نقش معماری مدل‌رانه در یکپارچه‌سازی سامانه‌های فوق مقیاس وسیع: چالش‌ها و راهکارها

سید شروین استادزاده[۱]، فریدون شمس[۲]

[۱] دانشجوی دکترای تخصصی مهندسی نرم‌افزار، گروه مهندسی کامپیوتر، دانشکدهٔ فنی و مهندسی، واحد علوم و تحقیقات دانشگاه آزاد اسلامی
تهران، ایران
ostadzadeh@sr.iau.ac.ir

[۲] دانشیار، گروه مهندسی کامپیوتر، دانشکدهٔ مهندسی برق و کامپیوتر، دانشگاه شهید بهشتی
تهران، ایران
f_shams@sbu.ac.ir

چکیده

چندی قبل موسسه مهندسی نرم‌افزار گزارشی را در ارتباط با چالش‌های امروزی توسعه فناوری اطلاعات به سفارش وزارت دفاع امریکا ارائه کرد که در آن به بررسی خصوصیات و چالش‌های سامانه‌های با مقیاس فوق وسیع پرداخته شده است. توجه به بحث مقیاس در سامانه‌های امروزی که ما را به سمت سامانه‌های فوق وسیع سوق می‌دهد، بسیار تامل برانگیز است. سامانه‌های با مقیاس فوق وسیع دارای ویژگی‌هایی هستند که باعث می‌شوند رویکرد فعلی روش‌های مهندسی نرم‌افزار نتوانند پاسخگوی نیازمندیهای توسعه آنها باشند. این ویژگی‌ها عمدتاً ناشی از «مقیاس» این گونه از سامانه‌ها است، که چالش‌های جدی را فراروی توسعه آنها ایجاد می‌کند. در این مقاله قصد داریم این چالش‌ها را مورد بررسی قرار داده، و از معماری مدل‌رانه به عنوان راهکاری اولیه برای حل برخی از آنها در حوزه یکپارچه‌سازی استفاده کنیم.

واژه‌های کلیدی

سامانه‌های فوق مقیاس وسیع (ULS) ، معماری مدل‌رانه (MDA)، یکپارچه‌سازی، معماری نرم‌افزار، مدیریت فناوری اطلاعات.

۱- مقدمه

یکی از اهداف وزارت دفاع آمریکا دستیابی به یک سلطه اطلاعاتی است [۱] تا به کمک آن بهره‌برداری و استفاده از جمع‌آوری، ترکیب، تحلیل و به کارگیری اطلاعات را در راستای اهداف ماموریتی خود محقق سازد. این هدف به شدت وابسته به سامانه‌های پیچیده است که شامل هزاران سکو، حسگر، گره‌های تصمیم‌گیری و جنگ‌افزاری است که از طریق شبکه‌های سیمی و بدون سیم ناهمگن به یکدیگر متصل شده‌اند. اندازه چنین سامانه‌هایی از هر نظر بسیار فراتر از مقیاس سامانه‌های امروزی است: از نظر تعداد خطوط کد برنامه؛ افراد درگیر در سامانه؛ داده‌هایی ذخیره شده، بازیابی شده، دستکاری شده و پالایش شده؛ میزان اتصالات و وابستگی بین واحدی مولفه‌های نرم‌افزاری؛ عناصر سخت‌افزاری؛ و ... . به چنین سامانه‌هایی، سامانه‌های با مقیاس فوق وسیع[۱] (ULS) گفته می‌شود.

مقیاس در سامانه‌های فوق وسیع باعث تغییر همهٔ چیز می‌شوند. سامانه‌های فوق وسیع لزوماً به شکل نامتمرکز هستند؛ توسط تعداد زیادی از ذینفعان با نیازهای متضاد، توسعه و به کار گرفته می‌شوند؛ به طور مستمر تکامل پیدا می‌کنند؛ و از قطعات ناهمگن تشکیل می‌شوند. افراد تنها کاربران سامانه‌های فوق وسیع نیستند، بلکه عنصری از سامانه محسوب می‌شوند. خرابی‌های نرم‌افزاری و سخت‌افزاری یک امر کاملاً عادی محسوب می‌شوند و نمی‌توان آنها را یک استثناء در نظر گرفت. سامانه‌های فوق وسیع همزمان مورد استفاده قرار می‌گیرند و نیاز به روش‌های نوین برای کنترل دارند. این ویژگی‌ها لزوم به کارگیری روش‌هایی را برای استفاده، تولید، استقرار، مدیریت، مستندسازی و تکامل سامانه‌های فوق وسیع اجتناب‌ناپذیر می‌سازد.

دفتر معاونت وزیر ارتش امریکا - شاخه تدارکات و فناوری[۲] (ASA ALT) پروژهٔ تحقیقاتی را برای موسسه مهندسی نرم‌افزار[۳] (SEI) تعریف کرد که نتیجه آن یک تحقیق ۱۲ ماهه بر روی

نرم‌افزارهای سامانه‌های فوق وسیع بود. وزارت دفاع به طور مشخص این سوال را مطرح کرده بود: «قابلیت رویکردهای فعلی مهندسی نرم‌افزار را در ساخت سامانه‌های آینده که شامل بیلیون‌ها خط کد برنامه است، چگونه ارزیابی می‌کنید؟». پاسخ به این سوال گزارشی [۱] شد که به چالش‌های فراروی توسعه سامانه‌های ULS و زمینه‌های تحقیقاتی مرتبط با آن می‌پردازد. اگرچه مسئله بیلیون‌ها خط کد برنامه یک چالش اولیه محسوب می‌شد، با این حال، افزایش اندازه کد برنامه باعث افزایش مقیاس در بسیاری از زمینه‌های دیگر خواهد شد. این مقیاس فوق وسیع چالش‌های فراوانی را ایجاد خواهد کرد که بر روی بسیاری از مباحث مهندسی نرم‌افزار تاثیرات شگرفی را خواهد گذاشت.

در این مقاله قصد داریم ضمن بررسی چالش‌های فراروی توسعه سامانه‌های با مقیاس فوق وسیع، راهکارهایی مبتنی بر معماری مدل‌رانه برای برخی از آنها ارائه کنیم. معماری مدل‌رانه یکی از آخرین استانداردهای گروه مدیریت شیء[۴] (OMG) است. تحقیقات ما نشان می‌دهد این استاندارد قابلیت بسیار خوبی در توسعه معماری‌های سازمانی و سامانه‌های اطلاعاتی بین سازمانی دارد [۴،۳،۲]. سعی خواهیم کرد راهکارهایی را مبتنی بر این استاندارد برای کمک به یکپارچه‌سازی سامانه‌های ULS ارائه کنیم. اگرچه لازم به ذکر است چالش‌های فراروی توسعه سامانه‌های فوق وسیع بسیار فراتر از استانداردهای موجود هستند. با این حال، می‌توان انتظار داشت استانداردهای موجود راهکارهای اولیه را برای این چالش‌ها ارائه کنند و در تحقیقات بعدی با غنی‌سازی و تقویت استانداردها به راه‌حل‌های قطعی برای چالش‌های ذکر شده دست یافت.

ساختار ادامه این مقاله به شکل زیر است: در بخش ۲ به معرفی ویژگی‌ها و چالش‌های فراروی سامانه‌های فوق وسیع می‌پردازیم. در ادامه (بخش ۳) نگاهی اجمالی به معماری مدل‌رانه خواهیم داشت. بخش ۴ بحث اصلی این مقاله است که در آن راهکارهای پیشنهادی مبتنی بر معماری مدل‌رانه برای چالش‌های یکپارچه‌سازی سامانه‌های ULS تشریح می‌شود. در پایان (بخش ۵)، نتیجه‌گیری و پیشنهاد برای کارهای تحقیقاتی آتی انجام شده است.

۲- سامانه‌های با مقیاس فوق وسیع

گزارش موسسه مهندسی نرم‌افزار در ارتباط با سامانه‌های فوق وسیع [۱] به بررسی ویژگی‌ها، چالش‌ها و زمینه‌های تحقیقاتی آتی در این حوزه می‌پردازد. در این بخش به بیان ویژگی‌ها و چالش‌های فراروی توسعه سامانه‌هایی با مقیاس فوق وسیع می‌پردازیم. این بخش می‌تواند تصویری روشن را از سامانه‌های فوق وسیع به خوانندگان ارائه نماید.

۲-۱- ویژگی‌های سامانه‌های ULS

سامانه‌های با مقیاس فوق وسیع دارای ویژگی‌هایی هستند که باعث می‌شوند رویکرد (فعلی و مورد استفاده) روش‌های مهندسی نرم‌افزار نتوانند پاسخگوی توسعه آنها باشند. این ویژگی‌ها عمدتاً ناشی از «مقیاس» این گونه از سامانه‌ها است. البته ماهیت سامانه‌های ULS به مواردی فراتر از «اندازه» آنها برمی‌گردد. در واقع، اندازه باعث می‌شود بسیاری از مواردی که در سامانه‌های معمولی غیرمهم یا کم اهمیت بودند، تبدیل به موارد بااهمیت شوند. مشکلات ناشی از مقیاس، نیازمند روش‌های حل جدید و تعریف مفاهیم نو برای طراحی، توسعه، کارکرد، و تکامل سامانه‌ها است. می‌توان ۷ ویژگی را برای سامانه‌های با مقیاس فوق وسیع در نظر گرفت. در ادامه، ضمن بیان این ویژگی‌ها، مشخص می‌کنیم چرا هر یک از آنها باعث می‌شود رویکرد فعلی مهندسی نرم‌افزار در ارضاء آن ویژگی ناتوان باشد.

۲-۱-۱- کنترل نامتمرکز

مقیاس سامانه‌های ULS تنها به شکل بسیار محدودی اجازه کنترل مرکزی و سلسله مراتبی داده، توسعه، تکامل، و کارکرد را می‌دهد. حتی مقدار محدود کنترل سلسله مراتبی که امروزه در سامانه‌های بسیار بزرگ امکان‌پذیر است، در سامانه‌های ULS مورد تردید است، و در نتیجه مدل‌های متفاوتی را برای کنترل طلب می‌کند. از میان مفروضاتی که امروزه در صنعت توسعه نرم‌افزار وجود دارند و توسط این ویژگی زیر سوال می‌روند، می‌توان به مورد زیر اشاره کرد:

- **تمام تضادها باید متحدالشکل رفع شده باشند.** سامانه‌های امروزی بر اساس این ایده شکل می‌گیرند که همه تضادها باید شناسایی و رفع شده باشند. انتظار داریم یک فرایند تحلیل و رفع تضاد، و نیز یک سازمان تصمیم‌گیری در مورد آن وجود داشته باشد. این در حالی است که مقیاسِ سامانه‌های ULS رفع همهٔ تضادها و رفع آنها به شکل متمرکز را برای ما غیرممکن می‌سازد. در یک اکوسیستم، هیچ مجوز مرکزی برای حل تضاد وجود ندارد. برای انجام این کار باید از سایر مکانیزم‌ها (ی غیرمرکزی) استفاده کرد. در واقع این مکانیزم‌ها به صورت محلی عمل می‌کنند. همچنین، یک نوع تضاد ممکن است در جاهای مختلف یک سامانه ظاهر شود و مکانیزم رفع آنها نیز متفاوت باشد. طراحان سامانه‌های بزرگ امروزی معمولاً چنین تصوری ندارند که تحمل تضاد و رفع آن در زمان‌ها و مکان‌های مختلف ممکن است متفاوت باشد. همچنین، آنها معمولاً روال‌هایی را برای تعیین تضادهایی که می‌توانند مهمترین تاثیر را بر دوام کلی سامانه داشته باشند، در نظر نمی‌گیرند. البته واقعیت این است که تعیین چنین روال‌هایی چندان هم ساخت‌یافته نیست.

۲-۱-۲- نیازمندیهای ذاتاً متضاد و ناشناخته

مقیاس و پیچیدگی مسائلی که سامانه‌های ULS باید حل کنند، اغلب ما را به وضعیتی سوق می‌دهند که در آن نیازمندیهای یک سامانه تا زمان استفاده از آن سامانه ناشناخته‌اند. حتی، گاهی پس از آن که سامانه مورد نظر عملیاتی شد، درک ما از مسئله دچار تغییر می‌شود. هر

تلاش برای حل مسئله در واقع فهم ما را از مسئله بیشتر می‌کند، و باعث می‌شود مسئله جدیدی مطرح شده و نیاز به تلاشی دیگر برای حل آن باشد. به این شکل، بسیاری از مسائلی که سامانه‌های ULS باید حل کنند پایان‌پذیر نیستند. از طرف دیگر، سامانه‌های ULS به دلیل اندازه و ماهیت‌شان باید طیف وسیعی از نیازمندیها را ارضاء کنند. هر چقدر دامنه این نیازمندیها وسیع‌تر باشد، تنوع و تضاد در بین آنها افزایش می‌یابد. همچنین، یکپارچگی راه‌حل‌ها نیاز به دانش در حوزه‌های مختلف و در نتیجه یک دانش بین دامنه‌ای دارد، که به دست آوردن آن چندان ساده نیست. از میان مفروضاتی که امروزه در صنعت توسعه نرم‌افزار وجود دارند و توسط این ویژگی زیر سوال می‌روند، می‌توان به موارد زیر اشاره کرد:

- **نیازمندیها می‌توانند از قبل شناسایی شوند و به تدریج با کسب تجربه کار با سامانه گسترش یابند.** امروزه، نیازمندیهای یک سامانه از قبل به طور کامل شناسایی نمی‌شوند، و بسیاری از سامانه‌ها تنها با شناسایی و ارضاء نیازمندیهای کلیدی ساخته می‌شوند. اما در سامانه‌های ULS ما اصطلاحاً با مسائل شروری[5] مواجه هستیم که نه تنها نیازهای آنها شناخته شده نیست (چون توافقی بر روی خود مسئله وجود ندارد)، بلکه ثابت هم نیست (چون هر راه‌حل مسئله را تغییر می‌دهد، و هیچ راه‌حلی مسئله را کاملاً حل نمی‌کند).
- **تصمیمات مربوط به مصالحه بین نیازمندیها پایدار هستند.** در یک سامانه ULS، تعداد ذینفعان بسیار بالاست و نیاز به مصالحه‌های متفاوتی وجود دارد که در طول زمان نیز تغییر می‌کنند. یک سامانه امروزی معمولاً با این فرض ساخته نمی‌شود که مصالحه‌های مختلف برای کاربران گوناگون تغییر کند.

۲-۱-۳- تکامل و استقرار مداوم

یکی دیگر از پیامدهای «اندازه» این است که سامانه‌های ULS برای مدت طولانی باید به ارائه خدمات بپردازند. در واقع، اندازه این نوع از سامانه‌ها جایگزینی یا از رده خارج شدن آنها را غیرممکن می‌سازد. سامانه‌های ULS نیز همانند سامانه‌های بسیار بزرگ امروزی به طور مداوم تکامل پیدا می‌کنند تا نیازمندیهای جدید و تغییریافته را برآورده کنند. با این حال، ما به تکاملی متفاوت از تکامل در سامانه‌های بسیار بزرگ امروزی نیاز داریم. هنگامی که از تکامل یک سامانه صحبت می‌کنیم، منظورمان تغییرات هدایت‌شده‌ای است که بر اساس قواعد و سیاست‌ها و به شکل محلی انجام می‌شود، بدون آنکه یکپارچگی آن سامانه را از بین ببرد. اما، یکپارچگی در سامانه‌های با مقیاس فوق وسیع توسط گروه‌های مختلفی از ذینفعان انجام می‌شود. هیچ تضمینی وجود ندارد که این تغییرات کاملاً قاعده‌مند بوده و بر اساس قواعد از پیش‌تعریف شده انجام پذیرند. از میان مفروضاتی که امروزه در صنعت توسعه نرم‌افزار وجود دارند و توسط این ویژگی زیر سوال می‌روند، می‌توان به مورد زیر اشاره کرد:

- **بهینه‌سازی سامانه در فاصله‌های زمانی گسسته (ساخت- بهره‌برداری- ساخت) مطرح می‌شوند.** سامانه‌های ULS همچنان که به کار خود ادامه می‌دهند باید اصلاح، بهینه، و بازسازی شوند. در بسیاری از سامانه‌های امروزی، ما معمولاً به گروه‌های مختلف اجازه انجام تغییرات را به شکل همزمان نمی‌دهیم. اما، در سامانه‌های ULS اغلب لازم است تغییرات به شکل موازی انجام شوند، زیرا انتظار برای انجام تغییرات به شکل زنجیره‌ای غیرقابل قبول است.

۲-۱-۴- عناصر ناهمگن، ناسازگار، و در حال تغییر

اندازه سامانه‌های ULS بدین معنی است که عناصر آن (همچون سخت‌افزار، نرم‌افزار، روال‌ها، قواعد، افراد، و ...) ناهمگن، ناسازگار و در حال تغییر هستند. عناصر نرم‌افزاری به دلیل گوناگون بودن منابع آنها ناهمگن هستند (زبان‌های برنامه‌سازی متفاوت، سکوهای مختلف، متدولوژی‌های متفاوت، و ...). از آنجا که ایجاد نرم‌افزارها نیز در شرایط (مکان‌ها، زمانبندی‌ها، فرایندها، اهداف، ذینفعان، و ...) متفاوت انجام شده است، احتمالاً در طراحی، ساخت، و بهره‌برداری با یکدیگر ناسازگارند. بخش‌های مختلف یک سامانه همواره در حال تغییر هستند. محیط عملیاتی تغییر می‌کند؛ بخش‌های خراب سخت‌افزار باید جایگزین شوند؛ نرم‌افزارها و سخت‌افزارها به روز می‌شوند؛ و پیکربندی مولفه‌ها اصلاح می‌شوند. از میان مفروضاتی که امروزه در صنعت توسعه نرم‌افزار وجود دارند و توسط این ویژگی زیر سوال می‌روند، می‌توان به موارد زیر اشاره کرد:

- **تاثیر یک تغییر می‌تواند به قدر کفایت پیش‌بینی شود.** فرض بر این است که در زمان تغییر یک عنصر، اطلاعات کافی درباره ضروری بودن یا غیرضروری بودن آن عنصر وجود دارد. بر اساس همین اطلاعات می‌توان جایگزینی‌ها را انجام داد، و تاثیرات آنها را از قبل دانست. به دلیل مقیاس در سامانه‌های ULS تاثیرات مربوط به عنصر اصلی و عنصر جایگزین‌شده به روشنی مشخص نیست.
- **اطلاعات پیکربندی دقیق و به شدت کنترل‌پذیر هستند.** هنگامی که می‌خواهیم تاثیر تغییرات را در یک سامانه امروزی مشخص کنیم، برای فرایند تغییر باید پیکربندی یک سامانه کاملاً شناخته‌شده و قابل شناسایی باشد. در سامانه‌های ULS دقت اطلاعات پیکربندی کاملاً مشخص نیستند، و تغییرات باید بدون در نظر گرفتن دقت این اطلاعات انجام شوند.
- **مولفه‌ها و کاربران نسبتاً همگن هستند.** در طراحی و پیاده‌سازی سامانه‌های امروزی، انتظار داریم توانایی کاربران و چگونگی استفاده آنها از سامانه مورد نظر مشخص باشند. اما در

سامانه‌های ULS به دلیل وسعت و گوناگونی افراد درگیر، چگونگی استفاده از سامانه باید با در نظر گرفتن این موارد لحاظ شود. همچنین، به دلیل مقیاس سامانه‌های ULS، اطمینان از درستی منطق مولفه استفاده شده بسیار دشوارتر است. در طراحی سامانه‌های فوق وسیع باید به این ناهمگونی‌ها که معمولاً در ساخت سامانه‌های امروزی در نظر گرفته نمی‌شوند، توجه داشت.

۲-۱-۵- از بین رفتن تدریجی مرز بین افراد و سامانه

افراد نه تنها کاربران یک سامانه ULS هستند، بلکه بخشی از رفتار کلی آن نیز محسوب می‌شوند. در واقع، مرز بین سامانه و نقش‌های کاربر/توسعه‌دهنده به روشنی مشخص نیست. یک شهر را در نظر بگیرید. افرادی که در این شهر ساکن هستند، ممکن است تغییر و نگهداری آن را نیز به عهده داشته باشند. در واقع یک شخص نقش‌های متفاوتی دارد. در یک سامانه ULS نیز چنین وضعیتی رخ می‌دهد. برخی از مواقع یک شخص یک کاربر است، در زمانی دیگر یک نقش نگهداشت را بر عهده دارد، و وقت دیگر عملکردهای سامانه را اضافه و کم یا تصحیح می‌کند. در نظر گرفتن افراد به عنوان بخشی از سامانه‌های ULS به این معنی است که با تغییر توانایی‌های محاسباتی و پیکربندی سامانه‌ها باید فرایندها و روال‌های مربوطه جهت کمک به درک افراد در راستای اهداف و ماموریت‌های آنها اصلاح شود. زیرا افراد بخشی از خود سامانه ULS هستند. از میان مفروضاتی که امروزه در صنعت توسعه نرم‌افزار وجود دارند و توسط این ویژگی زیر سوال می‌روند، می‌توان به موارد زیر اشاره کرد:

- **افراد تنها کاربر سامانه هستند.** درک و تحلیل تاثیرات برخی از جنبه‌های سامانه‌های ULS، بدون در نظر گرفتن رفتار افراد به عنوان عناصر آن غیرممکن است. در نظر گرفتن رفتار انسانی در تحلیل کلی وظایف یک سامانه موضوع جدیدی نیست. با این حال، در سامانه‌های ULS این مسئله یک موضوع کاملاً ضروری و حیاتی محسوب می‌شود.
- **رفتار جمعی افراد مورد توجه نیست.** هنگامی که صحبت طراحی و تحلیل سامانه‌های ULS مطرح می‌شود، به دلیل مقیاس بسیار بالای آنها، رفتار جمعی گروه کاربران و توسعه‌دهندگان یک عامل بسیار مهم در چگونگی نگرش، پذیرش، و استفاده از سامانه محسوب می‌شود.
- **تعاملات اجتماعی یک موضوع مرتبط نیستند.** امروزه، تاکید طراحی سامانه‌های اطلاعاتی تنها بر روی فناوری است (چگونه سامانه‌ها را به اندازه کافی سریع، مطمئن، وظیفه‌مند، و ... کنیم). معمولاً یک دیدگاه اجتماعی- فنی[۶] وجود ندارد. در سامانه‌های ULS، چگونگی استفاده گروه کاربران از فناوری و نیز چگونگی پشتیبانی فناوری از نیازهای گروه کاربران اهمیت دارد. در نظر نگرفتن این موارد یک خطای بزرگ محسوب می‌شود.

۲-۱-۶- خرابی‌های طبیعی

از آنجا که زیربنای فیزیکی یک سامانه ULS بسیار وسیع است، خرابی سخت‌افزار دیگر یک امر غیرعادی نیست بلکه به طور طبیعی اتفاق می‌افتد. همچنین، از آنجا که مولفه‌های نرم‌افزاری فراتر از ظرفیتی که طراحی شده‌اند تحت فشار قرار می‌گیرند، رفتار آنها نیز ممکن است نامطلوب باشد که این مسئله نیز کاملاً عادی است. فرض کنید یک پروتکل ارتباطی در هر یک میلیون انتقال فایل یکبار با شکست مواجه می‌شود. اگر انتقال فایل یک میلیون بار در روز اتفاق افتد، بطور متوسط یک شکست در روز خواهیم داشت. در سامانه‌های ULS معمولاً بروز خرابی آنقدر متداول است که در واقع می‌توان گفت خطا «همیشه» رخ می‌دهد. با توجه به مقیاس چنین سامانه‌هایی، مسئله خرابی باید به شکل یک مشکل پیوسته در طراحی و ساخت لحاظ شود. از میان مفروضاتی که امروزه در صنعت توسعه نرم‌افزار وجود دارند و توسط این ویژگی زیر سوال می‌روند، می‌توان به موارد زیر اشاره کرد:

- **خطا بندرت رخ می‌دهد.** مقیاس در سامانه‌های ULS، تعداد خرابی‌ها را در واحد زمان افزایش می‌دهد. با افزایش اندازه و کاربرد یک سامانه ULS، وقوع خطا امری کاملاً «طبیعی» است. در حالی که در سامانه‌های امروزی بروز خطا یک امر غیرمتداول و استثنایی در نظر گرفته می‌شود.
- **می‌توان خطاها را برطرف کرد.** بسیاری از روش‌های مهندسی نرم‌افزار علاقمند به پیشگیری و تشخیص خطاها هستند. اگرچه در سامانه‌های بسیار بزرگ نمی‌توان اطمینان حاصل کرد که همهٔ خرابی‌ها برطرف شده‌اند، در سامانه‌های ULS، این مسئله بغرنج‌تر است. سامانه‌های ULS باید به موضوع تحمل خطا توجه بیشتری از آنچه امروز به آن می‌شود، داشته باشند (البته باید سامانه‌های خاصی را که به دلیل شرایط استفاده از آنها نیاز به ضریب اطمینان بسیار بالا دارند مستثنی کرد).

۲-۱-۷- پارادایم‌های جدید برای استفاده و سیاست‌گذاری

به دلیل اندازه سامانه‌های ULS، افرادی که مسئول ساخت آنها هستند (احتمالاً مدیران، توسعه‌دهندگان، فروشندگان، و ...) نمی‌توانند به طور قطعی تعریف شوند. نمی‌توان نیازمندیهای متغیر و غیرقطعی ذینفعان را کنترل کرد. نمی‌توان نیازمندیها را به شکل متمرکز و سراسری نظارت کرد. در حقیقت اندازه سامانه‌های ULS باعث بروز یک چالش اساسی برای مدیران می‌شود. اگر نتوان نیازمندیهای واقعی ذینفعان را به طور کامل مشخص کرد، چگونه می‌توان فرایند بستن قرارداد، طراحی، و ساخت را کنترل کرد. از میان مفروضاتی که امروزه در صنعت توسعه نرم‌افزار وجود دارند و توسط این ویژگی زیر سوال می‌روند، می‌توان به مورد زیر اشاره کرد:

- **یک پیمانکار اصلی مسئول توسعه، راه‌اندازی، و تکامل سامانه است.** کنترل مرکزی که معمولاً در مدل‌های امروزی توسعه نرم‌افزار وجود دارد و توسط پیمانکار اصلی انجام می‌شود، با سامانه‌های ULS هماهنگی ندارد. به عنوان مثال، هیچ پیمانکار اصلی مسئول توسعه، راه‌اندازی، و تکامل اینترنت نیست. بلکه سازمان‌های مختلف، مسئولیت‌های متفاوتی در ایجاد اینترنت دارند. در واقع یک روش کنترل غیرمتمرکز و محلی وجود دارد.

۲-۲- چالش‌های توسعه سامانه‌های ULS

در بخش قبل ویژگی‌هایی را عنوان کردیم که باعث می‌شوند رویکرد (فعلی و مورد استفاده) روش‌های مهندسی نرم‌افزار برای توسعه سامانه‌های ULS با چالش‌های جدی روبرو شوند. می‌توان چالش‌های فراروی توسعه این سامانه‌ها را در سه حوزهٔ کلی مورد بررسی قرار داد: (۱) طراحی و تکامل، (۲) هم‌نواسازی[۷] و کنترل، و (۳) نظارت و ارزیابی.

۲-۲-۱- چالش‌های حوزهٔ طراحی و تکامل

چگونه می‌توان به شکل سیستماتیک ویژگی‌های اکوسیستمی فنی-اجتماعی سامانه‌های ULS را برآورده کرد؟ چگونه می‌توان زیرساخت‌های اکوسیستمی را طراحی کرد که شامل: سرویس‌های فراهم شده و به اشتراک گذاشته شده یک سامانه؛ قواعد (رسمی و اجتماعی) هدایت رفتار آن سامانه؛ نحوه استفاده عملی آن سامانه؛ زیرساخت‌های زنجیرهٔ تامین آن سامانه؛ موارد بااهمیت اقتصادی؛ و ... باشند؟ چگونه می‌توان فرایندهای سازمانی مربوط به تولید و به روز رسانی مستمر مولفه‌های سامانه‌های ULS و یکپارچه‌سازی آنها را طراحی کرد؟

درجه پیچیدگی و عدم قطعیت طراحی سامانه‌های ULS به اندازه‌ای است که روش‌های سنتی توسعه که معمولاً بر روی کنترل متمرکز تاکید دارند، قادر به کنترل آن نیستند. حتی برخی از روش‌های نامتمرکز همچون توسعه متن‌باز نیز نمی‌توانند آن را مدیریت کنند. باید روش‌های جدیدی را پیدا کرد که نه تنها شرکت‌های منفرد، پیمانکاران اصلی، زنجیره‌های تامین را هماهنگ کنند، بلکه قادر باشند همهٔ صنایع را همگام و متحد سازند. این مسئله باعث انفجار فضای طراحی می‌شود و بخش‌هایی را شامل می‌شود که امروزه به هیچ عنوان با آنها درگیر نبودیم. نگهداشت یکپارچگی مفهومی که در طراحی چنین سامانه‌هایی مطرح می‌شود فراتر از فعالیت‌های طراحی امروزی است. در واقع چالش‌های جدیدی در دانش، ابزار، و روش‌های امروزی مطرح می‌شوند. طراحی اکوسیستم‌ها نیاز به تفکر و تحقیقات جدید دارد که از همهٔ سطوح طراحی سامانه‌های ULS پشتیبانی کند.

سامانه‌های ULS در دنیای واقعی ما کاملاً تعبیه شده‌اند. این سامانه‌ها نه تنها از مولفه‌های فناوری اطلاعات تشکیل شده‌اند، بلکه شامل انواع مختلفی از ماشین‌ها، افراد و گروه‌ها، حسگرهای گوناگون، جریان‌های اطلاعاتی، و ... هستند. به طور سنتی ما نرم‌افزار را به شکل برنامه‌نویسی مولفه‌های کامپیوتری می‌بینیم. در حالی که باید دیدگاه‌مان را نسبت به نرم‌افزار و طراحی آن تغییر دهیم: نرم‌افزار عبارت است از برنامه‌نویسی همهٔ مکانیزم‌های پردازش اطلاعات و رفتارهای سامانه‌های ULS پیچیده.

افزایش توانایی‌های سامانه‌های امروزی و حرکت به سمت سامانه‌های ULS نیاز به فرایندهای طراحی و مهندسی مجدد دارد. نظریه‌های جدید در تکامل سامانه‌های ULS، قواعد و مکانیزم‌هایی را ایجاد می‌کنند که به انجام تکامل موثر ختم می‌شوند. با این حال، این نظریه‌ها فراتر از تحقیقاتی است که امروزه در زمینهٔ مدیریت تغییرات در حال انجام است.

۲-۲-۲- چالش‌های حوزهٔ هم‌نواسازی و کنترل

هم‌نواسازی یعنی مجموعه‌ای از فعالیت‌ها که مولفه‌های یک سامانه ULS را در جهت ارضاء اهداف ماموریتی به شکل معقولی با یکدیگر هماهنگ کنند. هم‌نواسازی با مدیریت و کنترل سروکار دارد، اما در مقیاس‌هایی فراتر از کنترل‌های سنتی، متمرکز، و یکدست. هم‌نواسازی به ترکیبی از طراحی پیش‌رو، ترویج و اجراء سیاست کلی، و تنظیم بلادرنگ پارامترهای عملیاتی نیازمند است.

هم‌نواسازی یک سامانه ULS نیاز به پشتیبانی از وابستگی متقابل و کنترل نتایج منطقی اقدامات محلی نسبت به تاثیرات کلان دارد. یک شهر را در نظر بگیرید. در یک شهر گروه‌های مختلف ممکن است به سرویس‌هایی بیش از آنچه وجود دارند، نیاز داشته باشند. در این حالت، معمولاً روال‌هایی برای تصمیم‌گیری برای سرویس‌های جدید وجود دارد. حکومت شهر اقدامات هر یک از شهروندان را کنترل نمی‌کند، اما قوانین عمومی را مشخص می‌کند تا تضاد، شکست، و سوء استفاده از منابع را به حداقل برساند. به طور مشابه این قوانین و سیاست‌ها باید در یک سامانه ULS تعریف شوند تا تعامل اجزاء آن را مقید سازند. این قوانین و سیاست‌ها چارچوبی را فراهم می‌کنند تا تداوم و انطباق یک سامانه ULS را در دنیایی با ماموریت‌ها و وظایف متغیر امکانپذیر سازد.

هم‌نواسازی در همهٔ سطوح سامانه‌های ULS مورد نیاز است. در یک سطح، فعالیت‌های خودکارسازی مبتنی بر فناوری‌های کلیدی باید هماهنگ شوند. در سطح دیگر، رفتار عملیاتی یک سامانه ULS نیازمند هماهنگی در تخصیص منابع به صورت بلادرنگ دارد. در بالاترین سطح، هم‌نواسازی می‌تواند بر روی اصول، سیاست‌ها، و فرصت‌ها انجام شود.

۲-۲-۳- چالش‌های حوزهٔ نظارت و ارزیابی

اثربخشی طراحی، تکامل، و هم‌نواسازی سامانه ULS باید ارزیابی شود. باید امکان نظارت و ارزیابی حالت، رفتار، و سلامت کلی یک سامانه ULS فراهم شود. نظارت و ارزیابی سامانه‌های پیچیده، و در نتیجه تنظیم پارامترهای سامانه، یک ایده جدید نیست. یک شهر را در نظر

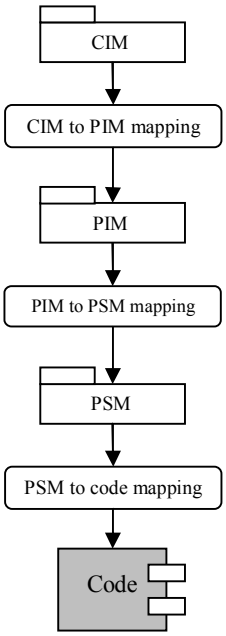
بگیرید. حسگرهایی می‌توانند اطلاعات وضعیت ترافیک را جمع‌آوری کنند. این اطلاعات به افراد مرتبط انتقال داده می‌شود، و آنها نیز می‌توانند بر اساس این اطلاعات تصمیم‌گیری کنند. مثلاً یک مسیر خاص را انتخاب کنند. در شبکه‌های مخابراتی، حمل‌ونقل، و توزیع برق نیاز به اندازه‌گیری و ارزیابی مستمر وجود دارد. اندازه‌گیری با نظارت بر حسگرهای تعبیه‌شده انجام می‌شود. سپس فعالیت‌های ارزیابی مشخص می‌کنند این اندازه‌ها چه معنایی دارند.

معیارهای موفقیت و سلامت کلی در سامانه‌های ULS بسیار متفاوت از سامانه‌های کوچکی است، که یک وظیفه در آنها بدون تغییر در زمان اجرا انجام می‌شود. به عنوان مثال، معیارهای موفقیت یک شهر را در نظر بگیرید. برخی از آنها مربوط به حکومت و برخی دیگر مربوط به شهروندان است. اگر برق در بخشی از شهر قطع شود، سایر بخش‌ها می‌توانند به کار خود ادامه دهند. دست‌کم تا زمانی که قطعی برق خیلی طول نکشد؛ در بخش‌های بحرانی شهر رخ ندهد؛ و به کرارت نباشد. زیرا، سرویس‌ها در یک شهر برای افراد مختلف توزیع شده است.

امروزه روش اصلی نظارت و ارزیابی سامانه‌ها از طریق به کارگیری معیارهایی انجام می‌شود. کیفیت یا وظیفه‌مندی یک سامانه از طریق مجموعه‌ای از معیارهای حیاتی و اندازه‌گیری آنها تعریف می‌شود. متاسفانه این روش برای سامانه‌های ULS مناسب نیست. بزرگی مقیاس، عدم تمرکز، توزیع‌شدگی، و ناهمگنی سامانه‌های ULS چالش‌هایی را پیش روی نظارت و ارزیابی موثر ایجاد می‌کنند. برای ارزیابی و نظارت درست سامانه‌های ULS، نیازمند تقویت روش‌های اندازه‌گیری جاری هستیم. همچنین، از آنجا که سامانه‌های ULS سامانه‌هایی اجتماعی- فنی هستند و افراد نیز در آن شرکت دارند، شاخص‌های نظارت و ارزیابی نه تنها باید وضعیت فنی را نشان دهند، بلکه باید عناصر انسانی، سازمانی، اقتصادی، و تجاری را نیز در نظر بگیرند.

۳- معماری مدل‌رانه[۸] (MDA)

معماری مدل‌رانه آخرین استاندارد گروه مدیریت شی است. معماری مدل‌رانه یک روش مدل‌سازی معماری سازمانی است که تحلیل‌گرها و توسعه‌دهندگان بتوانند از آن برای توصیف یک کار یا یک دارایی نرم‌افزاری استفاده کنند [۵]. MDA یک قدم بلند در راستای رسیدن به یک سازمان بلادرنگ است که در آن مدیران می‌توانند تغییرات مورد نیاز در معماری سازمان را انجام دهند و این تغییرات متعاقباً در کد لحاظ شوند [۹].



شکل۱: فرآیند MDA

۳-۱- انواع مدل‌ها در معماری مدل‌رانه

فرآیند MDA مفاهیم کلیدی یک سامانه را با جداسازی مدل‌های مربوط به آنها از یکدیگر متمایز می‌سازد. انواع مدل‌هایی که در فرآیند توسعهٔ MDA وجود دارند عبارتند از:

- ***مدل مستقل از محاسبه[۹] (CIM):*** یک مدل مستقل از محاسبه دیدی از سامانه بر اساس دیدگاه مستقل از محاسبه است. این دیدگاه بر روی محیط سامانه و نیازمندی‌های آن تاکید دارد. CIM جزییات ساختار سامانه‌ها را نشان نمی‌دهد.
- ***مدل مستقل از سکو[۱۰] (PIM):*** یک مدل مستقل از سکو دیدی از سامانه بر اساس دیدگاه مستقل از سکو است. این دیدگاه بر روی عملکرد یک سامانه تاکید می‌کند و در آن جزییات لازم برای پیاده‌سازی روی یک سکو خاص مخفی شده است. در واقع این دیدگاه بخشی از مشخصه کامل سامانه را نشان می‌دهد که از یک سکو به سکو دیگر بدون تغییر باقی می‌ماند. یک PIM درجه خاصی از استقلال سکو را ارائه می‌دهد که می‌تواند برای استفاده در تعدادی سکو از نوع مشابه استفاده شود.
- ***مدل خاص سکو[۱۱] (PSM):*** یک مدل خاص سکو دیدی از سامانه بر اساس دیدگاه خاص سکو است. این دیدگاه، دیدگاه مستقل از سکو را به همراه جزییات پیاده سازی بر روی یک سکو خاص نشان می‌دهد.

شکل ۱ نشان می‌دهد این مدل‌ها چگونه در یک فرآیند توسعه بر پایه MDA ایجاد می‌شوند.

۳-۲- استانداردهای معماری مدل‌رانه

OMG تعدادی از استانداردها و فرامدل‌های مختلف را به خدمت گرفته است تا با همکاری آنها معماری مدل‌رانه شکل گیرد. این استانداردها شامل MOF، UML، CWM، XMI، و نمایه‌های مختلف (مثل نمایه EDOC، EJB، ...) است. در این بخش نگاه کوتاهی به کاربرد هر یک از آنها می‌اندازیم. جزییات بیشتر را می‌توانید در [۱۱،۷] پیدا کنید.

- ***ابزار فراشی[۱۲] (MOF):*** ابزار فرا شی [۱۲] یک چارچوب یکپارچگی برای تعریف، دستکاری، و یکپارچگی فراداده‌ها و داده‌ها است. این کار با یک روش مستقل از سکو انجام می‌شود. تمام مدل‌ها و فرامدل‌های مورد استفاده در MDA بر اساس MOF تعریف می‌شوند.
- ***زبان مدل‌سازی یکپارچه[۱۳] (UML):*** UML یک زبان مدل‌سازی گرافیکی است که برای تصویرسازی، مستندسازی، و محدودسازی فرآورده‌های سامانه‌های شی‌گرا مورد استفاده قرار می‌گیرد. البته توجه داشته باشید که UML 1.x از تمام جنبه‌های مورد نظر MDA پشتیبانی نمی‌کند. از این رو نیاز به ایجاد یک

نگارش جدید از UML بود. UML 2.x نگارش جدید از استاندارد UML است که در MDA مورد استفاده قرار می‌گیرد [۸].

- ***زبان محدودیت شی[۱۴] (OCL):*** زبان محدودیت شی [۱۳] یک زبان کمکی برای UML است. این زبان به شما اجازه می‌دهد محدودیت‌ها و منطق فراروی استفاده از مدل‌ها را مشخص کنید. OCL یک زبان رسمی است که عبارات را در مدل‌های UML توصیف می‌کند.
- ***فرامدل انباره مشترک[۱۵] (CWM):*** این استاندارد یک فرامدل کامل و جامع است که داده‌کاوی را در پایگاه داده‌های یک سازمان امکان‌پذیر می‌سازد [۱۴]. می‌توان گفت جایگاه CWM در مدل‌سازی داده‌ها همانند جایگاه UML در مدل‌سازی برنامه‌ها است.
- ***فرامدل خدمات وب[۱۶] (WSM):*** این فرامدل جهت تسهیل توسعه مدل‌های سرویس‌های وب مبتنی بر MOF ایجاد شده است [۱۰].
- ***فرامدل تعریف فرآیند کسب‌وکار[۱۷] (BPDM):*** این فرامدل برای فرآیندهای کاری توسعه داده شده است [۱۵]. این فرامدل مستقل از زبان‌های تعریف فرآیند خاص است.
- ***مدل انگیزه کسب‌وکار[۱۸] (BMM):*** مدل انگیزه کسب‌وکار [۱۶] ساختاری را برای توسعه، ارتباط، و مدیریت برنامه‌های کسب‌وکار به روش سازمانی فراهم می‌کند.
- ***نمایش قواعد ساخت[۱۹] (PRR):*** نمایش قواعد ساخت [۱۷] برای برآوردن نیازهای نمایش قواعد ساخت در مدل‌های UML ایجاد شده است.
- ***معنی‌شناسی واژگان و قواعدکسب‌وکار[۲۰] (SBVR):*** این فرامدل برای استخراج قواعد کسب‌وکار به شکل عبارات کسب‌وکار مورد استفاده قرار می‌گیرد [۱۸]. همچنین می‌توان آن را برای تعریف عبارات کاری در فرهنگ واژگان کسب‌وکار و تعیین معنی آنها به کار برد.
- ***نمایه‌های UML:*** به کمک نمایه‌ها می‌توان زبان UML را برای یک ناحیه خاص از محاسبات (مثلاً محاسبات توزیع‌شده)، و یا یک سکو خاص (مثل EJB) سفارشی کرد. در UML 2.x طیف گسترده و کاملی از نمایه‌های مختلف تعریف شده است تا کاربری بالقوه MDA افزایش یابد. از بین این نمایه‌ها می‌توان به نمایه‌های CORBA، EJB، و Net. اشاره کرد.

۴- نقش MDA در یکپارچه‌سازی سامانه‌های ULS

در این بخش به بررسی مسئله اصلی این تحقیق می‌پردازیم: بررسی چالش‌های یکپارچه‌سازی سامانه‌های فوق وسیع و ارائه راهکارهای مبتنی بر MDA برای آنها. همان‌طور که پیش از این گفتیم، چالش‌های مختلفی فراروی توسعه سامانه‌های ULS وجود دارد. ابتدا چالش‌های یکپارچه‌سازی سامانه‌های فوق وسیع را ذکر کرده و در ادامه نشان می‌دهیم MDA چگونه می‌تواند در حل این چالش‌ها نقش داشته باشد.

چالش ۱: صنعت نرم‌افزار (و به ویژه توسعه سامانه‌های مقیاس پذیر) ویژگی‌هایی خاصی دارد که آن را از سایر صنایع جدا می‌کند. هر سال، (حتی گاه سریعتر) فناوری‌های جدید ابداع می‌شوند و به سرعت رایج می‌شوند (برای مثال جاوا، لینوکس، XML، HTML، SOAP، UML، J2EE، Net.، JSP، ASP، سرویس‌های وب، و ...). در محیط‌های مقیاس وسیع بسیاری از شرکت‌ها باید خود را با این تکنولوژی‌های جدید هماهنگ کنند، و با آنها حرکت کنند. در بحث یکپارچه‌سازی این وضعیت وخیم‌تر می‌شود. زیرا حتی اگر شما تنها بر روی نوع خاصی از تکنولوژی تمرکز کنید، برای تعامل با سایر نرم‌افزارها و یکپارچه‌سازی آنها نیازمند هماهنگی هستید. لازم به ذکر است که حتی یک تکنولوژی خاص نیز به سرعت تغییر می‌کند و نسخه‌های متعددی ایجاد می‌شوند که هیچ تضمینی برای سازگاری با نسخه‌های قدیمی وجود ندارد و توسعه‌دهندگان معمولا تنها دو یا سه نسخه قبلی را پشتیبانی می‌کنند. وجود مقیاس بسیار بالا در سامانه‌های ULS استفاده از طیف وسیعی از فناوری‌ها را اجتناب‌ناپذیر می‌سازد. چگونه می‌توان این فناوری‌ها را یکپارچه کرد؟

راهکار ۱: معماری مدل رانه با جدا سازی مدل‌های مختلف در سطوح تجرید گوناگون می‌تواند در یکپارچه‌سازی سامانه‌های ULS بسیار مفید باشد. زمانی که یک تکنولوژی تغییر می‌کند مدل‌های CIM و PIM بدون تغییر باقی می‌مانند و تنها نیاز است مدل‌های PSM تغییر کنند. انجام این کار حتی می‌تواند با ابزارهای خاص به طور خودکار انجام شود. البته هنوز این کار به شکل ۱۰۰٪ عملی نیست و به صورت نیمه خودکار انجام می‌شود. با این حال، قطعاً تغییر تنها مدل‌های PSM بسیار ساده‌تر و کم هزینه‌تر از تغییر کل نرم‌افزار است.

چالش ۲: فناوری‌های نوینی وجود دارند که هنوز به شکل کامل توسعه نیافته‌اند، اما در آینده نزدیک توسعه داده می‌شوند. سامانه‌های ULS چگونه می‌توانند با این فناوری‌ها یکپارچه شود؟

راهکار ۲: در اینجا ضرورت وجود یک زبان مشترک برای درک مدل‌های توسعه یک نرم‌افزار برای یکپارچه‌سازی آن کاملاً احساس می‌شود. MDA این مسئله را با ارائه استاندارد ابزار فرا شیء (MOF) حل کرده است. اگر تکنولوژی‌های جدید بر اساس استاندارد MOF توسعه داده شوند، سایر نرم‌افزارهای پیرو MOF[۲۱] به راحتی می‌توانند با آنها تعامل برقرار نموده و به این شکل متحد کردن آنها بسیار ساده‌تر از زمانی است که هیچ فرامدل مشترکی بین آنها وجود نداشته باشد. همان طور که در بخش ۳-۲ اشاره کردیم، MOF هسته استانداردهای

MDA است و تمامی مدل‌ها، فرامدل‌ها، و استانداردهای MDA پیرو MOF هستند.

چالش ۳: تولید و به روز رسانی مستندات در بهبود نگهداری و یکپارچه‌سازی نرم‌افزارها مفید است. در توسعه سامانه‌های ULS با توجه به مقیاس بسیار بالایی که دارند، آیا مستندات تولید می‌شود؟ چگونه می‌توان این مستندات را تولید و به روز رسانی کرد؟

راهکار ۳: مستندسازی همیشه یکی از نقاط ضعیف در فرآیند توسعه نرم‌افزار است و همواره به آن به عنوان یک کار جنبی نگاه می‌شود. بسیاری از توسعه‌دهندگان احساس می‌کنند که کار اصلی آنها تولید کد است. نوشتن مستندات در طی فرآیند توسعه زمان می‌برد و منجر به کند شدن فرآیند می‌گردد. مستندات وظیفه اصلی توسعه دهنده را پشتیبانی نمی‌کند و حضور آن، تنها وظیفه آنهایی را که بعدها می‌آیند، پشتیبانی می‌کند. بنابراین نوشتن مستندات طوری احساس می‌شود که گویی کاری است که تنها برای نمایش ایجاد می‌شود نه برای کار اصلی. غیر از مدیر پروژه که نوشتن مستندات را اجباری می‌کند، هیچ انگیزه‌ای برای نوشتن مستندات وجود ندارد. به همین دلیل است که مستندات با کیفیت کافی ایجاد نمی‌شوند و همین نکته باعث مشکلاتی در هنگام نگهداری و یکپارچه‌سازی نرم‌افزار می‌شود. این مسئله در سامانه‌های ULS بسیار وخیم‌تر است. MDA می‌تواند این وضعیت را بهبود بخشد. در معماری مدل‌رانه تولید مدل‌ها و مستندات جزء وظیفه اصلی محسوب می‌شود و تولید کد تا حد امکان به طور خودکار انجام می‌شود. ابزارهای MDA این توانایی را دارند که مدل‌های مختلف را به همراه مستندات آنها به خوبی نگهداری و یکپارچه سازند که نتیجه آن وجود اطلاعات کافی در زمان نگهداری، توسعه، و یکپارچه‌سازی نرم‌افزارها است.

چالش ۴: یکپارچه‌سازی نیاز به یک تغییر جهت عمده در سیاست‌های همکاری دارد. در سامانه‌های ULS به دلیل مقیاس بالا این ضرورت بیشتر احساس می‌شود. چگونه می‌توان سیاست‌های همکاری را تسهیل کرد؟

راهکار ۴: MDA برای همکاری و هماهنگی بین مدل‌ها بر روی یک استاندارد و روش واحد تاکید دارد. MOF به عنوان هسته تعامل و همکاری می‌تواند یک زبان مشترک را برای همکاری فراهم سازد. همچنین ابزارها و روش‌های خودکار نگاشت مدل‌ها می‌تواند بسیاری از همکاری‌های مورد نیاز را تسهیل کند.

چالش ۵: تلاش‌های یکپارچه‌سازی به دلیل وسعت حیطهٔ عمل آنها (خصوصاً در سامانه‌های ULS) تاثیرات وسیعی بر روی کسب‌وکار می‌گذارند. هنگامی که مهمترین وظایف یک کسب‌وکار به شکل یک راه‌حل یکپارچه در آمدند، درست عمل کردن این راه‌حل برای کسب‌وکار حیاتی است. چگونه می‌توان قبل از پیاده‌سازی و یکپارچه‌سازی از صحت عملکرد آنها مطمئن شد؟

راهکار ۵: در معماری مدل رانه، مدل‌ها ابزار هدایت و راهبری جریان توسعه نرم‌افزار هستند. این امکان وجود دارد که پس از تحلیل و طراحی مدل‌ها آنها را اجرا کرد تا از عملکرد آنها مطمئن شد. تکنیک‌های مربوط به Executable UML که در MDA مورد توجه قرار گرفته است این امکان را فراهم می‌کند که مدل‌ها مستقیماً اجرا و ارزیابی شوند. به این شکل امکان تست تحلیل و طراحی به شکل موثری امکان‌پذیر است و می‌توان قبل از پیاده سازی و یکپارچه‌سازی از صحت آنها مطمئن شد.

چالش ۶: علی‌رغم نیاز گسترده به راه‌حل‌های یکپارچه، تنها استانداردهای اندکی در این حوزه ایجاد شده است. این استانداردها نیز با هم هماهنگ نیستند و باعث آشفتگی در گسترش و تفسیر جدید استانداردها می‌شوند. چگونه می‌توان قابلیت همکاری و یکپارچه‌سازی را بین محصولات «پیرو استاندارد[۲۲]» (محصولاتی که از استانداردهای خاصی تبعیت می‌کنند) مختلف افزایش داد؟

راهکار ۶: معماری مدل رانه تاکید می‌کند که همهٔ استانداردها باید پیرو MOF باشند. وجود چهار سطح مختلف در استاندارد MOF (سطوح M0 تا M3) باعث می‌شود تعریف استانداردها ساده شود. از آنجا که MOF دارای سطح تعریف فرافرامدل[۲۳] است می‌توان استانداردهای جدیدی را تعریف کرد که خود آنها از یک فرااستاندارد بالادستی پیروی کنند.

چالش ۷: سامانه‌های ULS یک سامانه فنی- اجتماعی است. یکپارچه‌سازی آنها نیازمند ترکیب مجموعه‌ای از مهارت است که برخی از آنها در حوزه مهندسی نرم‌افزار و فناوری اطلاعات نیستند.

راهکار ۷: معماری مدل رانه راهکاری را برای این مشکل ارائه نمی‌دهد، اما بالقوه قابلیت حل این چالش را دارد. اگر بتوان استانداردها و مدل‌هایی را برای سایر تخصص‌های غیر IT که مبتنی بر MOF باشند ایجاد کرد، می‌توان از آنها در یکپارچه‌سازی سامانه‌های ULS استفاده کرد.

۵- نتیجه‌گیری

در این مقاله به بررسی ویژگی‌ها و چالش‌های فراروی توسعه سامانه‌های با مقیاس فوق وسیع پرداختیم. وجود مقیاس بسیار بزرگ این سامانه‌ها باعث تفاوت‌های جدی بین سامانه‌های ULS و سامانه‌های امروزی شده است، به طوری که رویکردهای (فعلی و مورد استفاده) مهندسی نرم‌افزار قادر به پاسخگویی به نیازهای آنها نیستند. برای حل این چالش‌ها باید رویکرد نوین و متفاوتی را در پیش گرفت. معماری مدل‌رانه یکی از جدیدترین استانداردهایی است که قابلیت‌های نوینی را در اختیار

صنعت نرم‌افزار قرار می‌دهد. سعی کردیم به کمک معماری مدل‌رانه راهکارهای اولیه‌ای برای چالش‌های یکپارچه‌سازی سامانه‌های ULS ارائه کنیم.

لازم به ذکر است معماری مدل‌رانه قطعا نمی‌تواند یک پاسخ کامل به چالش‌های توسعه سامانه‌های ULS باشد. ما هنوز در ابتدای راه توسعه سامانه‌های فوق وسیع هستیم. در ادامه این کار تحقیقاتی باید به سایر حوزه‌های چالشی نیز توجه کرد و سعی نمود برای آنها نیز راهکارهایی را ارائه کرد. همچنین می‌توان استانداردهای پیرو MOF برای سایر حوزه‌های غیرفناوری نیز تدوین کرد تا یکپارچه‌سازی سامانه‌های ULS به درستی فراهم شود.

۶- مراجع

[1] SEI, *Ultra-Large-Scale Systems: The Software Challenge of the Future*, Software Engineering Institute (SEI), Carnegie Mellon University, June 2006.

[2] Ostadzadeh, S.S., Shams, F., Ostadzadeh, S.A., "*An MDA-Based Generic Framework to Address Various Aspects of Enterprise Architecture*," Advances in Computer and Information Sciences and Engineering, Springer, pp. 455-460, August 2008. (ISBN 978-1-4020-8740-0)

[3] Ostadzadeh, S.S., Shams, F., Ostadzadeh, S.A., "*A Method for Consistent Modeling of Zachman Framework*," Advances and Innovations in Systems, Computing Sciences and Software Engineering, Springer, pp. 375-380, August 2007. (ISBN 978-1-4020-6263-6)

[4] Ostadzadeh, S.S., Shams, F., Ostadzadeh, S.A., "*Employing MDA in Enterprise Architecture*," Proceedings of the 12th International CSI Computer Conference (CSICC'07), pp. 1646-1653, February 2007.

[5] Frankel, D.S., *Model Driven Architecture: Applying MDA to Enterprise Computing*, OMG Press, Wiley Publishing, 2003.

[6] D'Souza, D., *Model-Driven Architecture and Integration*, Kinetium, March 2002.

[7] Mellor, S.J., Scott, K., Uhl, A., Weise, D., *MDA Distilled: Principles of Model-Driven Architecture*, Addison Wesley, 2004.

[8] Pilone, D., Pitman, N., *UML 2.0 in a Nutshell*, O'Reilly, 2005.

[9] Brown, A., *An Introduction to Model Driven Architecture*, IBM, 2004.

[10] Barry, Douglas K., *Web Services and Service-Oriented Architectures: The Savvy Manager's Guide*, Morgan Kaufmann Publishers, 2003.

[11] Object Management Group, *MDA Guide*, Version 1.0.1, OMG Document omg/2003-06-01, June 2003.

[12] Object Management Group, *Meta Object Facility (MOF) 2.0 Core Specification*, Version 2.0, OMG Document ptc/03-10-04, October 2003.

[13] Object Management Group, *Object Constraint Language (OCL) Specification*, Version 2.0, OMG Document formal/06-05-01, May 2006.

[14] Object Management Group, *Common Warehouse Metamodel (CWM) Specification*, OMG Document formal/03-03-02, March 2003.

[15] Object Management Group, *Business Process Definition Metamodel (BPDM)*, OMG Document bei/04-08-03, August 2004.

[16] Object Management Group, *Business Motivation Model (BMM) Specification*, OMG Document dtc/06-08-03, August 2006.

[17] Object Management Group, *Production Rule Representation (PRR) RFP*, OMG Document br/03-09-03, September 2003.

[18] Object Management Group, *Semantics of Business Vocabulary and Business Rules (SBVR) Specification*, OMG Document dtc/05-11-01, November 2005.

[19] Tabet, S., et al., "*OMG Production Rule Representation - Context and Current Status*," W3C Workshop on Rule Languages for Interoperability, 2005.

زیرنویس‌ها

[1] Ultra-Large-Scale Systems
[2] Assistant Secretary of the U. S. Army (Acquisition, Logistics, & Technology)
[3] Software Engineering Institute
[4] Object Management Group
[5] Wicked Problems
[6] Socio-technical
[7] Orchestration
[8] Model-Driven Architecture
[9] Computation-Independent Model
[10] Platform-Independent Model
[11] Platform-Specific Model
[12] Meta-Object Facility
[13] Unified Modeling Language
[14] Object Constraint Language
[15] Common Warehouse Metamodel
[16] Web Service Metamodel
[17] Business Process Modeling Language
[18] Business Motivation Model
[19] Production Rule Representation
[20] Semantics of Business Vocabulary and Business Rules
[21] MOF-Compliant
[22] Standard-Compliant
[23] Meta Meta Model