یک زبان توصیف معماری برای سیستم‌های قابل بازپیکربندی پویا

ندا نوروزی
دانشجوی کارشناسی ارشد
دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف
noroozi@ce.sharif.edu

سید حسن میریان حسین‌آبادی
استادیار
دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف
mirian@sharif.edu

چکیده

با توجه به افزایش نیاز به سیستم‌های نرم‌افزاری با رفتار پویا، موضوع استفاده از معماری‌های قابل بازپیکربندی پویا امروزه بسیار مورد توجه قرار گرفته است. برای ساخت یک سیستم قابل بازپیکربندی پویا نیازمند یک زبان توصیف معماری هستیم که توانایی بیان تمامی نیازمندی‌های این دسته از سیستم‌ها را داشته باشد. علی رغم این موضوع در بیشتر زبان‌های توصیف معماری به پیکربندی سیستم به صورت پویا نگریسته می‌شود. لذا در این مقاله به ارائهٔ یک زبان توصیف معماری به نام MyxADL برای سیستم‌های قابل بازپیکربندی پویا پرداخته‌ایم. زبان مذکور در کنار توصیف ساختار سیستم، قادر به بیان و توصیف شرایط و سیاست‌های بازپیکربندی نیز می‌باشد. در راستای قابل استفاده کردن زبان MyxADL، ابزارهای پشتیبان لازم برای زبان مذکور ارائه و ساخته شده است.

کلمات کلیدی

زبان‌های توصیف معماری، بازپیکربندی پویا، زبان xADL 2.0، سیاست‌های بازپیکربندی

۱- مقدمه

پویایی از اصلی‌ترین ویژگی‌های سیستم‌های با دسترسی بالا و دارای عمر طولانی[۱۱] می‌باشد. لذا، قابلیت تطبیق[۱] یکی از ویژگی‌های کیفی مهم نرم‌افزار است که سیستم‌های دارای عمر طولانی و سیستم‌هایی که شرایط محیطی آنها پیوسته در حال تغییر است، به آن نیازمندند. با توجه به آنکه محیط یک سیستم نرم‌افزاری همواره در حال تغییر است، یک سیستم نرم‌افزاری باید انعطاف‌پذیر باشد. در رابطه با راه‌های دستیابی به انعطاف‌پذیری در یک سیستم می‌توان به روش‌های متفاوتی اشاره داشت[۱۷] مانند استفاده از الگوریتم‌های گوناگون در شرایط محیطی مختلف، بازپیکربندی مؤلفه‌ها، مهاجرت کد، استفاده از نسخه‌های دیگر[۲] و غیره. در مقاله حاضر، روش بازپیکربندی پویا به عنوان یکی از راه‌کارهای دستیابی به پویایی در سیستم‌های نرم‌افزاری در نظرگرفته شده است. در سیستم نرم‌افزاری قابل بازپیکربندی پویا، پیکربندی سیستم با توجه به شرایط محیطی آن و در حین اجرای سیستم می‌تواند تغییر کند. منظور از تغییر در پیکربندی، تغییر توپولوژی مؤلفه‌های سیستم می‌باشد که از طریق اضافه و یا حذف یک مؤلفه از سیستم و یا اتصال و انفصال یک یا چند مؤلفه از یکدیگر حاصل می‌شود. برای انجام یک عمل بازپیکربندی کامل در سیستم، ممکن است مجبور به استفاده یک و یا ترکیبی از عملیات بازپیکربندی شویم.

برای ساخت یک سیستم قابل بازپیکربندی پویای خودکار وجود دو عنصر اصلی ضروری می‌نماید. عنصر اول قابلیت تشخیص تغییر در شرایط محیطی سیستم را به صورت خودکار و یا نیمه خودکار دارد. به بیانی دقیق‌تر می‌بایست زمان و نحوه اعمال تغییرات را در سیستم نرم‌افزاری به صورت خودکار و یا نیمه خودکار مشخص نماید. عنصر دوم مربوط به زیرساخت‌هایی[۳] می‌باشد که برای پیاده‌سازی تغییرات و انجام عملیات بازپیکربندی در اختیار عنصر اول قرار می‌گیرد. زیرساخت حمایت‌کننده از سیستم‌های قابل بازپیکربندی پویای خودکار، می‌بایست در کنار یک عامل بازپیکربندی‌کننده، دارای یک زبان توصیف معماری[۴] باشد که قادر به توصیف رفتارهای پویای سیستم است. به صورت کلی می‌توان گفت برای دستیابی به یک سیستم قابل بازپیکربندی پویا باید دارای توانایی‌های زیر باشیم[۷]:

۱- توانایی توصیف معماری در حالِ حاضر سیستم
۲- توانایی توصیف شرایط ایجاد کننده تغییر در سیستم
۳- توانایی تحلیل و بررسی تغییرات ایجاد شده در سیستم برای حصول اطمینان از درستی تغییرات
۴- توانایی اجرای تغییرات بر روی سیستم زمان اجرا بدون نیاز به خاموش کردن و یا اجرای مجدد کل سیستم

بررسی انجام شده بر روی زبان‌های توصیف معماری با تکیه بر توانایی آنها در توصیف سیستم‌های پویا، مؤید این موضوع می‌باشد که بیشتر زبان‌های توصیف معماری فاقد این توانایی می‌باشند. به بیانی ساده‌تر، بیشتر زبان‌های توصیف معماری دارای نگرشی ایستا به پیکربندی سیستم هستند و تنها تعداد اندکی از آنها توانایی حمایت از برخی از ویژگی‌های پویا را دارند. این امر خود مبین نیاز به ایجاد یک زبان توصیف معماری است که به صورت کامل قادر به حمایت از نیازمندی‌های سیستم‌های پویا باشد.

در مقاله حاضر ما با تمرکز بر بحث بازپیکربندی پویا به عنوان یکی از راه‌کارهای دستیابی به پویایی در نرم‌افزار به ایجاد و معرفی یک زبان توصیف معماری که قادر به حمایت از نیازمندی‌های این دسته از سیستم‌های نرم‌افزاری باشد، پرداخته‌ایم. زبان توصیف معماری جدید تولید شده، MyxADL نام دارد که نسخهٔ توسعه داده شده‌ای از زبان توصیف معماری xADL 2.0 می‌باشد. در MyxADL امکان توصیف شرایط و تغییرات محیطی، توصیف نقشه‌های بازپیکربندی و بیان ساختار سیستم وجود دارد. همچنین ابزارهای پشتیبانی لازم نیز برای این زبان فراهم شده است.

در ادامه در بخش دوم، مروری کلی بر زبان‌های توصیف معماری خواهیم داشت. در بخش سوم، موضوع بازپیکربندی پویا به صورت اجمالی مورد بررسی قرار می‌گیرد. در بخش چهارم این مقاله به معرفی و بررسی زبان MyxADL پرداخته‌ایم. در نهایت، بخش پنجم به نتیجه‌گیری و کارهای آینده اختصاص داده شده است.

۲- زبان‌های توصیف معماری

زبان‌های توصیف معماری، زبان‌هایی نمادین می‌باشند که در جهت حمایت از توسعهٔ نرم‌افزار بر مبنای معماری ایجاد شده‌اند. تحقیقات انجام شده در این زمینه باعث ایجاد زبان‌های توصیف معماری مختلفی شده است. هر یک از زبان‌های توصیف معماری با داشتن ابزارها و امکانات مناسب قادر به مدلسازی جنبه‌های مختلف معماری نرم‌افزار هستند. در بین گروه‌های تحقیقاتی، تعریف واحدی برای زبان‌های توصیف معماری وجود ندارد که در نتیجهٔ آن مجموعهٔ واحدی از شرایط و نیازمندی‌های یک زبان توصیف معماری ارائه نشده است[۱۴]. لکن علی رغم این موضوع، به طور کلی یک زبان توصیف معماری باید قادر به مدلسازی مؤلفه، رابط[۵] و پیکربندی[۶] سیستم باشد. همچنین برای مفید و قابل استفاده بودن یک زبان توصیف معماری به صورت عام، آن زبان می‌بایست ابزارهای پشتیبان[۷] مناسب را در اختیار کاربران خود بگذارد. هدف از رسمی‌سازی معماری توسط زبان‌های توصیف معماری، استفاده از رسمیت آنها توسط ابزارهای نرم‌افزاری برای تحلیل و تغییر معماری است. گرچه ابزارهای پشتیبانی جزء اصلی یک زبان توصیف معماری محسوب نمی‌گردند، لکن میزان مفید و قابل استفاده بودن یک زبان توصیف معماری به صورت مستقیم به ابزارهای لازم جهت طراحی، تحلیل و تکامل معماری و همچنین تولید سیستم‌های اجرایی و غیره وابسته است. یکی از جنبه‌های مهمی که در ابزارهای پشتیبانی زبان‌های توصیف معماری مورد توجه قرار می‌گیرد، میزان توانایی آنها در امر حمایت از پویایی است. لازم به ذکر است که توانایی این ابزارها برای مدلسازی پویایی برای تضمین اعمال درست تغییرات در سیستم نرم‌افزاری کافی نیست. بلکه ابزارهای تحلیلی لازم برای اطمینان از حفظ یکپارچگی و سازگاری[۸] سیستم نیاز است. به کمک این ابزارها می‌توان از برقراری تمامی ویژگی‌ها و محدودیت‌ها بر روی سیستم جدید اطمینان حاصل کرد. همچنین از نگاشت درست تغییرات سطح معماری بر روی عناصر پیاده‌سازی مربوطه، اجرای پیوستهٔ زیر سیستم‌های حیاتی با توجه به اعمال تغییرات در سیستم، مطمئن شد.

بیشتر زبان‌های توصیف معماری به پیکربندی به صورت ایستا نگاه می‌کنند[۱۸]. در کار تحقیقی انجام شده بر روی تعداد زیادی از زبان‌های توصیف معماری در [۱۴] تنها چهار زبان C2SADL[۶]، [۱۳]، Rapide[۱]، [۶]، [۱۳]، Darwin[۶]، [۱۶] و Weaves[۱۴] از گونه‌های مختلفی از پویایی حمایت می‌کنند. علاوه بر زبان‌های فوق می‌توان به زبان ACME/Armani[۵]، [۱۰] نیز در این میان اشاره داشت. زبان‌های Rapide و Darwin تنها امکان پشتیبانی از پویایی محدود شده[۹] را دارا هستند. این نوع پویایی می‌بایست قبل از زمان اجرا در سیستم مشخص شده باشد. در زبان Darwin امکان تکرار مؤلفه‌ها در زمان اجرا از طریق نمونه‌سازی پویا وجود دارد. در زبان Rapide از پیکربندی شرطی استفاده می‌شود. با استفاده از جملات where در این زبان امکان لازم برای بازنویسی پیکربندی سیستم در زمان اجرا ایجاد می‌شود. برای این منظور از عملیات اتصال و انفصال مؤلفه‌ها استفاده می‌شود. البته لازم به ذکر است در جهت حمایت از پویایی توسعه‌ای نیز بر روی زبان توصیف معماری Wright صورت گرفته است که موجب ایجاد زبان توصیف معماری Dynamic Wright شده است. در این زبان نیز از روش مشابه روش Rapide استفاده می‌شود. زبان‌های توصیف معماری C2SADL، Weaves و ACME/Armani بدون هیچ گونه قید و شرطی از پویایی در سیستم‌های نرم‌افزاری حمایت می‌نمایند. امکان تغییرات تصادفی در این زبان‌ها وجود دارد و سازگاری سیستم در زمان اجرا تضمین می‌گردد. در C2SADL و ACME/Armani امکان افزودن، حذف و یا جایگزینی مؤلفه‌ها توسط عملیات افزایش، حذف، اتصال و انفصال آنها از یکدیگر وجود دارد. چنین امکانی در زبان توصیف معماری Weaves با افزودن یک واسط قابل برنامه‌نویسی کاربردی[۱۰] به مدل Weaves وجود دارد.

در راستای حمایت از پویایی در زبان‌های فوق، با توجه به آنکه دو زبان Darwin و Rapide از پویایی محدود حمایت می‌کنند، ابزارهای ترجمهٔ مورد استفادهٔ این زبان‌ها در زمان ترجمه آنها تمامی حالت‌های مختلف پیکربندی سیستم را مورد تحلیل و بررسی قرار می‌دهند. زبان Weaves دارای یک ویرایشگر بصری به نام Jacquard است. این

ویرایشگر از واسطهای قابل برنامه‌نویسی برنامه‌های کاربردی جهت مدلسازی پویایی استفاده می‌نماید. لازم به ذکر است که این ویرایشگر یک ابزار تجاری می‌باشد. ابزار ArchStudio 2.0 در راستای حمایت از زبان C2SADL ایجاد شده است. ابزار مذکور قابلیت ایجاد و اتصال مؤلفه‌ها و رابط‌های جدید را به سیستم زمان اجرایی دارد. لکن تولیدکنندگان این ابزار در نسخه‌های جدید آن به جای استفاده از زبان C2SADL با توجه به محدودیت‌های این زبان، از زبان توصیف معماری جدید xADL 2.0 بهره برده‌اند. در زبان ACME/Armani نیز مترجم خاص این زبان نیز در یک پروژه تحقیقاتی تولید شده است.

۳- بازپیکربندی پویا

با عنایت به آنکه زبان MyxADL به منظور پاسخ‌گویی به نیازهای توصیفی سیستم‌های قابل بازپیکربندی پویا ایجاد شده است، در این فصل مرور کلی بر بحث بازپیکربندی پویا خواهیم داشت.

بازپیکربندی یک سیستم نرم‌افزاری می‌تواند آن را با تغییرات پیش آمده در محیط تطبیق می‌دهد. بازپیکربندی در یک سیستم نرم‌افزاری، می‌تواند در سه مقطع زمانی اتفاق بیافتد: در زمان طراحی، در زمان کامپایل و در زمان اجرا. در بازپیکربندی در زمان اجرا، بر خلاف بازپیکربندی در زمان‌های قبل از اجرا، به حالت اجرایی سیستم وابسته می‌باشد. به طور کلی بازپیکربندی در زمان اجرا در سیستم‌های نرم‌افزاری می‌تواند به دو صورت ایستا و پویا رخ دهد. در بازپیکربندی به صورت ایستا، سیستم در برخورد با تغییرات ایجاد شده در شرایط و نیازمندی‌های آن، متوقف می‌شود و پس از اعمال تغییرات لازم و تطبیق با شرایط جدید، مجدداً راه‌اندازی می‌شود. در مقابل بازپیکربندی ایستا در زمان اجرا، در روش بازپیکربندی پویا نیازی به توقف، ترجمه و شروع مجدد نمی‌باشد. به بیان بهتر بازپیکربندی پویا به معنی تغییر در پیکربندی یک سیستم در زمان اجرا است؛ به نحوی که به در دسترس بودن سیستم خللی وارد نشود. منظور از تغییر در پیکربندی، تغییر توپولوژی مؤلفه‌های سیستم می‌باشد که از طریق عملگرهای بازپیکربندی انجام می‌شود. برای انجام یک عمل بازپیکربندی کامل در سیستم، ممکن است مجبور به استفاده یک و یا ترکیبی از عملگرهای بازپیکربندی شویم. عملگرهای بازپیکربندی در یک سیستم نرم‌افزاری شامل موارد زیر است:

- افزودن یک مؤلفهٔ جدید به سیستم
- حذف یک مؤلفه از سیستم
- ارتقاء دادن و یا جابجایی یک مؤلفه
- تغییر توپولوژی سیستم با استفاده از عملگر الحاق یک مؤلفه به مؤلفهٔ دیگر (عملگر افزودن یک رابط به سیستم)
- تغییر توپولوژی سیستم با استفاده از عملگر انفصال یک مؤلفه از مؤلفهٔ دیگر (عملگر حذف یک رابط از سیستم)

به طور کلی می‌توان یک عمل بازپیکربندی پویا را در یک سیستم نرم‌افزاری شامل مراحل زیر دانست [۱۵]:

۱- *مرحله آغازین بازپیکربندی:* تشخیص نیاز به انجام بازپیکربندی در سیستم با توجه به مقایسه صورت گرفته بین شرایط مطلوب سیستم با شرایط جاری

۲- *مرحله تعیین نقشه بازپیکربندی:* تعیین سیاست بازپیکربندی به صورت مجموعه‌ای ترتیب‌دار از عملگرهای بازپیکربندی

۳- *مرحله اجرای بازپیکربندی:* انجام و اجرای عملیات بازپیکربندی در سیستم

۴- *مرحله اطمینان از صحت بازپیکربندی:* حصول اطمینان از اجرای کامل و صحیح عملیات بازپیکربندی و اطمینان از قرارگیری سیستم در وضعیت مجاز

یکی دیگر از مسائل مطرح در مسألهٔ بازپیکربندی نحوهٔ مدیریت آن می‌باشد. مسألهٔ مدیریت تغییرات در سیستم نرم‌افزاری از اهمیت بالایی برخوردار است زیرا به تعیین قسمت‌هایی که نیازمند تغییر هستند، کمک می‌کند. همچنین زمینهٔ مناسب برای تعریف و پیاده‌سازی تغییرات را ایجاد می‌نماید. علاوه بر دلایل بالا در جهت حفظ سازگاری سیستم تغییرات را کنترل می‌نماید. مدیریت بازپیکربندی شامل چهار مرحلهٔ ذکر شده در بالا برای انجام یک عمل بازپیکربندی کامل است. به طور کلی مدیریت بازپیکربندی در یک سیستم نرم‌افزاری می‌تواند به دو صورت مجتمع شده و توسط یک مؤلفه به همین نام و یا به صورت توزیع شده در بین مؤلفه‌های سیستم انجام پذیرد. علاوه بر تقسیم‌بندی انجام شده نحوهٔ مدیریت بازپیکربندی را با توجه به میزان خودکار و یا غیرخودکار بودن آن نیز می‌توان مورد بررسی قرار داد. که این امر به طور مستقیم متأثر از آن است که مدیر بازپیکربندی در انجام مراحل ذکر شده در یک عمل بازپیکربندی تا چه میزان به صورت مستقل و خودکار عمل نماید. برای انجام مدیریت بازپیکربندی به صورت خودکار به توصیف وضعیت کنونی سیستم و وضعیت جدیدی که خواهان انتقال به آن هستیم، نیازمندیم. لذا می‌بایست قادر به توصیف و بیان شرایط تغییر، نقشه‌های بازپیکربندی، محدودیت‌های سیستم و پیکربندی آن باشیم. در نتیجه در یک سیستم قابل بازپیکربندی پویا که از مدیریت خودکار در آن استفاده می‌شود، نیازمند یک زبان توصیف معماری با توانایی توصیف موارد فوق هستیم. همچنین از زبان توصیف معماری می‌توان در انجام تحلیل‌های لازم در جهت اطمینان از درستی نقشه‌های بازپیکربندی نیز استفاده کرد.

۴- زبان توصیف معماری MyxADL

همانطور که پیشتر نیز به آن اشاره شد، برای داشتن یک سیستم پویای خودکار، باید قادر به توصیف معماری سیستم، توصیف شرایط تغییر اثرگذار بر روی سیستم وتوصیف نحوه اعمال تغییرات بر روی

سیستم نرم‌افزاری باشیم. با توجه به عدم حمایت زبان‌های توصیف معماری از تمامی نیازمندی‌های سیستم‌های قابل بازپیکربندی پویا در مقاله حاضر به معرفی یک زبان توصیف معماری جدید، زبان MyxADL، برای این دسته از سیستم‌های نرم‌افزاری می‌پردازیم. MyxADL یک نسخهٔ توسعه یافته از زبان توصیف معماری xADL 2.0 می‌باشد.

۴-۱- زبان توصیف معماری xADL 2.0

در سال‌های اخیر، در نتیجه گسترش پروژه‌های تحقیقاتی در زمینه‌های خاص معماری نرم‌افزار، حجم زیادی از اطلاعات معماری[۱۱] بوجود آمده است. داده‌های جدید بوجود آمده می‌بایست قابل انعکاس در مدل‌های معماری باشند[۷]. این در حالی است که زبان‌های توصیف معماری معمول بوجود آمده تنها قادر به نمایش و بیان مجموعه‌ای از ویژگی‌های یک مدل معماری هستند و قابلیت گسترش را نیز ندارند. لذا علل نیاز به داشتن یک زبان توصیف معماری به همراه ابزارهای آن که به سادگی قابل توسعه باشند و بتوان ویژگی‌های مختلف سیستم‌های نرم‌افزاری مربوط به دامنه‌های متفاوت را به کمک آن نمایش داد، روشن است.

زبان xADL 2.0 یک زبان توصیف معماری پیمانه‌ای قابل گسترش مبتنی بر XML است[۳]. هسته اصلی این زبان بر اساس xArch [۹] است. xArch یک هسته ساده است برای نمایش عناصر اصلی مدل معماری نرم‌افزاری که شامل مؤلفه، رابط و پیکربندی آنها است. برای عناصر تعریف شده در xArch هیچ گونه ویژگی و یا محدودیت خاصی در نظر گرفته نشده است. در ضمن این هستهٔ مرکزی نیز با استفاده از شماهای XML بوجود آمده است. لذا با توجه به عمومی بودن xArch و فناوری شمای XML، این هسته اولیه به آسانی قابل گسترش و تطبیق با نیازهای دامنه‌های جدید است. شمای مربوط به xArch موارد زیر را مشخص می‌نماید:

- نمونه‌های مؤلفه، رابط، واسط و اتصال (ارتباط بین واسطها)
- زیربخش‌های معماری، جهت نمایش مؤلفه‌ها و رابطهای ترکیبی
- گروه، جهت دسته‌بندی کردن منطقی عناصر اولیه و ساده معماری که از بیرون به صورت یک جزء واحد دیده شوند.

یکی از نکات مورد توجه در زبان xADL 2.0، استفاده از شماهای مختلف برای توصیف معماری در زمان طراحی و در زمان اجرا است. معمولاً اموری که در زمان طراحی مطرح هستند در زمان اجرا وجود ندارند. به عنوان نمونه ممکن است که مجموعه‌ای از عناصر در زمان طراحی به صورت عناصر اختیاری مطرح شوند، این در حالی است که در زمان اجرا مشخص می‌شود که این عناصر وجود دارند و یا خیر. یکی دیگر از شماهای مورد استفاده در xADL شمای مربوط به پیاده‌سازی می‌باشد. در صورتی که بتوان بین توصیف انجام شده از سیستم و پیاده‌سازی آن در توصیف رابطه برقرار کرد، می‌توان به کمک توصیف سیستم، سیستم نرم‌افزاری اجرایی را ایجاد و مدیریت کرد. در نسخهٔ کنونی زبان توصیف معماری xADL 2.0، شماهای این زبان برای توصیف یک سیستم قابل بازپیکربندی مناسب نمی‌باشند و تنها می‌توان ساختار سیستم را به کمک آن توصیف کرد. لذا زبان xADL 2.0 برای پشتیبانی از تکامل نرم‌افزار نیاز به توسعه دارد که در این راستا زبان MyxADL ایجاد شده است.

۴-۲- توسعه‌های انجام شده

جهت مناسب شدن زبان xADL 2.0 برای توصیف معماری سیستم‌های پویا توسعه‌های لازم را انجام داده‌ایم که نتیجه آن ایجاد نسخهٔ MyxADL بوده است. این امر با افزودن چندین شمای XML به ساختارهای زبان xADL 2.0 حاصل شده است. همچنین برای سهولت تولید، نگهداری و مدیریت فایل‌های توصیف معماری ایجاد شده توسط زبان MyxADL، تغییرات لازم را نیز در برخی ابزارهای زبان xADL 2.0 بوجود آورده‌ایم و آنها را با نسخهٔ جدید این زبان، تطابق داده‌ایم.

۴-۲-۱- توصیف ویژگی‌های کیفی

ویژگی‌های کیفی مورد نیاز یک سیستم نرم‌افزاری را نمی‌توان پس از تولید بر اساس نیازمندی‌های وظیفه‌مندی، به نرم‌افزار اضافه کرد[۴]. بنابراین فقط در صورتی یک ویژگی کیفی در یک سیستم نرم‌افزاری وجود خواهد داشت که از ابتدای فرایند تولید برای دست‌یابی به آن برنامه‌ریزی شود. لذا معماری نرم‌افزار یک روش مناسب برای توجه به ویژگی‌های کیفی سیستم نرم‌افزاری، قبل از زمان توسعه آن می‌باشد[۲].

در سیستم‌های پویا، در شرایط محیطی مختلف معمولاً مجبور به استفاده از عناصر نرم‌افزاری با ویژگی‌های کیفی متفاوت می‌باشیم. به طور مثال در یک سیستم سیار با داشتن توان قدرتی محدود، در حالت عادی می‌توان از یک مؤلفهٔ نرم‌افزاری با کارایی بالا و مصرف زیاد انرژی استفاده کرد. در صورت کم شدن میزان عمر باطری سیستم، مجبور به تطبیق با شرایط جدید محیطی هستیم. لذا، می‌توان با تعویض مؤلفه قبلی با یک مؤلفهٔ نرم‌افزاری جدید با کارایی و مصرف انرژی کمتر، سیستم را با شرایط جدید محیطی تطبیق داد.

با داشتن یک توصیف معماری نرم‌افزار که شامل توصیف ویژگی‌های کیفی نیز باشد، قادر به انجام برخی از عملیات تحلیلی بر روی سیستم در زمان طراحی و اجرا خواهیم بود که در ادامه به آنها اشاره شده است:

- تحلیل‌های زمان طراحی

با داشتن یک توصیف کامل و ایجاد ابزارهای مناسب می‌توان قسمتی از فرایند تحلیل و ارزیابی معماری را به صورت خودکار انجام داد. به عنوان نمونه در فرایند ارزیابی معماری به روش ATAM[۱۲][۴]، به ویژگی‌های کیفی توجه خاص شده است. لذا می‌توان با تولید

توصیف مناسب از معماری سیستم و ایجاد ابزارهای لازم برای این منظور، بخشی از فرایند ارزیابی معماری به روش ATAM را به صورت کارا و خودکار انجام داد. از جمله کارهای انجام شده در زمینه خودکارسازی بررسی ویژگی‌های کیفی در معماری نرم‌افزار کار انجام شده در [۱۲] می‌باشد. در پروژهٔ مذکور روشی برای تحلیل و بررسی ویژگی‌های کیفی براساس توصیفات صورت گرفته از ویژگی‌های کیفی معماری نرم‌افزار ارائه شده است. همچنین ابزار لازم نیز در این راستا تولید شده است.

- تحلیل‌های زمان اجرا

همانطور که گفته شد، در یک سیستم نرم‌افزاری پویا در شرایط محیطی مختلف، مجبور به استفاده از عناصر نرم‌افزاری با ویژگی‌های کیفی متفاوت هستیم. به طور کلی می‌توان انجام یک عمل تطبیق در یک سیستم نرم‌افزاری پویا را به چهار مرحله تقسیم کرد:مرحله آغاز تغییر،مرحله انتخاب نقشه تغییر،مرحله پیاده‌سازی نقشه تغییر و مرحله درستی‌یابی و ارزیابی سیستم تطبیق‌داده شده در سیستم نرم‌افزاری. با توجه به مطالب ذکر شده، روشن است که در صورت بیان و تعیین ویژگی‌های کیفی در توصیف معماری نرم‌افزار و داشتن ابزارهای تحلیلی لازم می‌توان در زمان تغییر، توسط عامل تحلیل‌گر، توانایی انتخاب خودکار را به سیستم نرم‌افزاری پویا بیافزاییم. به بیان دیگر، مرحله انتخاب نقشهٔ تغییر توسط سیستم و به صورت خودکار و حتی بدون نیاز به بیان کامل نقشهٔ بازپیکربندی در زمان طراحی صورت می‌پذیرد. این امر به ویژه در سیستم‌های پویای نرم‌افزاری خودکار اهمیت می‌یابد. زیرا در این گونه سیستم‌ها، می‌بایست بدون دخالت نیروی انسانی و با توجه به شرایط توصیف شده در زمان طراحی، سیستم نرم‌افزاری با شرایط جدید تطبیق نماید.

زبان MyxADL قادر به بیان ویژگی‌های کیفی بر روی عناصر نرم‌افزاری شامل مؤلفه‌ها و رابط‌های سیستم می‌باشد. ویژگی‌های کیفی بر روی نوع عناصر قابل تعریف می‌باشند. در نتیجه مؤلفه‌ها و یا رابط‌های یک نوع یکسان دارای مجموعهٔ واحدی از ویژگی‌های کیفی می‌باشند. هر ویژگی کیفی دارای نوع مشخص و یک مقدار می‌باشد. مقدار ویژگی‌های کیفی برای مؤلفه‌ها و رابط‌های توصیف شده در بخش توصیف ساختار سیستم تعیین می‌گردد. این امر باعث آن می‌شود که بتوان عناصر نرم‌افزاری از یک نوع واحد با مجموعه ویژگی‌های کیفی یکسان ولی مقادیر مختلف تعریف کرد.

۴-۲-۲- توصیف سیاست‌های بازپیکربندی

در یک دسته‌بندی کلی انجام شده بر حسب معیار زمان، تغییرات نرم‌افزاری را به دو دستهٔ تغییرات قابل پیش‌بینی و تغییرات خاص تقسیم می‌نمایند. تغییرات قابل پیش‌بینی تغییراتی می‌باشند که در زمان طراحی قابل پیش‌بینی باشند و در مقابل تغییرات موردی تغییراتی هستند که در زمان طراحی پیش‌بینی نشده و در زمان اجرا به وقوع می‌پیوندند. در یک سیستم نرم‌افزاری قابل بازپیکربندی پویای خودکار، باید سیستم قادر به انجام تمامی مراحل بازپیکربندی بدون دخالت نیروی انسانی و به صورت خودکار باشد. برای این منظور، سیستم می‌بایست شامل یک مدیر بازپیکربندی با وظیفه‌مندی تشخیص تغییر در شرایط محیطی، تعیین نقشه تطبیق در سیستم و مدیریت و اجرای نقشه تعیین شده باشد. در یک سیستم نرم‌افزاری که از تغییرات قابل پیش‌بینی حمایت می‌کند، شرایط تغییر در زمان طراحی، در سیستم تعیین می‌شود و نقشه‌های بازپیکربندی لازم برای تطبیق با شرایط جدید در اختیار سیستم قرار می‌گیرد. نقشهٔ بازپیکربندی متشکل از مجموعه‌ای از عملیات بازپیکربندی و ترتیب اِعمال آنها بر روی سیستم اجرایی است. مدیر بازپیکربندی با استفاده از سیاست‌های بازپیکربندی، نقشهٔ بازپیکربندی را در زمان تغییر در سیستم مشخص می‌نماید. سیاست بازپیکربندی متشکل از تعریف شرایط تغییر و نقشهٔ بازپیکربندی مربوط به آن است.

```
- <Policy>
    <Event eventId="Event1"/>
    <Evaluate>file://C:/Evalitions/EvlalEvent1.java </Evaluate>
  - <Action>
      <Remove removeId="ChatClient3"/>
    </Action>
  </Policy>
- <Policy>
    <Event eventId="Event2"/>
    <Evaluate>file://C:/Evalitions/EvlalEvent2.java </Evaluate>
  - <Action>
      <Remove removeId="ChatClient1"/>
    </Action>
  </Policy>
- <Policy>
    <Event eventId="Event3"/>
    <Evaluate>file://C:/Evalitions/EvlalEvent3.java </Evaluate>
  - <Action>
    - <Add>
      - <Component id="ChatClient2">
          <Description>Chat Client 2 Component</Description>
        + <Interface id="ChatClient2.IFACE_TOP"></Interface>
        + <Interface id="ChatClient2.IFACE_BOTTOM"></Interface>
          <Component_Type type="simple" href="#Client_type"/>
        </Component>
      </Add>
    </Action>
  </Policy>
</Policies>
```

شکل ۱- یک فایل سیاست بازپیکربندی

ما با توسعهٔ زبان xADL 2.0 و افزودن بخش توصیف سیاست‌های بازپیکربندی، امکان لازم برای توصیف سیستم‌های پویای خودکار حمایت‌کننده از تغییرات قابل پیش‌بینی را بوجود آورده‌ایم. سیاست‌های بازپیکربندی مانند دیگر بخش‌های زبان MyxADL درقالب XML هستند و در فایل جداگانه‌ای نسبت به توصیف معماری سیستم قرار می‌گیرند. این امر سبب می‌شود که بتوان به راحتی در زمان اجرا بدون تأثیر بر سایر بخش‌های سیستم، سیاست‌های بازپیکربندی جدید را به سیستم افزود و یا از آن حذف کرد. لذا این امکان در سیستم

بوجود می‌آید که از تغییرات پیش‌بینی نشده و موردی[۱۳] در سیستم در زمان طراحی نیز حمایت کند. همچنین برای انجام یک عمل بازپیکربندی کامل در سیستم، ممکن است مجبور به استفاده از یک و یا ترکیبی از عملیات ساده بازپیکربندی شویم. عملگرهای اولیهٔ بازپیکربندی قابل تعریف در زبان MyxADL شامل افزودن/حذف یک مؤلفه، افزودن/حذف یک رابط، افزودن/حذف یک اتصال، جایگزینی یک مؤلفه با نسخهٔ جدید آن می‌باشد. در شکل ۱ نمونهٔ ساده شده‌ای از فایل سیاست‌های بازپیکربندی نشان داده شده است. مطابق با شکل، هر سیاست بازپیکربندی علاوه بر تعیین رخداد تغییر و نقشهٔ بازپیکربندی مربوطه، شامل یک بخش سوم نیز می‌باشد. در این بخش آدرس مکان فایل مربوط به کلاس ارزیابی‌کننده تغییر بوجود آمده در سیستم، مشخص شده است. در کلاس ارزیابی‌کننده با توجه به رخداد روی داده در سیستم، بررسی می‌شود که آیا نقشهٔ مشخص شده در سیاست بازپیکربندی حاضر بر روی سیستم قابل اجرا هست یا خیر.

۴-۲-۳- توصیف شرایط محیطی

توسعهٔ دیگر انجام شده بر روی زبان xADL 2.0 برای دستیابی به یک زبان توصیف معماری برای سیستم‌های پویای خودکار افزودن امکانات لازم برای توصیف شرایط محیط اجرایی سیستم نرم‌افزاری می‌باشد. با توجه به این امر می‌توان عوامل بیرونی مؤثر بر روی سیستم نرم‌افزاری را مشخص کرد. لذا این امکان بوجود می‌آید که سیستم تنها با نظارت بر برخی از عوامل بیرونی، شرایط تغییر را تشخیص دهد. در نتیجهٔ این امر می‌توان مرحلهٔ آغاز عملیات بازپیکربندی، مرحله تشخیص خطا را به صورت خودکار انجام داد. در فایل توصیف شرایط بازپیکربندی، متغیرهای محیطی که بر روی سیستم اثرگذار هستند مشخص می‌شوند و در کنار هر یک از متغیرهای محیطی محدودهٔ مجاز آنها تعیین می‌گردد. در صورتی که مقدار زمان اجرای متغیرهای محیطی تعریف شده در محدودهٔ مجاز نباشد، این امر می‌تواند آغازگر عملیات بازپیکربندی در سیستم باشد. در شکل ۲ نمونهٔ ساده‌ای از یک فایل توصیف شرایط محیطی نشان داده شده است.

۴-۲-۴- توصیف محدودیت‌های سیستم

آخرین توسعهٔ انجام شده بر روی زبان xADL 2.0 در جهت شکل‌گیری زبان MyxADL افزودن ساختارهای لازم برای بیان یک محدودیت در سیستم نرم‌افزاری است. هر محدودیت به صورت یک عبارت منطقی بر روی ویژگی‌های مؤلفه‌ها و رابطها بیان می‌گردد. محدودیت‌های بیان شده بر روی سیستم می‌بایست همواره در سیستم برقرار باقی بمانند. لذا نقشهٔ بازپیکربندی تعیین شده برای اجرا براساس این محدودیت‌ها بررسی و کنترل می‌گردد. در صورتی که در مدل بدست آمده پس از اجرای نقشهٔ بازپیکربندی، محدودیت‌ها نقض نشده باشند، آن نقشه معتبر شمرده می‌شود و قابل اجرا بر روی سیستم نرم‌افزاری است.

```
- <Context>
  - <Variable variableId="var2">
      <Name>Bandwidth</Name>
      <Observer> file://c:/observers/bandwidthDriver.java </Observer>
    - <Range>
      - <BooleanExpression>
          <Not>0.4</Not>
        </BooleanExpression>
      </Range>
      <Event eventId="Event1"/>
    </Variable>
  - <Event eventId="Event1">
    - <BooleanExpression>
      - <Expression>
        - <CompExpression>
          - <Equal>
              <exp1>Bandwidth</exp1>
              <exp2>0.4</exp2>
            </Equal>
          </CompExpression>
        </Expression>
      </BooleanExpression>
    </Event>
  </Context>
```

شکل ۲- یک فایل توصیف شرایط محیطی

۴-۳- وارسی در زبان MyxADL

نوع تحلیلی که یک زبان توصیف معماری ارائه می‌نماید، به مدل معنایی زبان و به بیان ساده‌تر به نوع و نحوهٔ توصیف آن از معماری بستگی دارد. به طور مثال در زبان توصیف معماری Wright از مدل CSP برای تحلیل روابط بین مؤلفه‌ها و رابطها در جهت وارسی امکان وجود بن‌بست در سیستم استفاده می‌شود. در حالیکه در زبان دیگری مانند زبان C2SADL تنها محدودیت‌های سبک معماری مورد بررسی قرار می‌گیرد. با توجه به این امر، در یک دسته‌بندی کلی انواع تحلیل‌هایی که توسط زبان‌های توصیف معماری مختلف ارائه می‌شوند را می‌توان شامل موارد زیر دانست[۱۴]:

- تحلیل‌هایی که شامل شبیه‌سازی رفتار سیستم نرم‌افزاری هستند.
- تحلیل‌هایی که شامل ویرایش و ترجمه زبان می‌باشند.
- تحلیل‌هایی که شامل بررسی محدودیت‌های تعریف شده در سبک معماری و سیستم نرم‌افزاری است.

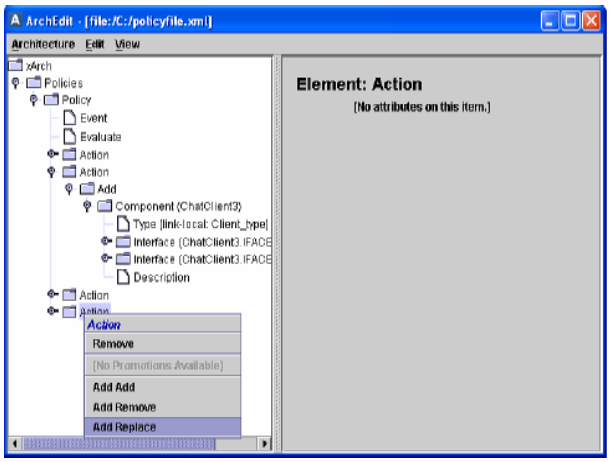
با توجه به مطالب مذکور و دسته‌بندی انجام شده از نوع تحلیل‌هایی که یک زبان توصیف معماری می‌تواند از آن حمایت نماید، زبان توصیف معماری MyxADL از تحلیل‌های نوع دوم و سوم حمایت می‌نماید و در این راستا ابزارهای پشتیبانی لازم را نیز دارا است. با توجه به آنکه در زبان MyxADL تنها به توصیف ساختار معماری سیستم(نحوهٔ اتصالات)، نوع مؤلفه‌ها، نوع رابطها و نمونه‌های آن‌ها پرداخته می‌شود، لذا وارسی‌های انجام شده در این زبان تنها محدود به تحلیل‌های ساختاری می‌گردد. در این دسته از تحلیل‌ها درستی[۱۴] و

کامل بودن[۱۵] توصیف انجام شده از سیستم مورد بررسی قرار می‌گیرد. با توجه به ابزارهای مورد استفاده برای زبان MyxADL، از روی توصیف سیستم می‌توان سیستم نرم‌افزاری اجرایی را تولید کرد. در این راستا تحلیل‌های لازم بر روی نحو و معنای زبان انجام می‌شود. همچنین محدودیت‌های تعریف شده بر روی معماری سیستم نیز مورد بررسی قرار می‌گیرد. اهمیت تحلیل‌های انجام شده در زمان اعمال نقشه‌های بازپیکربندی مشخص می‌گردد. با توجه به این امر تضمین می‌گردد که پس از اجرای یک نقشهٔ بازپیکربندی هنوز توصیف جدید تولید شده از سیستم درست و کامل باشد و محدودیت‌های تعریف شده در آن صادق باقی بماند. در نسخهٔ کنونی زبان MyxADL، مدلی از رفتار سیستم ارائه نمی‌شود، بنابراین تحلیل‌های رفتاری در مورد نسخهٔ کنونی این زبان بی معنی می‌باشد. در نتیجه زبان MyxADL از این دسته از تحلیل‌ها حمایت نمی‌نماید.

۴-۴- ابزارهای پشتیبان

همانگونه که در بخش‌های پیشین نیز بدان اشاره شد، علی رغم آنکه ابزارهای پشتیبان جزء اصلی زبان‌های توصیف معماری محسوب نمی‌گردند، لکن برای قابل استفاده شدن آنها بسیار مهم می‌باشند. با توجه به این امر ابزارهای پشتیبانی لازم برای زبان MyxADL، نیز توسعه داده شده است. دستهٔ اول ابزارهای پشتیبانی این زبان شامل ویرایشگر[۱۶] خاص این زبان می‌گردد. برای تولید این ویرایشگر از ویرایشگر ArchEdit که در زبان xADL 2.0 توسعه یافته بود، استفاده شده است. توسط این ابزار امکان ایجاد فایل‌های توصیفی در زبان MyxADL، مدیریت و انجام برخی از تحلیل‌های ساختاری بر روی آن ایجاد می‌شود. در شکل ۳ نمایی از این وایرایشگر نشان داده شده است.

دستهٔ دوم ابزارهای پشتیبان زبان MyxADL مربوط به ابزار استفاده کننده از این زبان جهت تولید و مدیریت سیستم‌های قابل بازپیکربندی توصیف شده به زبان مذکور می‌باشد. ابزار مذکور که نسخهٔ توسعه یافتهٔ ابزار ArchStudio 3[۸] می‌باشد، قادر است با توجه به توصیفات انجام شده از ساختار سیستم، آن را ایجاد نماید. ابزار تولید شده دارای یک مدیر بازپیکربندی است که به کمک توصیفات صورت گرفته از شرایط و سیاست‌های بازپیکربندی به صورت خودکار سیستم قابل بازپیکربندی پویای ایجاد شده را مدیریت می‌نماید. در این راستا مدیر بازپیکربندی به کمک فایل‌های توصیف شرایط بازپیکربندی امکان تشخیص خودکار رخداد تغییر را در سیستم دارد. در صورت بروز یک رخداد تغییر، مدیر بازپیکربندی توسط توصیفات انجام شده از سیاست‌های بازپیکربندی، نقشهٔ بازپیکربندی را تعیین و تحلیل‌های ساختاری لازم را بر روی آن اجرا می‌نماید و پس از اطمینان از درستی توصیف جدید و برقراری محدودیت‌های توصیف شده در آن، نقشهٔ بازپیکربندی را بر روی سیستم نرم‌افزاری اِعمال می‌نماید. لذا نسخهٔ توسعه داده شده ArchStudio 3 به کمک مدیر بازپیکربندی ایجاد شده در آن در کنار زبان MyxADL قادر به توصیف، ایجاد و مدیریت سیستم‌های قابل بازپیکربندی پویا است.



شکل ۳- نمایی از ویرایشگر زبان MyxADL

۵- نتیجه‌گیری و نگاهی به آینده

در این مقاله با رائهٔ زبان توصیف معماری MyxADL یک زبان توصیف معماری برای سیستم‌های قابل بازپیکربندی پویا معرفی شد. این زبان، یک توسعه بر روی زبان توصیف معماری xADL 2.0 است. زبان مذکور علاوه بر ویژگی‌ها و مزایای به ارث برده از زبان xADL 2.0، قادر به توصیف تمامی نیازمندی‌های سیستم‌های قابل بازپیکربندی پویا از جمله بیان شرایط تغییر، سیاست‌های بازپیکربندی و محدودیت‌های سیستم نرم‌افزاری می‌باشد. همچنین در راستای قابل استفاده کردن زبان MyxADL مجموعه‌ای از ابزارهای پشتیبانی لازم از جمله یک ویرایشگر و یک ابزار مدیریتی برای تولید و مدیریت سیستم‌های قابل بازپیکربندی پویا در کنار این زبان ایجاد شده است.

زبان مذکور بر خلاف زبان‌های Rapide و Darwin که تنها از پویایی‌های محدود و از پیش تعیین شده حمایت می‌کنند، مانند زبان‌های Weaves، C2SADEL و ACME/Armani قادر به پشتیبانی از پویایی‌های موردی و از پیش تعیین شده است. از جمله مزیت‌های زبان MyxADL نسبت به زبان‌های C2SADEL و ACME/Armani می‌توان به توانایی این زبان در بیان ویژگی‌های کیفی نام برد. علاوه بر این برتری، زبان مذکور بر خلاف عدم توانایی زبان C2SADEL در بیان شرایط و سیاست‌های تغییر، قادر به توصیف کامل آنها می‌باشد. علی رغم آنکه قدرت بیان محدودیت‌ها در زبان ACME/Armani از زبان MyxADL بیشتر است، لکن مدیریت، ایجاد تغییر در زبان MyxADL به مراتب آسان‌تر از زبان مذکور صورت می‌پذیرد. همچنین زبان MyxADL دارای قابلیت توسعه بالایی است که این امر به خاطر پیمانه‌ای بودن آن و استفاده از شماهای XML می‌باشد.

در ادامه این کار تحقیقاتی می‌توان زبان توصیف معماری نرم‌افزار MyxADL را با تمرکز بر بحث مدلسازی رفتاری مؤلفه‌ها و رابط‌ها تکمیل کرد. پیرو این امر می‌توان وارسی و تحلیل‌های انجام شده در

زبان MyxADL را از تحلیل‌های صرف ساختاری به تحلیل‌هایی که شامل تحلیل‌های رفتاری نیز باشند، ارتقاء داد. همچنین در نسخهٔ کنونی زبان MyxADL با وجود امکان توصیف ویژگی‌های کیفی در فرایند بازپیکربندی به آنها توجه نمی‌شود. می‌توان با توسعه ابزارهای تحلیلی لازم بر روی ویژگی‌های کیفی و استفاده از آنها در فرایند بازپیکربندی، مدیر بازپیکربندی مورد استفاده در ابزار پشتیبان تولید شده برای زبان مذکور را، در تعیین سیاست‌های بازپیکربندی هوشمند کرد، به گونه‌ای که با توجه به مشخصه‌های کیفی و تعریف عملگرهای اولیه بازپیکربندی، خود بسته به شرایط قادر به تولید نقشه‌های بازپیکربندی باشد.

مراجع

[1] R., Allen, R., Douence, D., Garlan, "Specifying and Analyzing Dynamic Software Architectures," *Proc. of the 1998 Conference on Fundamental Approaches to Software Engineering* (FASE'98) Lisbon, Portugal, pp. 21-37, March 1998.

[2] J., Andersson, "Issues in Dynamic Software Architectures," *Proc. of the Int. Software Architecture Workshop* (ISAW-4) , pp. 111-114, Jun. 2000.

[3] Architectures Group at UC Irvine. *xADL 2.0, Highly-extensible Architecture Description Language for Software and System*, January 4, 2005; http://www.isr.uci.edu/projects/xarchuci/

[4] L., Bass, P., Clements, R., Kazman, *Software Architecture in Practic*, Addsion-Weisly, 2nd edition, 2003.

[5] T., Batista, A., Joolia, G., Coulson, "Managing Dynamic Reconfiguration in Component based Systems," *Second European Workshop on Software Architecture* (EWSA), pp. 1-17, 2005.

[6] J. S., Bradbury, J. R., Cordy, J., Dingel, M., Wermelinger, "A Survey of Self-Management in Dynamic Software Architecture Specifications," *Proc. of the 2nd ACM SIGSOFT workshop on Self-managed systems*, pp. 28-33, 2004.

[7] E., Dashofy, A., van der Hoek, R.N., Taylor, "Towards architecture-based self-healing systems," *Proc. of the first workshop on Self-healing systems* (woss2002) , pp. 21-26, Charleston, South Carolina, November 2002.

[8] E. M., Dashfoy, "ArchStudio 3, Software Architecture-Based Development Environment", March 28, 2005; http://www.isr.uci.edu/projects/archstudio/

[9] E.,Dashfy,"xArch", http://www.isr.uci.edu/architecture/xarch/

[10] A., Joolia, T., Batista, G., Coulson, A. T., Gomes, "Mapping ADL Specifications to an Efficient and Reconfigurable Runtime Component Platform," *Proc. 5th Working IEEE/IFIP Conference of Software Architecture* (WICSA 5), pp. 131-140, Pittsburgh, Pennsylvania, USA, November 2005.

[11] A., Ketfi, N. , Belkhatir, "A metamodel-based approach for the dynamic reconfiguration of component-based software," *Proc. Of the 8th International Conference on Software Reuse* (ICSR8), pp. 264-273, Madrid, Spain, July 2004.

[12] M., Makarem, *Formal Verification and Specification of Software Architecture Properties*, master's thesis, Dept. Computer Eng., Sharif University of Tech., 2006. (in Persian)

[13] N., Medvidovic, "ADLs and Dynamic Architecture Changes," *Proc. of the Second International Software Architecture Workshop* (ISAW-2), pp. 24-27, San Francisco, CA, October 1996.

[14] N., Medvidovic, N. R., Taylor, "A Classification and Comparison Framework for Software Architecture Description Languages," *IEEE Transactions on Software Engineering*, vol. 26, pp. 79-93, 2000.

[15] M., Moghadam," *Reconfigurable Systems*", Tech. report, Pervasive Computing Lab, Sharif University of Tec., 2003 (in Persian).

[16] M., Shaw, D., Garlan, "Formulations and Formalisms in Software Architecture," In Computer Science Today, vol. 1000 of Lecture Notes in Computer Science, Springer-verlang, pp. 307–323, 1995.

[17] F. J., Silva, M., Endler, F., Kon, "A Framework for Building Adaptive Distributed Applications," *Proc. International Middleware Workshops*, pp.110-114, Rio de Janeiro, Brazil, 2003.

[18] W.T., Tsai, Y., Chen, Z., Cao, X., Bai, H., Huang, R. J., Paul, "Services-Oriented Dynamic Reconfiguration Framework for Dependable Distributed Computing," Proc. Of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), pp. 554-559, 2004.

---

1 Adaptability
2 Replication
3 Infrastructure
4 Architecture Description Language (ADL)
5 Connector
6 Configuration
7 Tool Support
8 Consistency
9 Constrained Dynamism
10 Application Programmable Interface (API)
11 Architectural Information
12 Architecture Tradeoff Analysis Method :
یک روش ارزیابی معماری با تمرکز بر ویژگی‌های کیفی
13 Ad hoc
14 Correctness
15 Completeness
16 Editor