ارائه روشی برای ارزیابی نیازهای غیروظیفه‌مندی در فرایند توسعه‌نرم‌افزار

(یادداشت پژوهشی)

سیما عمادی[۱]* **فریدون شمس**[۲]

(۱) دانشجوی دکتری، دانشکده مهندسی، دانشگاه آزاد اسلامی واحد علوم و تحقیقات

(۲) دانشجوی دکتری، دانشکده مهندسی برق و کامپیوتر، دانشگاه شهید بهشتی

تاریخ ثبت اولیه: ۸۷/۱/۱۹، تاریخ دریافت نسخه اصلاح شده: ۸۷/۸/۸، تاریخ پذیرش: ۸۷/۸/۱۹

چکیده با رشد روز افزون سیستم‌های نرم‌افزاری و افزایش کاربرد آنها، ساده بودن فرآیند توسعه و ارزیابی نیازهای غیروظیفه‌مندی این سیستم‌ها، در سطوح اولیه توسعه، بخصوص سطح معماری نرم‌افزار مهم است؛ چرا که اولین محصول فرآیند توسعه نرم‌افزار، در سطح معماری نرم‌افزار حاصل می‌شود. ارزیابی معماری با کمک روش‌های مختلفی مثل مدل‌های اجرایی انجام می‌شود. از ای‌ن رو، در این مقاله حاضر، چارچوبی برای اجتماع ارزیابی نیازهای غیروظیفه‌مندی در فرآیند توسعه نرم‌افزار و با استفاده از مدل‌های اجرایی ارائه می‌گردد. چارچوب پیشنهادی، مبتنی بر معماری مدل‌رانه است که یک مدل مستقل از بستر را به یک یا چند مدل خاص بستر، تبدیل می‌نماید. در این چارچوب، معماری با علامت‌گذاری‌های مختلفی توصیف می‌شود و طی مراحلی به یک یا چند مدل خاص بستر (مدل اجرایی)، تبدیل می‌گردد. مهمترین هدف این چارچوب، کاهش تعداد تبدیلات توصیفات نرم‌افزار به مدل‌های اجرایی است. برای رسیدن به این هدف، یک ساختار عمومی برای مدل‌های اجرایی مبتنی بر بسط‌های مختلف شبکه‌های پتری معرفی می‌شود.

واژه‌های کلیدی ارزیابی معماری، تحلیل نیازمندی‌ها، معماری مدل‌رانه، مدل اجرایی، نیازمندی‌های غیروظیفه‌مندی، چارچوب، شبکه پتری.

*عهده دار مکاتبات

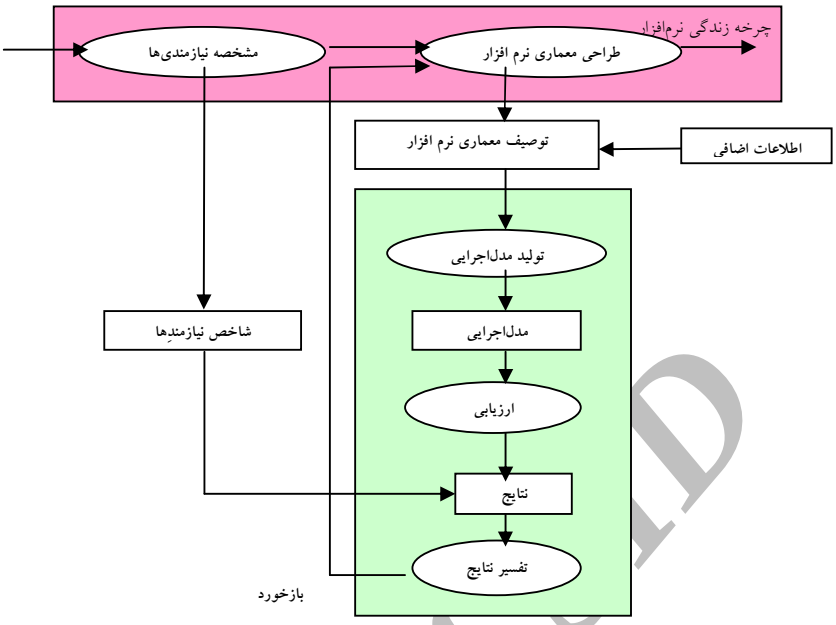
نشانی: استان یزد – دانشگاه آزاد اسلامی واحد میبد – گروه کامپیوتر

تلفن: ۰۹۱۵۵۳۲۲۵۶ **پست الکترونیکی:** emadi@sr.ian.ac.ir

۱- مقدمه

با افزایش پیچیدگی، اندازه و میزان سرمایه‌گذاری برای توسعه سیستم‌های نرم‌افزاری، توسعه آنها برمبنای اصولی که ضمن کاهش هزینه‌های سرمایه‌گذاری، کلیه نیازهای مورد نیاز توسعه‌دهندگان و کاربران این سیستم‌ها را برآورده کند، ضروری است [۱]. ارزیابی این نیازها در مراحل اولیه توسعه نرم‌افزار، بخصوص مرحله توسعه معماری نرم‌افزار، باعث کاهش هزینه‌های مصرفی، و تسهیل در اعمال تغییرات می‌گردد [۲ و ۳]. معماری نرم‌افزار، اولین محصول فرآیند توسعه یک سیستم نرم‌افزاری است که ساختار پویا و ایستای سیستم را نشان‌می‌دهد و برای چگونگی دستیابی به تمام نیازهای موجود، اعم از نیازهای وظیفه‌مندی و غیروظیفه‌مندی تصمیم می‌گیرد. این تصمیمات، روی کیفیت نهایی سیستم‌های نرم‌افزاری تأثیرگذارند؛ به همین دلیل بهترین مرحله برای ارزیابی نیازهای وظیفه‌مندی و غیروظیفه‌مندی، مرحله توسعه معماری نرم‌افزار است [۱و۲]. اما توسعه‌دهندگان نرم‌افزار با ارزیابی و تحلیل نیازهای غیروظیفه‌مندی در طی فرآیند توسعه آشنا نیستند و بین طراحی نرم‌افزار و تحلیل نیازهای غیروظیفه‌مندی، فاصله‌زیادی هست. با ایجاد یک مدل قابل اجرا از معماری، این نیازها در سطح معماری نرم‌افزار ارزیابی و فاصله موجود بین طراحان و تحلیل‌گران کم می‌شود. مدل‌های اجرایی در فازهای آغازین، رفتار پویای سیستم نهایی را در شرایط مختلف و قبل از پیاده‌سازی معماری، ارزیابی می‌کنند و نیازهای بلادرنگ و واقعی کاربران نیز به راحتی حاصل می‌شود [۴ و۵]. شکل (۱)، اجتماع مدل‌های اجرایی را در سطح معماری نرم‌افزار به صورت فعالیت‌ها و خروجی هر فرآیند نشان می‌دهد. در این شکل، به توصیفات معماری نرم‌افزار، اطلاعات مورد نیاز برای ارزیابی صفات غیروظیفه‌مند، اضافه می‌شود و سپس به یک مدل قابل اجرا، تبدیل می‌شود. این مدل اجرایی با استفاده از شاخص نیازهای غیروظیفه‌مندی، ارزیابی و نتایج حاصل از این ارزیابی، تجزیه و تحلیل می‌گردد. در نهایت، نتایج حاصل به معماری نرم‌افزار، باز خورد می‌شود.

برای اجتماع ارزیابی نیازهای غیروظیفه‌مندی در سطح معماری نرم‌افزار، روش‌ها و ابزارهای متعددی وجود دارد. این روش‌ها، معماری نرم‌افزار را به عنوان ورودی دریافت و به آن اطلاعات لازم برای ارزیابی نیازهای غیروظیفه‌مندی را اضافه می‌کنند تا از آن یک مدل قابل اجرا تولید نمایند. در نهایت، مدل قابل اجرا به کمک یک ابزار پشتیبان، تجزیه و تحلیل می‌گردد. این روش‌ها، معماری نرم‌افزار را با زبان‌های توصیف معماری، زبان مدلسازی یکپارچه (Unified Modeling Language) [۶]، نمودار ترتیب پیغام (Message Sequence Chart (MSC) و غیره توصیف می‌نمایند و خروجی آنها مدل‌های اجرایی مختلفی مثل شبکه‌های صف، شبکه‌های پتری، جبر فرایندی، فرایندهای تصادفی و غیره است. لذا اگر بتوان N علامتگذاری برای توصیف معماری را به M مدل قابل اجرا تبدیل نمود، آنگاه به N*M تبدیل نیاز است. در این مقاله برای کاهش تعداد این تبدیلات به N+M، چارچوبی مبتنی بر معماری مدل ارائه(Model Driven Architecture (MDA)) ارائه گردیده است تا علاوه بر اجتماع ارزیابی نیازهای غیروظیفه‌مندی در سطح معماری، هر نوع علامتگذاری حاصل از توصیف معماری را دریافت و طی مراحلی، مدل قابل اجرای آن را تولید نماید. در این چارچوب، طراح، اطلاعات لازم برای ارزیابی نیازهای غیروظیفه‌مندی را به علامت‌گذاری مورد استفاده برای توصیفات معماری اضافه می‌نماید. در چارچوب پیشنهادی نیز معماری نرم‌افزار، مستقل‌از هر نوع مدل اجرایی و با هر نوع علامت‌گذاری، توصیف و اطلاعات لازم برای ارزیابی نیازهای غیروظیفه‌مندی به آن اضافه می‌شود. درمراحل بعدی این توصیفات به یک مدل قابل اجرا، تبدیل و مورد ارزیابی قرارمی‌گیرد.



شکل ۱ اجتماع مدل اجرایی معماری نرم‌افزار در فرآیند توسعه

چارچوب پیشنهادی، محدودیتی در ارزیابی انواع صفات غیروظیفه‌مند و استفاده از انواع مدل‌های اجرایی و متدولوژی‌ها یا ابزار ارزیابی ندارد. به همین منظور، در بخش میانی این چارچوب، گرامری مشترک برای مدل‌های اجرایی معرفی می‌گردد که توصیفات معماری، طی مراحلی به این گرامر مشترک تبدیل‌می‌شوند. کارهای مختلفی در زمینه اجتماع تحلیل نیازهای غیروظیفه‌مند در فرایند توسعه نرم‌افزار انجام شده است. ضرورت این اجتماع در سطوح اولیه توسعه در [۷]، بیان شده است. چارچوب پیشنهادی در [۸]، نیازهای غیروظیفه‌مندی را با مدل‌های مفهومی(conceptual models) ترکیب و توسعه‌دهنده را به ساخت مدل‌های تک کاره محدود می‌کند.

استخراج نیازها در این چارچوب مبتنی‌بر مدل روابط بین موجودیت‌ها(Entity Relationship Models) است، در حالی که در روش پیشنهادی این مقاله، اطلاعات نیازهای غیروظیفه‌مندی به علامت‌گذاری‌های نرم‌افزار اضافه می‌گردد و به یک مدل مبتنی بر XML تبدیل می‌شود. در [۱۹و۲۰]، چارچوبی ارائه شده که کارایی و قابلیت اطمینان سیستم‌های مبتنی بر مؤلفه را در سطح معماری ارزیابی می‌نماید. ورودی این چارچوب، معماری توصیف شده با نمودارهای زبان مدلسازی یکپارچه و خروجی آن شبکه صف است. در این چارچوب که مبتنی بر توسعه مدل‌رانه (Model Driven Development(MDD)) است مشابه چارچوب پیشنهادی در این مقاله یک زبان میانی به نام D_KLAPER ما بین توصیفات معماری و مدل اجرایی قرار می‌گیرد. این زبان تنها به تعریف تبدیلات بین علامت‌گذاری‌های استفاده شده برای توصیف معماری و علامت‌گذاری‌های مدل اجرایی مورد نظر کمک می‌کند. اما زبان میانی در چارچوب پیشنهادی، مبتنی بر XML است. دلیل استفاده از XML در چارچوب پیشنهادی این است که XML زبانی است که به راحتی می‌توان با کمک آن داده‌ها را توصیف و دستکاری نمود، از طرفی ساختار سلسله مراتبی موجود در XML برای نمایش مؤلفه‌های معماری، اشیاء و ساختارهای مشابه بسیار مناسب است؛ زیرا به راحتی می‌توان با استفاده از آن، ساختار سلسله مراتبی، مؤلفه‌ها، ویژگی‌های قابل مشاهده آنها و ارتباط بین آنها را نشان داد. ضمن اینکه حاشیه نویسی‌های اضافه شده به نمودارهای توصیف

کننده معماری، در این ساختار نشان داده می‌شوند. از اهداف دیگر چارچوب پیشنهادی این است که در بخش میانی، یک گرامرعمومی برای مدل‌های ارزیابی مبتنی بر شبکه‌های پتری ارائه شود که این مدل نیز مبتنی بر XML است. دلیل دیگر استفاده از این زبان میانی در چارچوب پیشنهادی این است که تنها اطلاعات مورد نیاز برای ارزیابی کارایی و قابلیت اطمینان از توصیفات معماری استخراج شده و به صورت کدهای XML بیان می‌شود. سپس از این کدها مدل اجرایی مورد نظر استخراج می‌گردد.

چارچوب پیشنهادی در [۹]، ارزیابی نیازهای وظیفه‌مندی و غیروظیفه‌مندی را در سطح معماری نرم‌افزار ترکیب می‌کند، اما در چارچوب پیشنهادی این مقاله [۱۰]، دو صفت کارایی و قابلیت اطمینان در سطح معماری نرم‌افزار ارزیابی می‌گردد و مدل نهایی آن، بسط‌های مختلف شبکه پتری است. بخش میانی چارچوب، قوانین موجود برای تبدیل توصیفات معماری به مدل نهایی را به صورت یک گرامرعمومی بیان می‌کند. گرامر عمومی ارائه شده در این مقاله [۱۰]، مشابه PNML [۱۱] است. هدف PNML پشتیبانی از انواع شبکه‌های پتری بین ابزارهای مختلف است. اما تفاوت آن با روش پیشنهادی دراین مقاله، این است که روش پیشنهادی بر روی استخراج اطلاعات مربوط به کارایی و قابلیت اطمینان از مدل منبع و نگاشت آن به گرامر عمومی تأکید دارد. چارچوب پیشنهادی، علاوه برداشتن قابلیت PNML، پلی بین توصیفات معماری و بسط‌های مختلف شبکه‌های پتری نیز هست. یکی از اولین کارهایی که در رابطه با ایجاد ساختار مشترک بین بسط‌های شبکه‌های پتری صورت گرفته در [۱۲] مطرح شده است. دراین کار، یک گرامر عمومی برای شبکه‌های پتری معرفی شده است. تفاوت این کار با گرامر عمومی ارائه شده این مقاله این است که در این مقاله، ویژگی‌های جدیدی به بعضی از بسط‌های شبکه‌های پتری داده شده و گرامر بیان شده برای آنها کامل‌تر بیان شده است.

ساختار مقاله در ادامه به این شرح است: در بخش ۲، چارچوبی برای اجتماع ارزیابی صفات غیروظیفه‌مند در سطح معماری نرم‌افزار و در بخش ۳، بخش میانی چارچوب ارائه می‌شود. در بخش ۴، گرامر انواع شبکه‌های پتری و در بخش ۵، یک گرامر مشترک برای بسط‌های مختلف شبکه‌های پتری ارائه می‌گردد. بخش ۶، کاربرد چارچوب پیشنهادی را با ارائه یک مثال نشان می‌دهد و نهایتاً در بخش ۷، نتیجه‌گیری بیان می‌شود.

۲- ساختار چارچوب پیشنهادی

به‌دلیل اهمیت معماری نرم‌افزار، به عنوان اولین محصول فرآیند توسعه نرم‌افزار، در ارزیابی نیازهای وظیفه‌مندی و غیروظیفه‌مندی سیستم، در این مقاله، چارچوبی برای ادغام آنها ارائه می‌گردد. این چارچوب، مبتنی بر معماری مدل رانه است [۱۳] و در سطوح طراحی و معماری استفاده می‌شود. چارچوب پیشنهادی اهداف زیر را دنبال می‌کند:

الف- تجمع تحلیل کلیه نیازهای غیروظیفه‌مندی در سطوح اولیه توسعه نرم‌افزار

ب- خودکار نمودن این تحلیل به همراه بازخورد نتایج حاصل از تحلیل به مدل‌های نرم‌افزاری

ج- توصیف معماری با هر نوع علامت‌گذاری یا زبان توصیف معماری

د- ایجاد هر نوع مدل اجرایی از توصیفات فوق

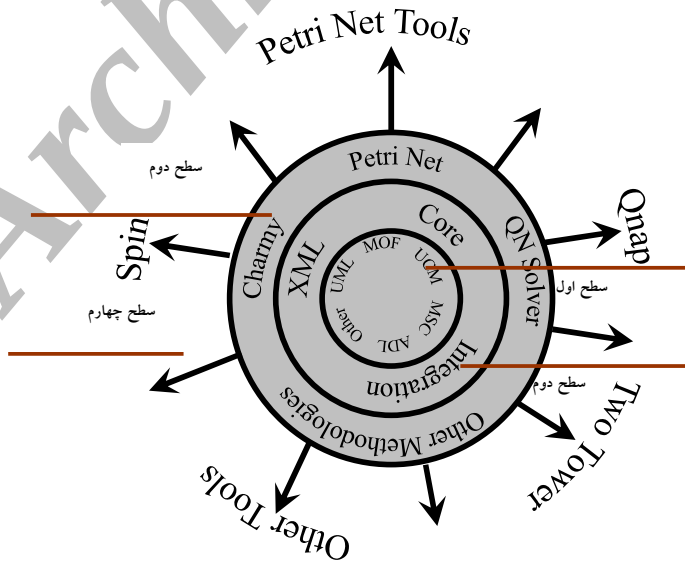
ه- ارائه ساختار مشترک بین توصیفات معماری و مدل‌های اجرایی

در این چارچوب، تبدیل توصیفات معماری (مدل منبع) به مدل مقصد در سه بخش صورت می‌گیرد. در بخش اول، توصیفات معماری به XML تبدیل می‌شود. در بخش دوم، این توصیفات به یک مدل میانی مبتنی بر XML تبدیل می‌شود و در بخش سوم، مدل میانی به یک مدل مقصد برای تحلیل تبدیل می‌گردد. برای این تبدیلات، مجموعه‌ای از قوانین در بخش‌میانی قرار می‌گیرد که عناصر مدل منبع را به

عناصر مدل مقصد نگاشت می‌دهد. این قوانین، روابط بین عناصر منبع و مقصد را نشان می‌دهند و اعمال بازخورد از مدل نهایی به مدل منبع را ساده می‌کنند. مدل میانی، یک مدل عمومی انعطاف‌پذیر است که می‌تواند هر نوع مدل منبع جدیدی را به یک مدل مقصد جدید دیگر تبدیل نماید. به همین علت در این مقاله، در بخش میانی چارچوب، یک گرامر عمومی برای شبکه‌های پتری ارائه شده است. این گرامر به گونه‌ای تعریف می‌شود تا هر نوع ابزار مبتنی بر شبکه‌های پتری از آن استفاده کند و توصیفات معماری که به این گرامر مشترک تبدیل می‌شوند به راحتی قابل تبدیل به هر نوع مدل مبتنی بر شبکه صف هستند.

در این چارچوب ابتدا یک مدل مستقل از بستر با یکی از علامت‌گذاری‌ها (مثل UML، MSC، UCM، ADL و ...) توصیف‌می‌شود و سپس به یک مدل میانی مبتنی‌بر XML و درنهایت به یک مدل خاص‌بستر، تبدیل‌می‌شود. به این مدل، پارامترهای لازم برای ارزیابی نیازها اضافه‌می‌گردد و سپس مدل ارزیابی و تحلیل می‌شود. بین توصیفات‌معماری و مدل‌اجرایی حاصل از آن (مثل شبکه‌صف، شبکه‌پتری و ...) یک مدل میانی عمومی که مبتنی بر XML است، قرارمی‌گیرد. همان طور که بیان شد، مدل میانی یک مدل عمومی و انعطاف‌پذیر است و قوانین و روابط میان علامت‌گذاری‌های مختلف توصیف معماری و مدل اجرایی را شامل می‌شود تا به کمک آنها نتایج به مدل نرم‌افزاری، به طور خودکار بازخورد شود. تفاوت کار ارائه شده در این مقاله با تحقیقات قبلی این‌است که در این مقاله باید نیازهای غیروظیفه‌مندی به مدل توصیف کننده معماری حاشیه‌نویسی شوند و توسعه‌دهنده به ساخت مدل‌های خاص محدود نمی‌شود.

شکل (۲)، چارچوب پیشنهادی در این مقاله را نشان می‌دهد. در سطح‌اول این شکل، معماری نرم‌افزار با هر نوع علامت‌گذاری توصیف می‌شود؛ ضمن اینکه پارامترهای لازم برای ارزیابی نیازهای غیروظیفه‌مندی به آنها حاشیه‌نویسی‌می‌شود. در سطح دوم، بخش میانی چارچوب که مبتنی بر XML است، قرار می‌گیرد. بین این دو سطح، فیلترهای ورودی توصیفات معماری را دریافت و به نمایش عمومی مبتنی بر XML موجود در سطح میانی تبدیل می‌کنند.



شکل ۲ ساختار چارچوب‌پیشنهادی

در سطح سوم، مدل‌های اجرایی مختلف برای ارزیابی معماری نرم‌افزار قرار می‌گیرد. بین سطوح دوم و سوم نیز فیلترهای خروجی، مدل مبتنی بر XML را دریافت و به یک مدل اجرایی مثل شبکه صف، شبکه پتری و غیره تبدیل می‌کنند. در نهایت خروجی سطح سوم نیز به ابزار مناسب در سطح چهارم داده می‌شود تا ارزیابی گردد. در این شکل، مدل مستقل از بستر سطح دوم می‌تواند به چند مدل خاص بستر تبدیل گردد. نتایج تحلیل نیز به نمایش مبتنی بر XML و سپس به مدل نرم‌افزاری مربوطه، بازخورد می‌گردد.

۳- ساختار میانی چارچوب پیشنهادی

همان طور که در بخش قبل مطرح شد، هدف بخش میانی استخراج یک مدل تحلیل از توصیفات معماری نرم‌افزار است. به همین منظور، در این بخش، یک گرامر عمومی برای کلیه بسط‌های شبکه‌های پتری بیان می‌گردد تا توسط فیلترهای خروجی به هر نوع شبکه‌پتری تبدیل گردد. قوانین موجود در این بخش، ارتباط بین موجودیت‌های توصیف معماری و مدل قابل اجرا را نشان می‌دهد تا تبدیلات صورت گیرد و نتایج حاصل از ارزیابی به معماری باز خورد شود. بخش میانی، اهداف زیر را دنبال می‌کند:

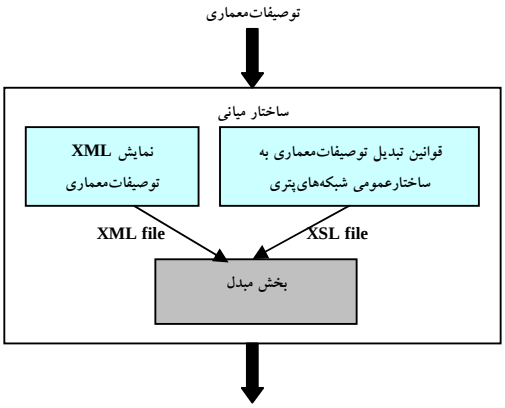
الف- نمایش‌آسان و قابل‌مدیریت همهٔ علامت گذاری‌های موجود در سطح اول

ب- تجمع تحلیل و بازخوردهای حاصل از آن

ج- ارائه گرامری مشترک برای همهٔ مدل‌های اجرایی

د- ارائه قوانینی به منظور نگاشت موجودیت‌های مدل‌های منبع به مدل‌های مقصد

شکل (۳)، ساختار بخش میانی را نشان می‌دهد. در این شکل، وظایف بخش میانی به دو قسمت تقسیم می‌شود. در بخش اول، اطلاعات مورد نیاز برای تحلیل نیازهای غیروظیفه‌مندی معماری نرم‌افزار از فیلتر ورودی و در قالب یک فایل XML، استخراج می‌شود. در بخش دوم، این اطلاعات به یک ساختار عمومی برای کلیه بسط‌های مدل‌های اجرایی تبدیل می‌شود. برای این تبدیل، روابط معنایی بین موجودیت‌های مدل منبع و مدل مقصد به صورت قوانینی مشخص می‌شود.



شکل ۳ ساختار میانی چارچوب

بخش میانی چارچوب به صورت افزایشی (incremental) پیاده‌سازی می‌شود؛ زیرا در این سطح برای هرجفت از علامت‌گذاری‌ها و مدل تحلیل، می‌توان به تدریج، قوانینی که بیانگر ارتباط بین عناصر آنها است، معرفی کرد.

همان طور که در بخش ۵، آورده خواهد شد، بخش مبدل یا فیلتر خروجی باید توصیفات معماری که به صورت کدهای XML بیان شده‌اند را به صورت یک گرامر عمومی بیان کند تا قابل تبدیل به هر یک از بسط‌های شبکه‌های پتری باشد و یا اینکه توسط هر یک از ابزارهای پشتیبان این بسط‌ها قابل استفاده باشد. از این رو، در بخش بعدی بعد از توصیف کلی شبکه‌های پتری، گرامر عمومی برای آنها ارائه می‌شود. پارامترهای حاشیه‌نویسی شده به توصیفات معماری نیز به صورت گذارهای زمانی، آنی، انواع متفاوت برای مهره‌ها و یا به صورت محافظ‌هایی برای کمان‌ها بیان می‌گردند.

٤- شبکه‌های‌پتری و بسط‌های مختلف آن

نظریه شبکه‌های پتری برای اولین بار توسط کارل پتری در اوایل ۱۹۶۰ ارائه گردید. شبکه‌های پتری یک نمایش گرافیکی واضح از سیستم‌ها به همراه توصیف ریاضی از

آنها ارائه می‌دهند و در کاربردهای مختلفی همچون کاربردهای موازی، غیرهمگام، توزیعی، غیرقطعی، تصادفی، تحمل‌پذیر خطا و غیره استفاده می‌شوند. همچنین چارچوبی برای تحلیل، اعتبارسنجی و ارزیابی نیازهای غیروظیفه‌مندی مثل کارایی و قابلیت اطمینان سیستم‌های پیچیده ارائه می‌دهند [۱۴].

در ادامه ابتدا مهمترین انواع شبکه‌های پتری بررسی می‌گردد و سپس گرامر آنها به صورت BNF (Backus Naur Form)، بیان می‌شود. گرامر بیان شده در این بخش مشابه گرامر ارائه شده در [۱۲] است با این تفاوت که این گرامرها کامل‌تر توصیف شده‌اند. همچنین به بعضی بسط‌های شبکه پتری قابلیت‌های جدیدی اضافه می‌شود و گرامر آنها نیز تعریف می‌گردد.

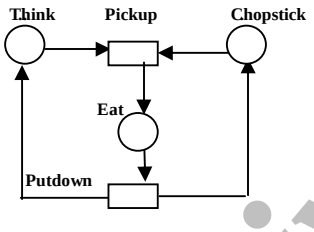
۴-۱ شبکه‌های مکان- گذار (Place/Transition Net).

شبکه مکان-گذار یک شبکه ۶ عضوی به صورت $PN=(P, T, I, O, M_0)$، است که $P=\{p_1,...,p_n\}$، مجموعه‌ای متناهی از مکان‌ها و $T = \{t_1,...,t_m\}$، مجموعه‌ای متناهی از گذارها است. همچنین $I: P\times T\rightarrow N_0$ مجموعه‌ای از کمان‌های ورودی، $O: P\times T\rightarrow N_0$ مجموعه‌ای از کمان‌های خروجی و $M_0: P\rightarrow N_0$ نیز علامت‌گذاری اولیه (initial marking) است [۱۵و ۱۶]. با این تعریف، گرامر BNF این شبکه‌ها به صورت‌زیراست:

```
NET          ::= start {ID} ELEMENT end
ELEMENT      ::= empty      |PLACE ELEMENT
                 |TRANSITION ELEMENT
                 |ARC ELEMENT
ID           ::= STRING
PLACE        ::= place {ID} {NAME INIT CAP}
NAME         ::= empty |name {STRING}
INIT         ::= empty |init {INTEGER}
CAP          ::= empty |capacity { INTEGER}
TRANSITION   ::= transition {ID} {NAME}
ARC          ::= arc{ID} {source{ID} target{ID} WEIGHT}
WEIGHT       ::= empty |weight {INTEGER}
```

به عنوان مثال شکل (۴)، شبکه پتری مسئله معروف شام فیلسوفان را نشان می‌دهد [۱۷]. در این شکل، سه مکان و دو گذار وجود دارد و توصیف گرامر آن براساس گرامر فوق به صورت زیر است:

```
place {think init{2}}      place {chopstick init{1}}    place {eat}
transition {Pickup}        transition {Putdown}
arc {a1}{source{think} target{pickup} weight{f1}  type{ordinary}}
arc {a2}{source{chopstick} target {pickup} weight {f2}}
arc {a3} {source{eat} target{putdown} weight{f4}}
arc  {a4} {source{pickup} target{eat} eight{f3}}
arc {a5}{source {putdown} target{think} weight {f5}}
arc {a6}{source{putdown} target{chopstick} weight {f6}}
```



شکل ۴ شبکه‌پتری مکان - گذار شام فیلسوفان [۱۶]

۴-۲ شبکه‌های پتری رنگی مبتنی‌برزمان (colored Timed Petri net).

Kurt Jenson، شبکهٔ پتری رنگی را به عنوان نسخهٔ توسعه یافته شبکه پتری معرفی کرد. دراین شبکه علاوه بر عناصر مکان، گذار و مهره از مفاهیم رنگ، محافظ(guard) و عبارات (expression) استفاده می‌شود [۱۵]. در توصیف رسمی، این شبکه‌ها به صورت $CPN=(\Sigma, P,T,I,O,H,C,G,E,M_0)$ تعریف می‌شوند. که Σ، مجموعه غیرتهی از رنگ‌ها، P, T, I, O, H و M_0 مشابه شبکه‌های مکان-گذار هستند. C یک تابع رنگ است که $P \rightarrow \Sigma$، تابع محافظ با G، مشخص می‌شود که یک عبارت منطقی را به گذارها نسبت می‌دهد. E، تابعی با نتیجه منطقی است که به کمان‌ها نسبت داده می‌شود [۱۵].

زمان یکی از مفاهیمی است که در ارزیابی

نیازمندی‌های کارایی و قابلیت اطمینان مهم‌است. در شبکه‌های پتری رنگی مبتنی بر زمان، هر مهره علاوه بر مقدار، زمان نیز دارد که به این مقدارزمانی، زمان مهر(timestamp) می‌گویند. همچنین به گذارها، تأخیرهای زمانی و احتمال شلیک شدن افزوده می‌شود. در این شبکه‌ها، گذار مورد نظر وقتی فعال‌است که مهره در مکان‌های ورودی آن گذار باشد و شرط مربوط به محافظ آن درست باشد. اما این گذار وقتی شلیک می‌شود که درحالت آماده‌باش قرار گیرد؛ یعنی باید زمان مهره‌های متعلق به مکان‌های ورودی گذار، کوچکتر و یا مساوی زمان فعلی گذار باشد.

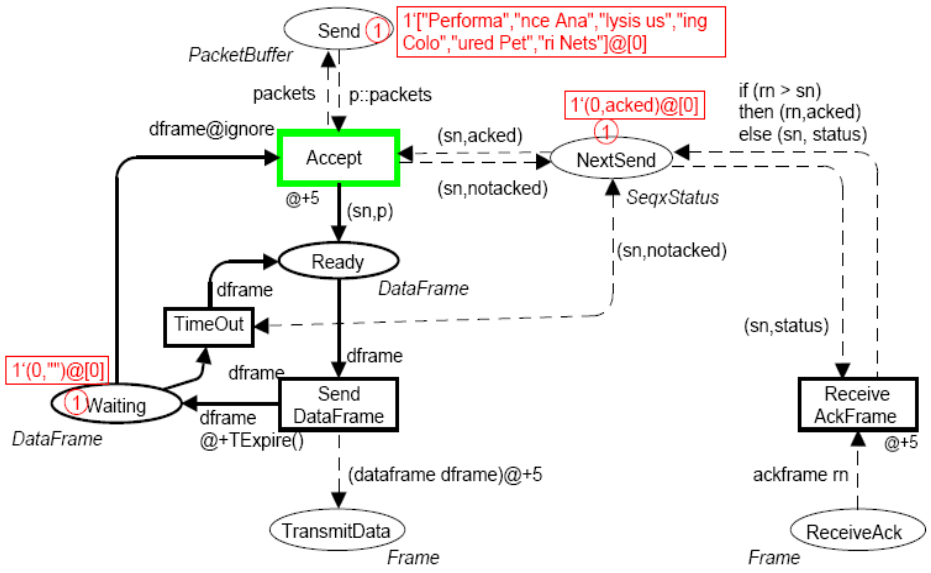
با توجه به تعریف فوق، گرامر BNF این شبکه به صورت زیر است:

```
ELEMENT ::= empty  |PLACE  ELEMENT
                   |TRANSITION
ELEMENT
                   |ARC
ELEMENT
                   |TYPEDEF
ELEMENT
                   |seeml  {FILENAME}
ELEMENT
FILENAME ::= STRING
PLACE   ::=  place {ID} { NAME INIT CAP
COLOUR}
INIT            ::=  empty  |init { MULTISET}
CAP             ::= empty  |capacity {MULTISET}
COLOUR      ::= empty |colour {COLOURSET}
COLOURSET ::= ML_DEFINITION
MULTISET  ::= INTEGER  |INTEGER' STRING
MSLIST   |INTEGER' STRING MSLIST
@+ T_WEIGHT
MSLIST          ::= empty  |+ MULTISET
TRANSITION ::= transition {ID} {NAME
GUARD T_WEIGHT}
GUARD               ::= empty  |guard
{BOOLEXPR}
BOOLEXPR    ::= ML_EXPRESSION
T_WEIGHT     ::= empty |weight{REAL}
ARC     ::= arc {ID} {source{ID} target{ID}
WEIGHT A_TYPE}
WEIGHT     ::= empty   |weight
{MULTISETEXPR}
A_TYPE      ::= empty |type(ordinary)  |
type(inhibitor)
NAME        ::= empty  |name {STRING}
MULTISETEXPR   ::= ML_EXPRESSION
```

```
ML_EXPRESSION ::= EXPRESSION
EXPRESSION      ::= { EXPRESSION } and
{EX PRESSION}  |{EXPRESSION} or
{EXPRESSION}        |not { EXPRESSION }
                |{EXPRESSION}=
{EXPRESSION }
                |{ EXPRESSION}<>
{EXPRESSION}
               |VALUE   |ML_FUNCTION|empty
VALUE         ::= value{true} |value {false}
TYPEDEF     ::= typedef {ID}
{ML_DEFINITION}
```

به عنوان مثال در یک شبکه ارتباطی، پروتکل توقف و انتظار(stop and wait)، ارتباطی مطمئن بین کامپیوترهای فرستنده و دریافت‌کننده برقرار می‌کند [۱۸]. شکل (۵)، شبکه پتری رنگی مبتنی بر زمان این پروتکل را نشان می‌دهد [۱۸]. گرامر توصیف‌کننده این شبکه به صورت زیر است:

```
Func1()=(if m>sn  then (m,acked)   else
(sn,status))
Color={packetbuffer, packet, dframe status seq
dataframe ackframe seqxstatus }
Place {ready, color{dataframe} }
Place {waiting init{1'(0,"")@+0} 1
color{dataframe}}
Place {transmitdata color{frame}}
Place{nextsend init {1' (0,acked) @ + 0} 1 color
{seqxstatus}}
Place {send  init{1'("performa","ance ana",
"lysing us",  "ing colo", "ured Pet", "ri
nets")@+0} 1 color{packetbuffer}}
Place {receiveack color{frame}}
transition {accept weight{5} }
transition {senddataframe weight{TExpire}}
transition {timeout}
transition {receiveackframe weight{5}}
arc{a1source{send} target{accept}
weight{P::packets} type{ordinary}}
arc {a2 source {accept} target {send}
weight{packets} type{ordinary}}
arc {a3 source {accept} target {ready}
weight{(sn,P)}  type{ordinary}}…..
arc {a14 source {nextsend} target {timeout}
weight{(sn,notacked)}    type{ordinary}}
```



شکل ۵ شبکه‌پتری پروتکل توقف و انتظار [۱۸]

۳-٤ شبکه‌های پتری تصادفی (Stochastic Petri Net)

شبکه‌های پتری، برای آنالیز کیفی ویژگی‌های منطقی سیستم‌ها، استفاده می‌شوند و برای استفاده در آنالیز کمی باید به گذارهای آنها، زمان با توزیع نمایی اضافه شود. به این شبکه‌ها، شبکه‌های پتری‌تصادفی (Stochastic Petri Net (SPN)) ، گفته می‌شود. در شبکه‌های پتری تصادفی، تأخیرآتش شدن گذارها را متغیرهای تصادفی با توزیع نمایی، مشخص می‌کنند و تأخیر شلیک شدن به هر گذار، به طور تصادفی، نسبت داده می‌شود که تابع چگالی احتمال آن یک نمایی منفی با نرخ λ_i است [۱۶].

توصیف رسمی شبکه‌های پتری‌تصادفی به صورت SPN= (P, T, I, O,H, W, M_0) است، که P، T، I، O، H و M_0 مشابه شبکه‌های پتری رنگی است و W، نیز تأخیرنسبت داده شده به هر گذار است و به صورت یک متغیر تصادفی با توزیع نمایی منفی بیان می‌شود. با تعریف فوق، گرامر BNF این شبکه به صورت زیر است که بخش‌های مشابه با شبکه مکان- گذار آن حذف شده است:

```
TRANSITION ::= transition {ID} {NAME
T_WEIGHT}
T_WEIGHT ::= weight{REAL}
|weight {EXPONENTIAL_DISTR}
| ML_FUNCTION
EXPONENTIAL _ DISTR ::=
ML_EXPRESSION
ARC              ::= arc {ID} {source{ID}
target{ID}
                     WEIGHT A-TYPE}
WEIGHT           ::= empty |weight
{INTEGER}

|weight{MULTISETEXPR}
```

٤-٤ شبکه‌های پتری رنگی تصادفی (Stochastic Colored Petri Net)

در چارچوب پیشنهادی برای ارزیابی کارایی و قابلیت اطمینان در سطح معماری نرم‌افزار به شبکه‌های پتری رنگی، قابلیت تصادفی نیز اضافه شده است. در واقع به گذارهای این شبکه‌ها، زمان با توزیع نمایی منفی اضافه می‌گردد. توصیف رسمی شبکه‌های پتری رنگی تصادفی به صورت

$GSPN=(\sum,P,T,I,O,H,C,G,E,W,M_0)$ است. که Σ ، P، M_0 ،E ،G ،C ،H ،O ،I ،T همان توصیف رسمی شبکه‌های پتری رنگی مبتنی بر زمان است و W، تأخیر نسبت داده شده به هر گذار است که به صورت یک متغیر تصادفی با توزیع نمایی‌منفی بیان می‌شود. براساس تعریف فوق، گرامر BNF این شبکه به صورت زیر است که بخش‌های مشابه با شبکه پتری رنگی مبتنی بر زمان آن حذف شده است:

T_WEIGHT ::= weight{REAL}
|weight {EXPONENTIAL_DISTR}
|ML_FUNCTION

۴-۵ شبکه‌های پتری تصادفی تعمیم یافته (*Generalized Stochastic Petri Net*). در صورتی که در شبکه‌های پتری تصادفی، زمان به بعضی از گذارهای آن، نسبت داده شود، آن شبکه، شبکه پتری تصادفی تعمیم یافته نامیده می‌شود. در اینصورت گذارها دراین شبکه به دو گروه گذارهای آنی (immediate transition) و زمانی (timed transition)، تقسیم می‌شوند که گذارهای آنی را با ▮ و گذارهای تأخیری را با ▯ ، نشان می‌دهند [۱۶].

توصیف رسمی شبکه‌های‌پتری‌تصادفی به صورت $GSPN=(P,T,I,O,H,W,prio,M_0)$ است که P، T، I، O، H، W و M_0 مشابه شبکه‌های پتری تصادفی است. با این تفاوت‌که گذارها می‌توانند از نوع زمانی یا آنی باشند. W، نیز تأخیر نسبت داده شده به هر گذار است که به صورت یک متغیرتصادفی با توزیع نمایی منفی بیان می‌شود. این تأخیر برای گذارهای‌زمانی، تأخیرآتش شدن و برای گذارهای آنی تناوب نسبی آتش شدن را نشان می‌دهد. Prio، اولویت هر گذار را نشان می‌دهد که گذارهای آنی با اولویت بزرگتر از صفر (اولویت بیشتر) و گذارهای‌زمانی با اولویت‌صفر مشخص می‌شوند. براساس تعریف فوق، گرامر BNF این شبکه به صورت زیر است که بخش‌های مشابه با شبکه مکان-گذار آن حذف شده است:

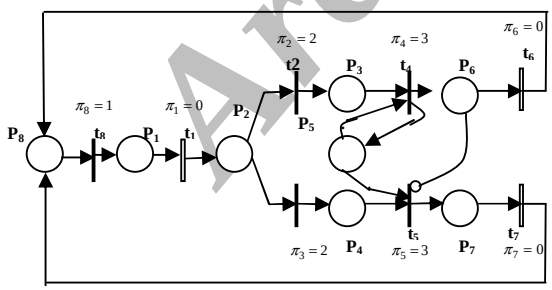
TRANSITION ::= transition {ID} { NAME PRIO T_WEIGHT}
PRIO ::= empty |prio {INTEGER}
T_WEIGHT ::= weight{REAL}
|weight {EXPONENTIAL_DISTR}
|ML_FUNCTION
ARC ::= arc{ID} {source{ID} target{ID}
WEIGHT A_TYPE}
WEIGHT ::= empty |weight{INTEGER}
|weight {MULTISETEXPR}

به عنوان مثال، شکل (۶)، یک شبکه‌پتری تصادفی تعمیم یافته را نشان می‌دهد که گرامر آن براساس تعاریف فوق، به صورت‌زیر توصیف می‌شود:

Place {p1 k}, Place {p2}, Place {p3}, Place {p4},
Place {p5 init{1}}, Place {p6}, Place {p7}, Place {p8}
transition {t1 prio{0}}, transition {t2 prio{2}},
transition {t3 prio{2}}, transition {t4 prio{3}}
transition {t5 prio{3}},
transition {t6 prio{0}}, transition {t7 prio{0}},
transition {t8 prio{1}}, transition {t9 prio{1}}
arc{a1 source{p1} target{t1} type(ordinary)}
……
arc{ai source{p6} target{t5} type(inhibitor)} …..

۵- گرامر کلی بسط‌های مختلف شبکه‌های‌پتری

همان طور که در بخش قبلی مشاهده‌می‌شود، تمام بسط‌های‌مختلف شبکه‌های پتری دارای یک ساختار کلی مشابه (یعنی مجموعه‌ای از مکان‌ها، گذارها و مهره‌ها) هستند که دسته‌ای ویژگی‌ها به این ساختار اضافه می‌گردد.



شکل۶ یک شبکه پتری تصادفی تعمیم‌یافته نمونه

این شباهت ساختاری، منجر به ایجاد یک ساختارکلی بین این شبکه‌هامی‌شود تا به راحتی قابل‌تبدیل به یکدیگر باشند. این ساختارکلی در بخش

میانی چارچوب پیشنهادی قرار می‌گیرند تا توسط فیلتر خروجی به یکی از بسط‌های شبکه پتری تبدیل گردد. درواقع در بخش میانی قوانینی قرارمی‌گیرد تا این ساختارعمومی به یک مدل تحلیل تبدیل‌گردد. با توجه به توصیفات ارائه شده در بخش قبلی برای انواع شبکه‌های پتری و شباهت‌های موجود بین آنها، توصیف رسمی این شبکه‌ها به صورت زیر نیز قابل بیان است:

شبکه‌های مکان / گذار:

$PN= (P,\ T,\ I, O, M_0)$

شبکه‌های‌پتری‌رنگی مبتنی‌برزمان:

$CPN= (PN, \Sigma, C, G, H, E)$

شبکه‌های‌پتری‌تصادفی:

$SPN= (PN,H, W)$

شبکه‌های‌پتری‌رنگی‌تصادفی:

$SCPN= (CPN,W)$

شبکه‌های‌پتری‌تصادفی تعمیم‌یافته:

$GSPN= (SPN, PRIO)$

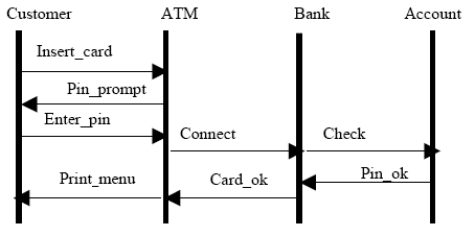
مطابق توصیفات فوق، شبکه‌های مکان/گذار ساده‌ترین نوع شبکه‌های پتری‌هستند و ویژگی‌های الحاقی به مکان‌ها، گذارها و مهره‌ها برای آنها منظور نمی‌شود. در پیاده‌سازی این گرامر عمومی، برای ویژگی‌های اضافی موجود که در یک بسط شبکه پتری قابل استفاده نیست از حالت پیش‌فرض استفاده می‌شود. از فوائد گرامر ارائه شده این‌است که در حین سادگی و قابل فهم بودن، اجازه توسعه‌های بعدی را برای در برگرفتن بسط‌های دیگر شبکه‌های پتری فراهم می‌کند. یکی دیگر از مزایای آن این است که برخلاف ساختار گرافیکی شبکه‌های پتری که تنها قابل فهم توسط انسان است، این گرامر، علاوه بر قابل فهم بودن توسط انسان، رابط مناسبی بین بسط‌های گوناگون شبکه‌های پتری نیز هست. طبق تحقیقات انجام شده در این تحقیق، توصیفات معماری حاشیه نویسی شده با پارامترهای کارایی و قابلیت‌اطمینان مطابق با الگوریتم‌های پیشنهادی و گرامرعمومی فوق به یک شبکه‌پتری‌کلی تبدیل‌می‌شود و سپس مطابق با ابزارمناسب به شبکه‌پتری‌متناسب با آن ابزار تبدیل‌می‌شود. گرامر BNF مشترک بسط‌های‌مختلف شبکه‌های‌پتری، به صورت‌زیراست:

```
ELEMENT        ::= empty   |PLACE
ELEMENT  |TRANSITION   ELEMENT
         |ARC          ELEMENT
         |TYPEDEF      ELEMENT
         |seeml {FILENAME}  ELEMENT
FILENAME       ::= STRING
PLACE          ::= place {ID} { NAME INIT
                   CAP COLOUR}
INIT           ::=  empty  |init { MULTISET}
CAP            ::=  empty  |capacity
{MULTISET}
COLOUR m       ::=  empty |colour
{COLOURSET}
COLOURSET      ::=  ML_DEFINITION
MULTISET       ::=  INTEGER
 |INTEGER' STRING MSLIST
 |INTEGER' STRING MSLIST
 @+ T_WEIGHT
MSLIST         ::=  empty  |+ MULTISET
TRANSITION  ::=  transition {ID} { NAME
GUARD   PRIO T_WEIGHT}
GUARD          ::=  empty  |guard {BOOLEXPR}
BOOLEXPR       ::=  ML_EXPRESSION
ML_EXPRESSION  ::=  EXPRESSION
EXPRESSION ::= {EXPRESSION} and
{EXPRESSION} {EXPRESSION} or
{EXPRESSION}  |not { EXPRESSION }
 |{EXPRESSION}={EXPRESSION}
|{EXPRESSION} <> {EXPRESSION}
|VALUE |ML_FUNCTION |empty VALUE ::=
value{true} |value {false}
T_WEIGHT ::= weight{REAL}
|ML_FUNCTION |weight
{EXPONENTIAL_DISTR}
EXPONENTIAL_DISTR ::= ML_EXPRESSION
PRIO  ::=  empty  |prio {INTEGER}
ARC   ::=  arc{ID} {source{ID} target{ID}
           WEIGHT A_TYPE}
WEIGHT ::=  empty  |weight {INTEGER}
|weight {MULTISETEXPR}
MULTISETEXPR  ::=  ML_EXPRESSION
A_TYPE    ::=  empty |type(ordinary)
|type(inhibitor)
```

٦- مثال کاربردی

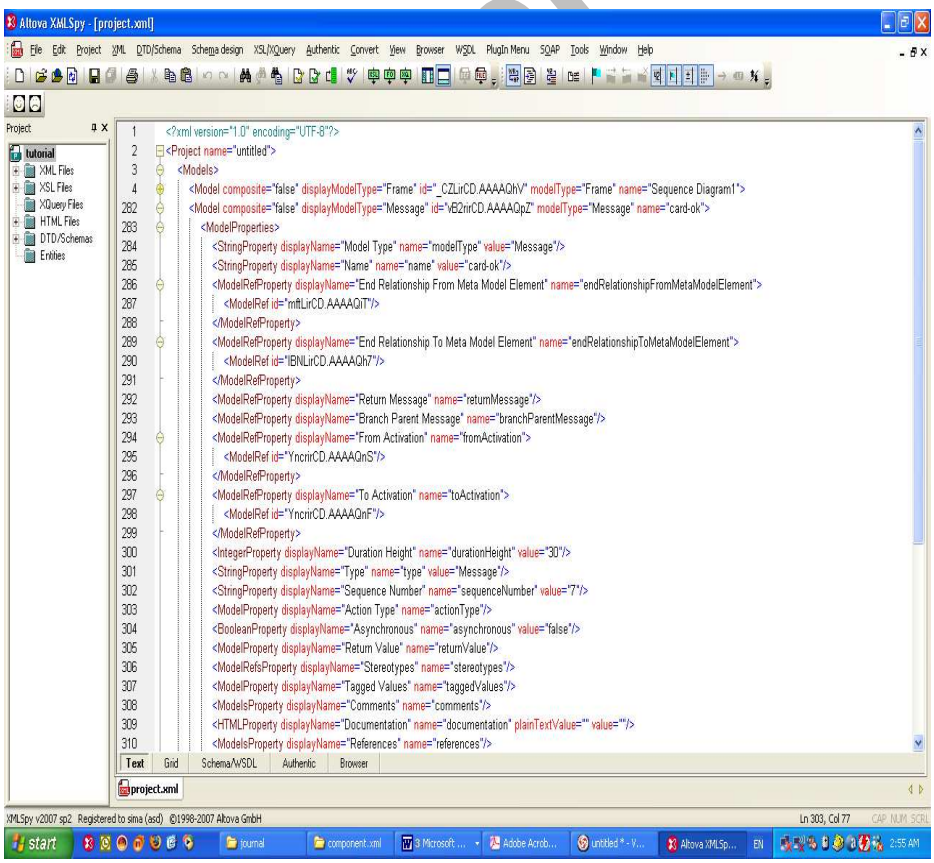
در این بخش با ارائه مثالی، کاربرد چارچوب‌پیشنهادی در طی‌مرحله توسعه‌معماری، نشان‌داده‌می‌شود. مثال استفاده‌شده در این بخش، سیستم ماشین‌خودپرداز (Auto Teller Machine (ATM))، است. شکل (۷)، نمودارترتیب مربوط به موردکاربری "شناسایی‌کاربر" را

نشان می‌دهد. دراین نمودار، ابتدا کاربر کارت را وارد دستگاه خودپرداز می‌کند و بعد از مشاهده پیغام "ورود شماره رمز" توسط دستگاه خودپرداز، رمز را وارد می‌نماید. دستگاه خودپرداز نیز به شبکه بانک، متصل می‌شود و تست رمز عبور را درخواست می‌نماید، سپس نتیجه بررسی به‌ترتیب به دستگاه خودپرداز و مشتری برمی‌گردد. این نمودارترتیب که بخشی از توصیفات معماری سیستم‌خودپرداز است با استفاده از فیلتر ورودی موجود در چارچوب به XML، تبدیل می‌شود. سپس قوانین بخش میانی، فایل حاوی توصیف معماری در قالب XML را به گرامر عمومی بسط‌های مختلف شبکه‌های پتری تبدیل می‌کند. در نهایت فیلتر خروجی فایل حاصل را به ابزار ارزیابی پشتیبان شبکه پتری موردنظر تبدیل می‌کند.

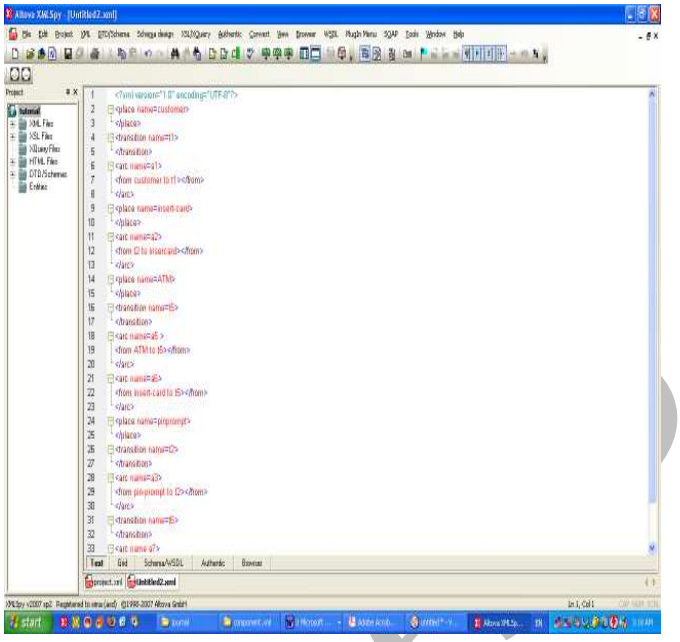


شکل ۷ نمودار ترتیب شناسایی کاربر توسط دستگاه خودپرداز

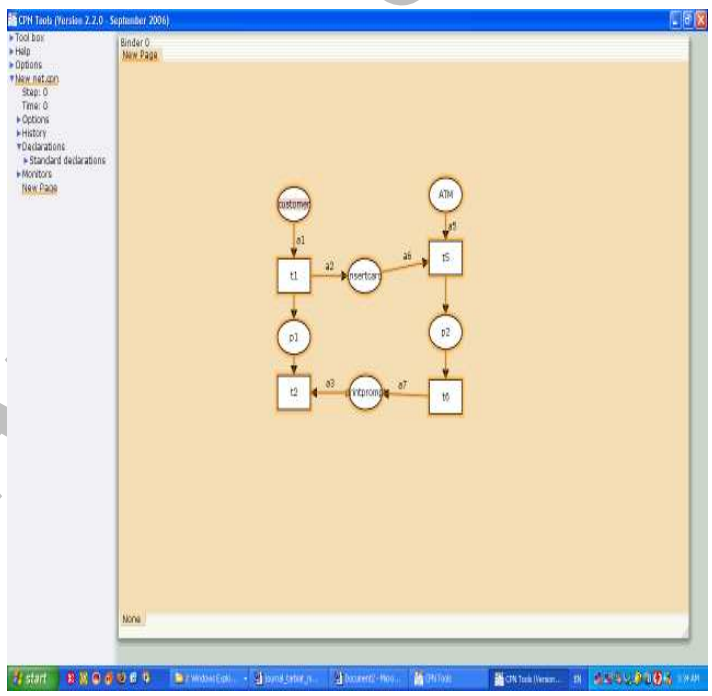
شکل (۸) بخشی از کد XML نمودار ترتیب سیستم‌خودپرداز و شکل (۹) بخشی از کد XML را که معادل گرامرعمومی است، را نشان‌می‌دهد. شکل (۱۰) نیز شبکه پتری رنگی حاصل از فیلترخروجی را با توجه به ابزار پشتیبان آن یعنی CPN tools، نشان‌می‌دهد.



شکل ۸ XML حاصل از فیلتر



شکل ۹ گرامرعمومی حاصل از کد **XML** شکل (۸) در قالب کد **XML**



شکل ۱۰ شبکه‌پتری حاصل از نمودار ترتیب شکل (۹) در ابزار **CPN Tools**

۷- نتیجه‌گیری

یکی از مهمترین سطوح ارزیابی نیازهای‌وظیفه‌مندی و غیروظیفه‌مندی در طی فرآیند توسعه نرم‌افزار، مرحله معماری نرم‌افزار است. برای اجتماع ارزیابی در مرحله معماری، مدلی اجرایی از معماری ایجاد می‌شود و پارامترهای لازم برای ارزیابی به این مدل اضافه می‌گردد. در این مقاله چارچوبی مبتنی بر معماری مدل‌رانه ارائه‌گردید که ارزیابی و تحلیل نیازمندی‌ها را در طی توسعه معماری انجام می‌دهد.

از اهداف این چارچوب، توصیف‌معماری با هر علامت‌گذاری یا زبان توصیف معماری و ارزیابی آن با هر نوع مدل اجرایی یا ابزار ارزیابی است. این چارچوب که مبتنی بر معماری مدل‌رانه است، در سطح اول از معماری، یک مدل مستقل‌ازبستر ایجادمی‌کند و آن را در سطح دوم به یک مدل مبتنی بر XML تبدیل‌می‌نماید. در سطح دوم، افزون بر نمایش مبتنی بر XML، قوانین و روابط میان موجودیت‌های علامت‌گذاری‌های مختلف مشخص‌می‌شود تا نتایج به مدل نرم‌افزاری سطوح قبلی بازخورد شود. درنهایت، مدل به یک مدل‌اجرایی، برای تحویل به ابزار ارزیابی مناسب، تبدیل‌می‌گردد. در بخش میانی چارچوب از یک گرامرعمومی مبتنی بر بسط‌های مختلف شبکه‌های پتری استفاده می‌شود که این گرامرمشترک، عمومیت این چارچوب را برای استفاده توسط بسط‌های مختلف شبکه‌پتری و ابزارهای تحلیل گسترش‌می‌دهد. همان طور که مشاهده می‌شود معمار تنها بایستی اطلاعات مربوط به کارایی و قابلیت‌اطمینان را به توصیفات معماری اضافه نماید تا به طور خودکار به یکی از بسط‌های شبکه‌پتری تبدیل‌گردد. بدین وسیله فاصله دانشی مابین معمار و تحلیل‌گر با استفاده از این چارچوب به حداقل خود می‌رسد. ضمن اینکه با بازخورد نتایج حاصل از ارزیابی و با استفاده از قوانین موجود در سطح میانی چارچوب، می‌توان تغییرات‌لازم را به توصیفات‌معماری اعمال‌نمود.

مراجع

1. Bass, Clements, and kazman, "Software Architecture in Practice, Addison Wesely", (2002).

2. Clements, Kazman and Klein, "Evaluating Software Architecture Methods and Case Studies", Addison Wesely, (2002).

3.ANSI/IEEE Standard 729-1983. "Software Engineering Standards". IEEE, New york, (1989).

4. Ammar, Nikzadeh and Dugan, "Risk Assessment of Software-System Specification", Reliability IEEE Transactions, Vol. 50, Issue 2, pp. 171-183., (2001).

5. Balsamo, D.M, Inverardi, and M. Simeoni, "Model-Based Performance Prediction in Software Development: A Survey", IEEE Transaction on Software Engineering, , NO. 5, pp. 295-310, (2004).

6. UML 2.0 Superstructure Specification, OMG Adopted Specification ptc/ 03 - 08 - 02, available: www.omg.org/docs/ptc/03-08-02.pdf.

7. Dutoit, Kerkow, Paech, and Von, "Functional requirements, nonfunctional requirements", and rchitecture should not be separated, 8th International Workshop on Requirements Engineering: Foundation for Software Quality, Essen, Germany, September 9-10, pp. 102-107, (2002).

8. Cysneiros, De Melo, and Do Prado, "A framework for integrating non-functional requirements into conceptual models, Requirements Engineering journal", Vol. 6, No. 2, Springer-Verlag London Limited, pp. 97–115, Apri (2001).

9. Cortellessa, Di Marco, Inverardi, Mancinelli and Pelliccione, "A framework for the integration of functional and non-functional analysis of software architectures, 2nd International Workshop on Test and Analysis of Component Based Systems", Barcellona (Spain), ENTCS, Vol 116, pp. 31-44, Electronic Notes in Theoretical Computer Science, available: **www.elsevier.nl/locate/entcs**, (2004).

10. Emadi, Shams, "An approach to Non-Functional Requirements Analysis at Software Architecture Level", IEEE, 8th international conference on computer and information technology, pp. 736-741, (2008).

11. Billington, Christensen, and etc. , "The Petri Net Markup Language: Concepts, Technology", and Tools, Proceedings of the 24th International Conference on Applications and Theory of Petri Nets, ICATPN 2003, Eindhoven, The Netherlands, June 23-27, pp. 483-505, (2003).

12. Bause, Kemper, "Kritzinger, Abstract Petri Net Notation, Petri Net Newsletter", Vol 49, pp.9-27, (1995).

13. Kontio, "Architectural Manifesto: MDA for the Enterprise: An architect's approach to more productive development", IBM publisher, Jul (2005).

14. Murata, "Petri Nets: Properties, Analysis, and Applications", Proceedings of the IEEE, pp. 541-580, (1995).

15. Jensen, "Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use", EATCS onographs on Theoretical Computer Science, Vol I, (1997).

16. Bernardi, "GreatSPN User's Manual (version 2.0.2)", Technical Report in university of Torino (Italy), (2002).

17. Y. Yu, H. Deng, Mo, "A Formal Method for Analyzing Software Architecture Models in SAM", IEEE, Proceedings of the 26 th Annual International Computer Software and Applications Conference (COMPSAC'02), (2002).

18. Kristensen, Christensen, Jensen, "The Practitioner's Guide to Coloured Petri Nets, Springer-Verlag", pp. 98-132, (1998).

19. Grassi, Mirandola, Sabetta, "A Model-Driven Approach to Performability Analysis of Dynamically Reconfigurable Component-Based Systems", WOSP'07, pp. 103- 114, (2007).

20. Grassi, Mirandola, "Sabetta, Filling the gap between design and performance/reliability models of component-based systems: A model-driven approach", The Journal of Systems and Software, pp. 528–558, (2007).