \{r\} \end{array}$$

$$(۶)\ if\ \ getResource(a) == r \wedge head(r) \in allTaskCapabilities(a)\ then\ r = tail(r)\ if\ getResource(a) == r \wedge \neg head(r) \in allTaskCapabilities(a)\ then$$

$$(۷)\ setResource(a,\varnothing) \wedge AvailableResource = AvailableResource \cup \{r\}$$

توصیف‌های بالا بر چند قانون استوار است که به شکل ذیل تعریف شده‌اند. در تعاریف (۸) و (۹) دو متغیر a و r از نوع Agent و Resource است. چون لازم است عملیات رهاسازی و اخذ وضعیت منبع را بدانند، تعریف یک مجموعه را جهت نشان دادن وضعیت منابع آزاد (یعنی غیر مرتبط با عامل‌ها) معرفی می‌کنـــد. تابـــع allTaskCapabilities در تعریف کلیه قابلیت‌های کاری یک عامل را برمی‌گرداند. تابع getResource در تعریف منبع فعلی مرتبط با عامل را برمی‌گرداند و setResource در تعریف یک عامل را به یک منبع مرتبط می‌سازد.

۸- $a \in Agent$

۹- $r \in Resource$

۱۰- $AvailableResource \in \mathrm{P}(Resource)$

۱۱- $allTaskCapabilities : Agent \rightarrow \mathrm{P}(TaskCapability)$

۱۲- $getResource : Agent \rightarrow Resource$

۱۳- $setResource : Agent \times Resource \rightarrow Agent$

توصیـــف صوری این عملیـــات در قوانیـــن (۵) تا (۷) بـــه ترتیب آمده اســـت. عملیـــات اخذ یک منبـــع را به یک عامل مرتبط می‌ســـازد (قانـــون). وقتی عملیات اخذ موفق می‌شود، پردازش اولین کار در لیست کارهای منبع انجام می‌گیرد و آن کار از لیســـت کارهای منبع حذف می‌گردد (قانون ٦). درنهایت منبع رهاسازی شده و به لیست منابع آزاد بازمی‌گردد (رهاسازی در قانون ۷).

با این‌که توصیف ٤ رفتار کلی سامانه را نشان می‌دهد، اما با توضیحات بخش ٥.٤ هنوز ســـازگار نیســـت؛ چراکه باید سامانه بتواند به حالت سکون نیز وارد گردد تا عملیات تطبیق‌پذیری انجام شود. ســـامانه هنگام نیاز به تطبیق در سازوکار کنترلی وارد حالت بازپیکربشکل ۸: عملیات سامانه با اســـتفاده از الگوی پیشنهادیندی ناظر/کنترل‌کننده (شکل ۸) می‌شـــود. در این وضعیت، باید ســـامانه بـــا مدل ناظر/کنترل‌کننده انطباق پیدا کند (OCModel در توصیف ۲).

توصیف ۵. تعریف انطباق وضعیت فعلی سامانه را بر مدل ناظر/کنترل‌کننده در قالب یک تابع توصیف می‌کند. این تابع مجموعه قابلیت‌های ناظر/کنترل‌کننده را با مدل سامانه مقایسه می‌نماید.

١٤- $conformsto : OCSystem \rightarrow Boolean$

مقـــدار صحیح برای تابع، به معنای انطباق ســـامانه با مدل می‌باشـــد و مقدار ناصحیح نیاز به خود-ترمیمی در سطح سازوکار کنترل است.

توصیف ۶. برای متوقف کردن رفتار ســـامانه به جهت تطبیق، توصیف ســـه عملیات ذکرشـــده باید به شکل زیر تغییر کند.

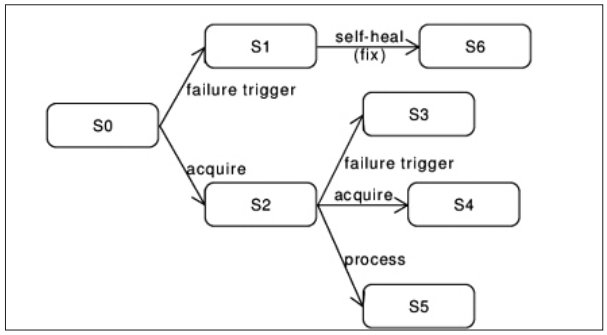
١٤- $acquire : Boolean \times Agent \times Resource \times \mathrm{P}Resource \rightarrow Agent \times \mathrm{P}Resource$

١٦- $process : Boolean \times Agent \rightarrow Agent$

١٧- $release : Boolean \times Agent \times \mathrm{P}Resource \rightarrow Agent \times \mathrm{P}Resource$

بخـــش Boolean در تعریف‌هـــای (١٥) تا (١٧) به مقدار بازگشتی تابع conformsto اشاره دارد. Agent به عامل انجام دهنـــده عملیات (اخذ، پردازش و رهاســـازی) بازمی‌گردد. عملیات اخـــذ، یـــک Resrouce را از مجموعه منابع موجود (Resource P) حذف می‌نمایـــد و به عامل موردنظر مرتبط می‌کند. در عملیات پردازش اولین کار از کارهای لازم‌الاجرا (ســـر دنباله[۵۳] کارها در تعریف) روی منبع انجام شـــده و آن کار از لیســـت کارها حذف می‌شود. عملیات رهاسازی

50- acquire
51- process
52- release
53- head



شکل ۹: چند حالت نمونه که محیط ماد برای درستی‌یابی ویژگی خود-ترمیمی بررسی می‌نماید

وظیفه بازگرداندن منبع فعلی به لیست منابع آزاد را دارد. هنگامی‌که مقدار بازگشتی تابع conformsto ناصحیح باشد، کلیه این عملیات متوقف می‌گردد.

خود-ترمیمی در الگوی ناظر/کنترل‌کننده خود-ترمیم از طریق چند قانون انجام می‌گیرد. وقتی شکست در یک قابلیت ناظر/کنترل‌کننده تشخیص داده می‌شود، آن قابلیت غیرفعال می‌گردد. درصورتی‌که به دلیل این شکست، سامانه از حالت انطباق با مدل خارج شود، باید عملیات خود-ترمیمی انجام شود.

توصیف ۷. توابع موردنیاز جهت تحقق خود-ترمیمی به شکل ذیل توصیف می‌گردند.

$$\text{۱۸- } \textit{if } \textit{conformsto}(oc) == \textit{false then } \textit{activate}(oc, \textit{missing}(oc)) \quad \text{where } oc : OCSystem$$

$$\text{۱۹- } \textit{missing} : OCSystem \rightarrow \mathrm{P}(OCCapability)$$

$$\text{۲۰- } \textit{activate} : OCSystem \times \mathrm{P}(OCCapability) \rightarrow OCSystem$$

۲.۶ درستی‌یابی صوری

با توصیف‌های ارائه‌شده در بخش پیشین، از زبان ماد جهت درستی‌یابی صوری استفاده شد. هدف سامانه تحت بررسی این بود که بتواند کلیه کارهای محول شده را انجام بدهد. به‌بیان‌دیگر، وقتی کلیه کارهای لازم روی منابع انجام بگیرد، کار سامانه انجام‌شده است. این موضوع شامل دو نوع سناریو می‌گردد: ۱) سناریوهایی بدون هیچ شکستی در سطح ناظر/کنترل‌کننده. ۲) سناریوهایی باوجود شکست در سطح ناظر/کنترل‌کننده. درستی‌یابی موفق در هر دو نوع سناریو منوط به ترمیم مشکلات احتمالی در سطح پایین‌تر سازوکار کنترلی است. در هر حالت، لازم است که عبارت «نهایتاً، هیچ منبع با کار انجام‌نشده وجود ندارد» درستی‌یابی شود. برای بیان این جمله لازم است که از منطق خطی زمانی استفاده گردد:

$$\text{۲۰- } \Diamond(\forall r : Resource \mid r = \varnothing)$$

برای درستی‌یابی سازوکار خود-ترمیمی دسته اول، تعدادی قانون نوشته شدند تا شکست را در سطح پایین و قابلیت‌های کاری ایجاد کرده و سازوکار خود-ترمیمی عادی آن‌ها را رفع نماید. برای دسته دوم قوانین، لازم است که علاوه بر قوانین دسته اول، قوانین دیگری برای ایجاد شکست در سطح ناظر/کنترل‌کننده درنظر گرفته شوند. در مطالعه موردی بخش بعدی، مثالی از این قوانین ارائه شده است. وقتی چنین شکست‌هایی به‌عنوان بخشی از رفتار عامل افزوده شدند، الگوی پیشنهادی به شکل موفقیت‌آمیزی توانست رفتار سامانه را به حالت عادی بازگرداند. به بیان دقیق‌تر این رفتارهای ایجادکننده شکست به نحوی نوشته شدند که پس از اجرا به‌عنوان رفتار عامل، مقدار بازگردانی شده از تابع conformsto (توصیف) را برابر با false نمایند. پس‌ازآن سامانه به حالت سکون وارد شده و عملیات خود-ترمیمی در سطح سازوکار کنترلی فعال گردید. درنتیجه، مقدار بازگشتی تابع conformsto برابر true شد. پس‌ازآن سامانه از حالت سکون خارج گردید و عملیات عادی سامانه ادامه پیدا کرد.

درستی‌یابی در محیط ماد از طریق بررسی مدل[54] و بررسی تمام حالت‌های ممکن (برای مطالعه موردی بخش بعد) صورت پذیرفت. برای انجام این کار، باید هر حالت جدید انتخاب و سامانه به آن حالت تغییر نماید. این فرایند برای هر حالت جدید تکرار می‌گردد، و نهایتاً یک ساختار درختی تشکیل داده می‌شود. شکل ۱۰ چند حالت نمونه را نشان می‌دهد. وقتی در یک حالت هیچ پیشرفتی ممکن نباشد، دو نتیجه می‌توان گرفت: ۱) تمام منابع پردازش شده‌اند و به‌بیان‌دیگر مسند[55] برقرار است (یعنی تمام عملیات روی تمام منابع انجام شده‌اند). ۲) هیچ عملیاتی قابل انجام نیست؛ یعنی

54- Model checking
55- Predicate

سامانه نتوانسته است عملیات را روی تمام منابع انجام بدهد.

رسیدن به وضعیت‌های بالا هدف نهایی درستی‌یابی در محیط ماد است. چون می‌توان وضعیت کلی سامانه را به صورت موفقیت یا شکست تعبیر نمود. در خصوص مثال استفاده شده در این مقاله، برخی از حالت‌ها و عملیات نمونه در شکل ۹ آمده‌اند.

۷-مطالعه موردی

در این بخش کاربردپذیری الگوی ناظر/کنترل‌کننده خود-ترمیم با اعمال آن روی مثال بخش ۵. ۱ نشان داده خواهد شد.

تعاریف (۲۲) و (۲۳) قابلیت‌های کاری و ناظر/کنترل‌کننده موردنیاز این مطالعه موردی هستند. تعریف (۲۳) از قابلیت‌های ناظر/کنترل‌کننده مربوط به معماری عمومی ناظر/کنترل‌کننده استفاده می‌کند. می‌توان از قابلیت‌های مطرح‌شده در «روش بازنشانی تغییرناپذیر» نیز استفاده کرد که در این صورت، جدول ۲: روابط میان قابلیت‌های ناظر/کنترل‌کننده در مثالقابلیت‌های ناظر/کنترل‌کننده مطابق جدول ۲ خواهند بود. بر اساس توصیف ۱، تعاریف (۲٤) تا (۲۷) توصیف روبات‌های مثال را نشان می‌دهد که هیچ قابلیت فعالی بجز مدیریت مدل ندارند.

۲۲- TaskCapabilities = {drill, insert, tighten}

۲۳- $OCCapabilities = \{Monitoring, Analyzing, Prediction, , Aggregation, \text{M}apping, RuleAdaptation, RulePerformanceEvaluation, OCModelManagement\}$ where $i = 1 \, to \, 3$

۲٤ – r_1, r_2, r_3 : Agent

۲۵- $r_1 = (1, \varnothing, TaskCapabilities, \{OCModelManagement\}, OCCapabilities - \{OCModelManagement\})$

۲٦- $r_2 = (2, \varnothing, TaskCapabilities, \{OCModelManagement\}, OCCapabilities - \{OCModelManagement\})$

۲۷- $r_3 = (3, \varnothing, TaskCapabilities, \{OCModelManagement\}, OCCapabilities - \{OCModelManagement\})$

در رابطه ، تعریف مدل ناظر/کنترل‌کننده برای حالت استفاده از معماری عمومی ناظر/کنترل‌کننده آمده است.

۲۸-

OCModel = (OCCapabilities,
{(needs, Prediction, Monitoring),(needs, Analying, Monitoring),
(needs, User Control, Rule Adaptation),(needs, Aggregation, Monitoring),
(needs, Aggregation, Analyzing),(needs, Aggregation, Prediction),
(needs, Mapping, Aggregation), (needs, Rule Performance Evaluation, Mapping),
(needs, Rule Adaptation, Simulation), (needs, Rule Adaptation, Mapping),
(mandatory, Monitoring, Monitoring), (mandatory, Mapping, Mapping),
(mandatory, Aggregation, Aggregation), (mandatory, OC Model Management,
OC Model Management)})

تعریف (۲۹) همین مدل را برای حالت استفاده از روش بازنشانی تغییرناپذیر نشان می‌دهد.

۲۹-

OCModel = ({Result Checker,Reconfiguration Algorithms, Monitoring},
{(needs, Reconfiguration Algorithms, Monitoring),
(needs, Result Checker, Reconfiguration Algorithms),
(mandatory, Monitoring, Monitoring),
(mandatory, Reconfiguration Algorithms,
Reconfiguration Algorithms),
(mandatory, Result Checker, Result Checker)})

عملیات اخذ، پردازش و رهاسازی طبق تعاریف (۱۵) تا (۱۷) استفاده می‌شوند. برای تحقق رخدادهای شکست در بخش قبل، تابع تعریف شده است. این تابع یک عامل/روبات را به‌عنوان ورودی گرفته و یکی از قابلیت‌های آن را (ناظر/کنترل‌کننده یا کاری) به نشانه خرابی غیرفعال می‌کند. این تابع (۳۰) در این جا برای تمام قابلیت‌ها توصیف شد، تا تمام قابلیت‌ها را بتوان دچار شکست نمود (مثلاً خرابی در قابلیت پایشگر یا قابلیت سوراخ کردن قطعات).

۳۰- $break : Agent \rightarrow Agent$

سپس با استفاده از امکانات منطق زمانی خطی هر شکست ممکن ارزیابی گردید. برای ارزیابی ویژگی خود-ترمیمی، حالت‌هایی که امکان خود-ترمیمی در آن‌ها وجود ندارد (حالت مرده)، کنار گذاشته شدند (همانند شکست در قابلیت‌های پایشگر در تمام عامل‌ها).

به کمک تعاریف ارائه شده، امکان درستی‌یابی صوری فراهم شد. با استفاده از زبان ماد، به شکل موفقیت‌آمیز ویژگی خود-ترمیمی از طریق مسند (۲۱) بیان گردید. به‌عبارت‌دیگر، نشان داده شد تمام حالت‌هایی که قابل ترمیم

هستند، توسط الگوی پیشـــنهادی به شکل موفقیت‌آمیزی ترمیم می‌شـــوند و ســـامانه به پردازش خود ادامه داده و تمام کارهای لازم روی منابع صورت می‌پذیرند. به‌عنوان یک جنبه دیگر از ارزیابی الگوی پیشنهادی، اثر حذف قابلیت خود-ترمیمی از سطح ناظر/کنترل‌کننده بررسی شد. برای این کار، قوانین خود-ترمیمی ســـازوکار کنترلی غیرفعال گردیدند. به جهت انجام این کار، قانون (۱۸) و بخش بولی مربوط به عملیات اخذ، پردازش و رهاســـازی حذف شدند (تعاریـــف ۱۵ تا ۱۷). با انجام این کار و حذف قابلیت ناظر/کنترل‌کننـــده، همان‌گونه که انتظار می‌رفـــت، روابط نیاز، اجباری و تضاد نقض شـــدند که در واقعیت به معنای عدم عملکرد صحیح در مؤلفه‌های کنترلی است.

۸- نتیجه‌گیری و کارهای آینده

الگوی ناظر/کنترل‌کننده خود-ترمیم که در این پژوهش پیشـــنهاد شد، مشکل شکســـت یا خرابی را در سطح ناظر/کنترل‌کننده به‌عنوان یک نمونه از یک سازوکار کنترلی حل می‌نماید. شباهت میان دستگاه عصبی مرکزی و سازوکار کنترلی کمک نمود تا با الهام گرفتن از طبیعت، مشـــکل حل شـــود؛ به‌صورت خاص، از پدیده ترمیم در دستگاه عصبی مرکزی گورخرماهی استفاده شد.

قابلیت ادامه عملیات پس از شکســـت یا خرابی در سطح ناظر/کنترل‌کننـــده نقطه قوت این الگو در مقایســـه با دیگر معماری‌های موجود ناظر/کنترل‌کننده اســـت. این هدف از طریق پیشـــنهاد مدل ناظر/کنترل‌کننده محقق شد که روابط میان مؤلفه‌های ناظر/کنترل‌کننده (در قالب قابلیت‌های ناظر/کنترل‌کننده) را به شکل روابط اجباری، نیاز و تضاد تعریف می‌نماید. این روابط باعث تعریف مدل و امکان انجام خود-ترمیمی بر اساس آن تا درجه قابل قبولی شد.

باوجود این ویژگی‌هـــا، محدودیت‌هایی برای این روش وجود دارد. اولین محدودیت الزام اتصال عامل‌ها به یکدیگر است. اگر این ارتباط بشـــکند، ممکن است سامانه به یک یا چند زیرسامانه ایزوله افراز گردد. در این حالت ممکن است که تناقض‌هایی میان تفســـیرهای مدل ناظر/کنترل‌کننده رخ دهد. چراکه زیرســـامانه‌های جدید روی همان مجموعه از منابع کار می‌کنند و ممکن است که رفتار پیش‌بینی‌نشده‌ای بروز کند. مثلاً یک منبع از دید یک سامانه به‌عنوان منبع در حال استفاده و از نظر سامانه دیگر، به‌عنوان منبع جدید در نظر گرفته شود. البته این مشکل در دیگر معماری‌های ناظر/کنترل‌کننده نیز وجود دارد.

یک محدودیت دیگر این روش، این است که مدل از قوانین ایستای «اگر-آنگاه» تشکیل‌شده که باعث عدم امکان تغییر در مدل در زمان حیات سامانه می‌گردد. این امر باعث عدم امکان افزودن مدل‌های جدید ناظر/کنترل‌کننده به سامانه می‌شود. اگر این قابلیت اضافه شـــود، باید به اثـــر قوانین جدید روی قوانیـــن جاری و امکان اســـتنتاج قوانین صحیح توجه نمود، چراکه ممکن است به شکل ناخواسته، قوانین به شکل نامتناهی یکدیگر را فعال کرده و سامانه هیچ‌گاه به نتیجه نرسد.

جـــدا از این موضوعات، به‌عنوان یـــک کار آتی، به نظر می‌رســـد که این الگو قابلیت توســـعه برای پوشـــش دیگر ویژگی‌های خود-* در سطح ناظر/کنترل‌کننده را دارا است. مدیر مدل در این موضوع می‌تواند نقشی کلیدی را بر عهده بگیـــرد. درنهایت این پژوهش به دنبـــال یک معماری ناظر/کنترل‌کننده است که بتواند به شکل پویا ساختار و رفتار خود را تغییر دهد تا گام‌های کوچکی در تحقق مفهوم سامانه‌های زنده منجر برداشته شوند.

مراجع

[1] J. A. Fernandez-Leon, "Behavioral robustness: An emergent phenomenon by means of distributed mechanisms and neuro-dynamic determinacy," BioSystems, vol. 107, pp. 34-51, / 2012.

[2] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," TAAS, vol. 4, / 2009.

[3] F. D. Macías-Escrivá, R. Haber, R. del Toro, and V. Hernandez, "Self-adaptive systems: A survey of current approaches, research challenges and applications," Expert Systems with Applications, vol. 40, pp. 7267-7279, / 2013.

[4] C. Cosentino and D. Bates, Feedback control in systems biology: Crc Press, 2011.

[5] H. Schmeck, C. Müller-Schloer, E. Cakar, M. Mnif, and U. Richter, "Adaptivity and self-organization in organic computing systems," ACM Trans. Auton. Adapt. Syst., vol. 5, pp. 10-1, sep/

Press, 2009.
[23] M. Gemberling, T. J. Bailey, D. R. Hyde, and K. D. Poss, "The zebrafish as a model for complex tissue regeneration," Trends in Genetics, vol. 29, pp. 611-620, / 2013.
[24] E. J. White, S. K. Kounelis, and C. A. Byrd-Jacobs, "Plasticity of glomeruli and olfactory-mediated behavior in zebrafish following detergent lesioning of the olfactory epithelium," Neuroscience, vol. 284, pp. 622-631, / 2015.
[25] E. C. Maier, A. Saxena, B. Alsina, M. E. Bronner, and T. T. Whitfield, "Sensational placodes: Neurogenesis in the otic and olfactory systems," Developmental biology, vol. 389, pp. 50-67, / 2014.
[26] L. Sulz and J. Bacigalupo, "Role of nitric oxide during neurogenesis in the olfactory epithelium," Biological research, vol. 39, pp. 589-599, / 2006.
[27] F. Nafz, J.-P. Steghöfer, H. Seebach, and W. Reif, "Formal Modeling and Verification of Self-* Systems Based on Observer/Controller-Architectures," in Assurances for Self-Adaptive Systems. vol. 7740, J. Cämara, R. r. de Lemos, C. Ghezzi, and A. Lopes, Eds., ed: Springer, 2013, pp. 80-111.
[28] M. Roth, J. Schmitt, R. Kiefhaber, F. Kluge, and T. Ungerer, "Organic Computing Middleware for Ubiquitous Environments," in Organic Computingâ€"A Paradigm Shift for Complex Systems, ed: Springer, 2011, pp. 339-351.
[29] R. Murch, Autonomic Computing, Information Management: IBM Press, 2004.
[30] A. Tarihi, H. Haghighi, and F. Shams Aliee, "An Improved Architectural Pattern for Organic Resource-Flow Systems," in Symposium on Computer Science and Software Engineering (CSSE) 2013, 2013.
[31] H. Seebach, F. Nafz, J.-P. Steghöfer, and W. Reif, "How to Design and Implement Self-organising Resource-Flow Systems," in Organic Computingâ€"A Paradigm Shift for Complex Systems, ed: Springer, 2011, pp. 145-161.
[32] B. Satzger, A. Pietzowski, W. Trumler, and T. Ungerer, "Using automated planning for trusted self-organising organic computing systems," in Autonomic and Trusted Computing, ed: Springer, 2008, pp. 60-72.
[33] J. Salzmann, R. Behnke, and D. Timmermann, "OC Principles in Wireless Sensor Networks," in Organic Computingâ€"A Paradigm Shift for Complex Systems, ed: Springer, 2011, pp. 503-516.
[34] P. Fischer, F. Nafz, H. Seebach, and W. Reif, "Ensuring Correct Self-reconfiguration in Safety-critical Applications by Verified Result Checking," in Proceedings of the 2011 Workshop on Organic Computing, New York, NY, USA, 2011, pp. 3-12.
[35] L. Bass, P. Clements, and R. Kazman, Software architecture in practice: Addison-Wesley, 2012.
[36] M. Clavel, F. DurÃ¡n, S. Eker, P. Lincoln, N. MartÃ-Oliet, J. Meseguer, et al., "Maude: specification and programming in rewriting logic," Theor. Comput. Sci., vol. 285, pp. 187-243, / 2002.
[37] M. Clavel, F. Dur'an, S. Eker, P. Lincoln, N. M. Oliet, J. e. Meseguer, et al., All About Maude - A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic, Lecture Notes in Computer Science: Springer, 2007.
2010.
[6] C. Müller-Schloer, H. Schmeck, and T. Ungerer, Organic Computing - A Paradigm Shift for Complex Systems: Springer, 2011.
[7] S. Skogestad and I. Postlethwaite, Multivariable feedback control: analysis and design vol. 2: Wiley New York, 2007.
[8] Y. Brun, G. D. M. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, et al., "Engineering self-adaptive systems through feedback loops," in Software engineering for self-adaptive systems, ed: Springer, 2009, pp. 48-70.
[9] A. Berns and S. Ghosh, "Dissecting Self-* Properties," in Proceedings of the 2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, Washington, DC, USA, 2009, pp. 10-19.
[10] R. r. De Lemos, H. Giese, H. A. Muller, M. Shaw, J. Andersson, M. Litoiu, et al., "Software engineering for self-adaptive systems: A second research roadmap," in Software Engineering for Self-Adaptive Systems II, ed: Springer, 2013, pp. 1-32.
[11] J. C. G. Walker, P. B. Hays, and J. F. Kasting, "A negative feedback mechanism for the long-term stabilization of the Earth's surface temperature," Journal of Geophysical Research, vol. 86, pp. 9776-9782, / 1981.
[12] S. Diaz, J. P. Grime, J. Harris, and E. McPherson, "Evidence of a feedback mechanism limiting plant response to elevated carbon dioxide," Nature, vol. 364, pp. 616-617, / 1993.
[13] A. K. Srivastava and D. Schlessinger, "Mechanism and regulation of bacterial ribosomal RNA processing," Annual Reviews in Microbiology, vol. 44, pp. 105-129, / 1990.
[14] U. Jenal and R. Hengge-Aronis, "Regulation by proteolysis in bacterial cells," Current opinion in microbiology, vol. 6, pp. 163-172, / 2003.
[15] S. Legg and M. Hutter, "Universal intelligence: A definition of machine intelligence," Minds and Machines, vol. 17, pp. 391-444, / 2007.
[16] G. N. Kryzhanovsky, Central nervous system pathology: a new approach: Springer Science & Business Media, 2012.
[17] L. Z. Holland, J. o. E. Carvalho, H. Escriva, V. Laudet, M. Schubert, S. M. Shimeld, et al., "Evolution of bilaterian central nervous systems: a single origin," EvoDevo, vol. 4, p. 27, / 2013.
[18] J. A. Fernandez-Leon, G. G. Acosta, and A. Rozenfeld, "How simple autonomous decisions evolve into robust behaviours?: A review from neurorobotics, cognitive, self-organized and artificial immune systems fields," Biosystems, vol. 124, pp. 7-20, / 2014.
[19] P. Cisek and J. F. Kalaska, "Neural mechanisms for interacting with a world full of action choices," Annual review of neuroscience, vol. 33, pp. 269-298, / 2010.
[20] U. Richter, M. Mnif, J. Branke, C. Müller-Schloer, and H. Schmeck, "Towards a generic observer/controller architecture for Organic Computing," GI Jahrestagung (1), vol. 93, pp. 112-119, / 2006.
[21] R. Frei and G. Di Marzo Serugendo, "Concepts in complexity engineering," International Journal of Bio-Inspired Computation, vol. 3, pp. 123-139, / 2011.
[22] M. Westerfield, L. I. Zon, and H. W. Detrich Iii, Essential zebrafish methods: cell and developmental biology: Academic