علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۸، شماره ۱ و ۳ (الف)، ۱۳۸۹
صفحات ۲۶-۳۶
مقاله عادی

مولد موتور گردش کار SWEG

مرتضی یوسف صنعتی[۱] سید حسن میریان حسین‌آبادی[۲]

[۱]دانشکده مهندسی، دانشگاه بوعلی سینا، همدان، ایران
[۲]دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران

چکیده

بسیاری از سازمان‌ها برای انجام مسئولیت‌های خود از گردش کارهای مختلفی استفاده می‌کنند. لذا تولید کنندگان نرم‌افزار با توجه به این موضوع، از موتورهای گردش‌کار نرم‌افزاری برای سرویس‌دهی به اعمال پایه‌ای و سیاست‌های گردش کارهای موجود در سازمان‌ها استفاده می‌نمایند. از طرف دیگر تولید نرم‌افزارهای موتور گردش‌کار که کلیه نیازهای گردش‌کاری سازمان‌ها را پاسخ دهد، نیاز به صرف هزینه و زمان زیادی دارد. در این میان با حضور روش‌های مهندسی خط محصول[۱] برای تولید نرم‌افزار مانند روش تولید نرم‌افزار مبتنی بر خانواده[۲]، هزینه تولید بسیار کاهش یافته و در زمان بسیار کمتری نرم‌افزار با کیفیتی بسیار مطلوب ایجاد می‌گردد. در این مقاله نحوه ایجاد نرم‌افزارهای موتور گردش‌کار، با استفاده از روش مبتنی بر خانواده بررسی شده و با ارایه یک زبان مدل‌سازی مخصوص اعضا خانواده موتورهای گردش‌کار، مشکلات تولید چنین نرم‌افزارهایی به میزان قابل ملاحظه‌ای کاهش یافته است.

کلمات کلیدی: تولید نرم‌افزار مبتنی بر خانواده، زبان مدل‌سازی کاربرد، موتور گردش کار.

۱- مقدمه

در مهندسی نرم‌افزار روش‌های متفاوتی برای تولید سامانه‌های نرم‌افزاری وجود دارد که هر یک دارای مزایا و معایبی می‌باشند و بسته به شرایط و اهداف، می‌توان یکی از آن‌ها را به خدمت گرفت. از جمله آن‌ها می‌توان به روش‌های مبتنی بر شی‌گرایی مانند RUP[۳] و روش‌های ساخت‌یافته مانند SSADM[۴] اشاره نمود. در تمام روش‌های موجود مراحل و گام‌های پیش‌بینی‌شده‌ای وجود دارد که با انجام آن‌ها می‌توان به محصول نهایی دست یافت. در این میان نیروها و کارشناسان با ایفای نقش‌های خود، تعامل با دیگر نقش‌ها و همکاری با دیگران انجام مراحل را عهده‌دار می‌باشند. بعلاوه در همه روش‌ها گام‌های اساسی تولید نرم‌افزار از جمله تحلیل، طراحی و پیاده‌سازی وجود دارد.

ابتدا نیازهای مشتری تبیین می‌شود، سپس طراحی انجام شده و در نهایت پیاده‌سازی صورت می‌پذیرد. بعد از این مرحله نیز تست انجام خواهد شد. در این میان نکته حائز اهمیت، تعداد مراحل، تعداد نقش‌ها و نیروهای لازم برای تولید محصول، محصولات میانی، زمان لازم برای تولید، کیفیت محصول، هزینه، قابلیت استفاده مجدد، انعطاف‌پذیری و قابلیت تغییر می‌باشد. البته در این میان روش‌های دیگری نیز وجود دارند که می‌توانند در زمان کوتاه و با هزینه کم سامانه‌هایی مطلوب ایجاد نمایند که یکی از این روش‌ها، روش تولید نرم‌افزار مبتنی بر خانواده است. این روش دارای مزایای مهمی است که در بخش بعدی بطور کامل در مورد آن توضیح داده خواهد شد.

در روش تولید نرم‌افزار مبتنی بر خانواده اصل بر شناخت خانواده‌ای از نرم‌افزارها است که دارای مشابهت‌های زیادی هستند و می‌توان اعضای آن‌ها را با این روش تولید نمود. در این میان یکی از دامنه‌های مطلوب، دامنه سامانه‌های حاوی گردش کار می‌باشد. در مورد دامنه گردش کار و جزییات مربوط به آن در بخش‌های بعدی توضیح داده شده است. ولی لازم به ذکر است که اعضای این دامنه بسیار زیاد هستند و دارای مشابهت‌های بسیاری نیز می‌باشند.

این دو خاصیت کلیدی باعث می‌شود تا با استفاده از روش مورد بحث بتوان سامانه‌های گردش کار را در زمانی کوتاه و با هزینه‌ای اندک تولید نمود. هدف این

مقاله بررسی و تولید مولد موتور گردش کار با استفاده از روش تولید نرم‌افزار مبتنی بر خانواده است. ابتدا روش تولید نرم‌افزار مبتنی برخانواده به همراه مزایا و معایب آن بررسی می‌شود. سپس جزییات مربوط به دامنه گردش کار مطرح خواهد شد. همچنین مزایای اعمال این روش بر روی دامنه کار نیز بوضوح قابل مشاهده خواهد بود. سپس زبان گردش کار، معماری مربوط به مولد موتور گردش کار و مثالی از نحوه ایجاد یک موتور نیز ارائه می‌گردد.

۲- روش تولید نرم‌افزار مبتنی بر خانواده

در بیشتر سازمان‌ها تحویل دیرهنگام سامانه‌های مورد نیاز و نیز گران بودن تولید نرم‌افزارها از مهمترین مشکلات به شمار می‌رود. از این رو وجود روش‌هایی که بتوانند با سرعت و بدون هزینه زیاد نیازهای مشتریان را پاسخگو باشند، ضروری به نظر می‌رسد. یکی از این روش‌ها، روش تولید نرم‌افزار مبتنی بر خانواده است که در آن فرایندها و عناصر خاصی جهت رسیدن به اهداف مهمی که از جمله آن‌ها افزایش کارایی پردازه تولید [۱]، بهبود کیفیت محصول و خدمات سازمان‌ها [۱]، تولید سریع، دقیق و کم هزینه سامانه‌ها می‌باشد، تعبیه شده است.

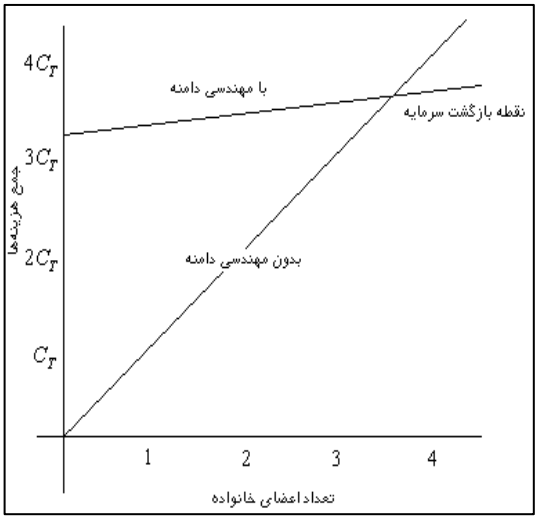
به عبارت دیگر این روش الگویی برای پردازه‌های تولید نرم‌افزار است که در آن از فشار بین تولید سریع و مهندسی مطمئن و دقیق کاسته شده است. به عبارت دیگر این روش یک پردازه سیستماتیک برای توسعه مجموعه‌ای از نرم‌افزارها است که دارای خصوصیات مشترک زیادی می‌باشند [۲] و می‌تواند در حل مشکلات زیادی به مشتریان و توسعه‌دهندگان نرم‌افزار کمک نماید. در زیر تعدادی از این مشکلات ذکر شده‌اند.

۱. عدم رضایت از کارایی پروژه یا محصول موجود
۲. نیاز به کاهش هزینه و زمان تولید
۳. کمبود کارمند
۴. نیاز به پاسخگویی سریع به درخواست‌کننده نرم‌افزار
۵. پیچیدگی مدیریت و نگهداری بسیاری از محصولات نرم‌افزاری

نکته قابل توجه این است که بطور نمونه شرکت‌های بزرگی مانند Nokia، HP و Philips با استفاده از این روش توانسته‌اند زمان و هزینه مهندسی را کاهش داده و نرخ تولید را از ۳ به ۵۰ برسانند [۳].

به طور کلی در این روش خانواده‌ای از نرم‌افزارها شناسایی می‌گردد. منظور از خانواده، مجموعه‌ای از نرم‌افزارهایی است که دارای مشترکات زیادی هستند یا به عبارت دیگر مشترکات آن‌ها می‌توانند هسته اصلی هر عضو این مجموعه را تشکیل دهند. بنابراین با شناخت خانواده و اعضای آن و تحلیل جنبه‌های مشترک می‌توان گام‌های ابتدایی این روش را به انجام رساند. اما قبل از تحلیل بررسی جنبه‌های اقتصادی موضوع جهت سرمایه‌گذاری روی خانواده از اهمیت ویژه‌ای برخوردار است. از این رو پس از سرمایه‌گذاری، تحلیل دامنه و انجام یکسری فعالیت‌ها و تولید محصول میانی، ابزارهایی تولید می‌گردند که توسط آن‌ها به راحتی می‌توان هر عضوی از خانواده مورد نظر را به سرعت، دقت و کیفیت مطلوب تولید نمود. یکی از دلایلی که این روش به سرعت می‌تواند محصول خود را تولید نماید، زمان لازم برای تولید یک عضو از خانواده است.

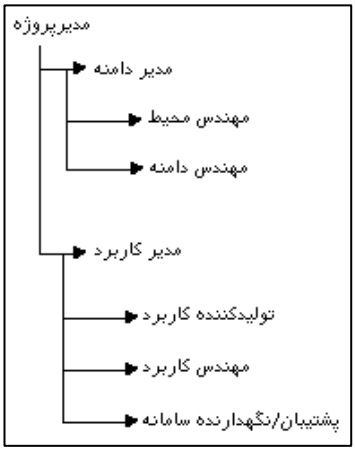
در مقایسه روش تولید نرم‌افزار مبتنی بر خانواده با سایر روش‌ها می‌توان به کمتر بودن هزینه و زمان تولید در این روش دست یافت. اگر تولید هر محصول (T) واحد زمان ببرد و نیز تولید N محصول مد نظر باشد، زمان لازم برای تولید آن‌ها در سایر روش‌ها T*N خواهد بود. شبیه همین مطلب نیز در مورد هزینه تولید نیز برقرار است، یعنی اگر هزینه لازم برای تولید محصول C فرض شود و N محصول مورد نیاز باشد، لذا کل هزینه برابر C*N خواهد بود و این در حالی است که در روش مورد مطالعه بحث هزینه و زمان به گونه‌ای دیگر می‌باشد.



شکل ۱- وجود یا عدم وجود مهندسی دامنه

در این روش اگر هزینه لازم برای انجام مهندسی دامنه I فرض گردد، هزینه تولید هر نرم‌افزار پس از انجام مهندسی دامنه C_f می‌شود. لذا اگر مهندسی دامنه با موفقیت انجام شده باشد، $C_f < C$ خواهد بود. پس کل هزینه $I+C_f*N$ می‌شود. نشان داده شده است که برای $N > 3$ میزان کل هزینه در صورت داشتن مهندسی دامنه موفقیت‌آمیز، کمتر از عدم وجود مهندسی دامنه است، پس $I+C_f*N < C*N$ [۴]. این مطلب در شکل ۱ قابل مشاهده می‌باشد.

همچنین در مورد زمان، می‌توان گفت که زمان لازم برای تولید اولین محصول توسط این روش، بدلیل انجام مهندسی دامنه زیادتر از دیگر روش‌ها می‌باشد، یعنی اگر زمان لازم برای انجام مهندسی دامنه، تولید محیط و توصیف کاربرد توسط زبان مدل‌سازی به ترتیب T_d، T_e و T_s باشد، لذا $T < T_d + T_e + T_s$ خواهد بود. اما بدلیل این‌که مهندسی دامنه و تولید محیط یکبار صورت می‌پذیرد، پس برای تولید محصولات بعدی زمان لازم فقط T_s یعنی همان زمان لازم برای توصیف کاربرد توسط زبان مدل‌سازی خواهد بود، لذا کاربردهای بعدی سریعتر از سایر روش‌ها، تولید می‌شوند و در نهایت $T_d+T_e+T_s*N < T*N$ می‌باشد.



شکل ۲- نقش‌ها در روش تولید نرم‌افزار مبتنی بر خانواده [۵]

در روش مورد بحث نقش‌های متفاوتی فعالیت می‌کنند. این نقش‌ها در شکل ۲ نشان داده شده‌اند. یکی از آن نقش‌ها مهندس دامنه است. او خانواده‌ای را که ارزش سرمایه‌گذاری داشته باشد، پیدا می‌کند. یعنی دامنه را مشخص می‌نماید.

وی همچنین به بررسی اعضای متفاوت دامنه می‌پردازد و مشترکات و تمایزات آن‌ها را به خوبی درمی‌یابد. این عمل تحلیل مشترکات و تمایزات نامیده می‌شود.

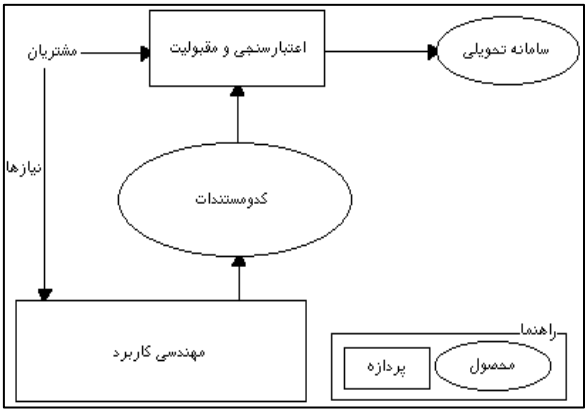
گام بعدی، مدل‌سازی اعضای خانواده و ایجاد مدل سراسری است که بر طبق تحلیل‌های بدست آمده صورت می‌گیرد. مهندس دامنه با طراحی زبانی که زبان مدل‌سازی کاربرد نامیده می‌شود، خانواده را مدل می‌نماید. تاکنون مهندس دامنه دو محصول ایجاد کرده است. اول مستندات مربوط به تحلیل مشترکات و تمایزات و دوم زبان مدل‌سازی کاربرد. ولی برای استفاده از زبان و تولید محصول توسط آن، دو راه حل وجود دارد. اولین راه حل ایجاد یک مترجم برای زبان مدل‌سازی است تا با استفاده از آن بتوان توصیفاتی را که نوشته می‌شود ترجمه، و در نهایت به یک محصول عملیاتی تبدیل کرد. راه حل دوم ایجاد یک ترکیب‌کننده است.

کار این ترکیب‌کننده الحاق‌کردن اجزاء کوچک نرم‌افزاری است که در تناظر با دستورات زبان مدل‌سازی می‌باشند. پس از الحاق، محصول عملیاتی مطلوب تولید می‌گردد. در صورت استفاده از ترکیب‌کننده چند نکته حائز اهمیت است. ابتدا باید یک کتابخانه از اجزاء نرم‌افزاری پیاده‌سازی شده که در اعضای دامنه وجود دارند ایجاد کرد و دوم یک تناظر بین دستورات زبان مدل‌سازی و اجزاء نرم‌افزاری برقرار نمود تا ترکیب‌کننده بتواند با توجه به تناظر اجزاء مطلوب را انتخاب نماید.

پس از تولید مترجم یا ترکیب‌کننده، نکته مهم ایجاد محیطی برای استفاده و توصیف نمودن کاربرد است. تولید این محصول نیز بر عهده مهندس دامنه است و وی باید محیطی را جهت ایجاد توصیف نرم‌افزار در اختیار دیگر نقش‌ها قرار دهد. تنها مشکل باقیمانده چگونگی فرایند تولید نرم‌افزار با استفاده از این محصولات است که در این میان مهندس دامنه با طراحی فرایند تولید نرم‌افزار و مستندسازی آن، مشکل را رفع می‌نماید. در این مرحله تقریبا مهندس دامنه تمام نقش خود را ایفا نموده است و نوبت به ایفای نقش توسط مهندس کاربرد می‌رسد.

مهندس کاربرد نیازهای مشتری را بخوبی تحلیل می‌نماید. سپس با استفاده از زبان مدل‌سازی تولیدی، نرم‌افزار مورد نیاز مشتری را در محیطی که برای وی توسط مهندس دامنه تولید شده است، توصیف می‌نماید. در نهایت با استفاده از مترجم یا ترکیب‌کننده نرم‌افزار مورد نیاز مشتری را تهیه می‌کند. البته قبل از تحویل، ابتدا از صحت عملکرد آن با جلب رضایت مشتری اطمینان حاصل می‌نماید. در صورت وجود کاستی و عدم رضایت مشتری، نقص‌های موجود را با بهبود و اصلاح توصیف، رفع نموده و مجددا نرم‌افزار را تولید می‌نماید و به مشتری تحویل می‌دهد.

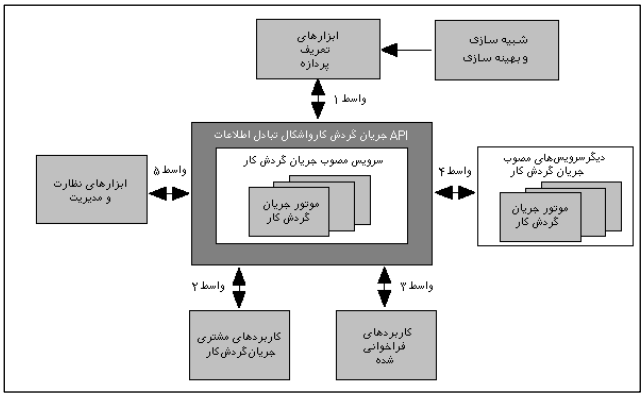
فرایند تولید محصول در شکل ۳ نشان داده شده است. پس از ارائه این روش، در بخش بعد سامانه مدیریت گردش کار و مدل مرجع آن مورد تحلیل و بررسی قرار خواهد گرفت.



شکل ۳- فرآیند تولید محصول

۳- سامانه مدیریت گردش کار و مدل مرجع آن

همه سازمان‌ها دارای اهدافی هستند و برای رسیدن به آن اهداف، پردازه‌های متفاوتی را اجرا می‌نمایند. این پردازه‌ها از فعالیت‌های سازمانی که برای رسیدن به هدف با ترتیب مشخصی اجرا می‌شوند، تشکیل شده‌اند. از این رو خودکارسازی این پردازه‌ها کمک شایانی به افزایش سرعت انجام کارها و کارایی سازمان‌ها می‌نماید [۶]. بنابراین خودکارسازی تمام یا قسمتی از پردازه‌ها در مدتی که اسناد، اطلاعات یا وظایف از یک همکار به همکار دیگر جهت انجام عملیات بر طبق مجموعه‌ای از قوانین سازمانی فرستاده می‌شود، گردش کار نامیده می‌شود. ولی گردش کار باید مدیریت شود که این امر بر عهده سامانه‌های مدیریت گردش کار است که در این میان مفهوم مهم مدیریت گردش‌کار به عنوان یک فناوری کلیدی برای هماهنگ نمودن پردازه‌های تجاری متفاوت نقش بسزایی را ایفا می‌نماید [۷]. پس سامانه‌های مدیریت گردش کار علاوه بر مدیریت، قابلیت تعریف پردازه، اجرا، مدل‌سازی [۸] و هماهنگ کردن آن را دارا می‌باشند. لازم به ذکر است که پردازه‌های مورد بحث باید از روی مدل‌های سازمانی توصیف شوند [۸]. همچنین این سامانه‌ها فعالیت‌ها و روند آن‌ها را مدیریت نموده، در زمان‌های معین یا شرایط معین اعمال لازم را انجام می‌دهند.



شکل ۴- مدل مرجع سامانه‌های گردش کار [۹]

شکل ۴ مدل مرجع اغلب سامانه‌های مدیریت گردش کار را نشان می‌دهد که در مورد هر یک از اجزاء مدل بطور مختصر توضیح داده می‌شود.

خدمت مصوب گردش کار

خدمت مصوب گردش کار یک محیط اجرایی را فراهم می‌آورد که در آن پردازه ایجاد می‌شود، فعال می‌گردد و از یک یا چند موتور گردش کار بهره می‌گیرد و مسئولیت تعامل با منابع خارجی را در صورت لزوم بر عهده دارد.

موتور گردش کار

موتور محیطی اجرایی برای نمونه گردش کار فراهم می‌کند و نوعاً امکانات زیر را نیز فراهم می‌آورد:

- تفسیر تعریف پردازه‌ها
- ایجاد، کنترل و نمونه‌سازی پردازه‌ها
- ورود همکار به سامانه و خروج از آن
- انجام اعمال مدیریتی
- انجام اعمال نظارتی
- ناوبری بین فعالیت‌های ترتیبی یا موازی

در اصل مهمترین جزء سامانه همین موتور است و مهمترین کارها را موتور انجام می‌دهد.

ابزارهای تعریف پردازه

این ابزارها قابلیت تحلیل، مدل کردن و مستندسازی پردازه‌ها را فراهم می‌کنند، بطوریکه خروجی حاصل از این ابزارها توسط موتور گردش کار قابل تفسیر و اجرا است [۱۰].

ابزارهای نظارت و مدیریت

این ابزارها برای نظارت بر حسن اجرای پردازه‌ها و نیز مدیریت آن‌ها تعبیه شده‌اند و گزارش‌های متنوعی را در مورد پردازه‌ها، فعالیت‌ها، حالات آن و سایر موارد مطلوب دیگر به مدیر یا مدیران سامانه ارائه می‌نمایند.

کاربردهای فراخوانی شده

در بعضی موارد برای انجام پردازه لازم است کاربردی خاص، فراخوانی شود. در این صورت واسط کاربری خاصی برای این امر تعبیه شده است که با استفاده از آن، خدمت مصوب گردش کار نیازمندی‌های کاربرد را به وی داده و اطلاعات لازم را از آن دریافت می‌کند.

کاربردهای مشتری گردش کار

کاربردهای مشتری، کاربردهایی هستند که مشتری گردش کار را در رفع نیازهای متفاوتی از جمله یادآوری و نمایش وظایفی که به آن‌ها محول شده، پیام‌هایی که به آن‌ها رسیده، یاری می‌دهند.

دیگر خدمت‌های مصوب گردش کار

خدمت مصوب سامانه می‌تواند با خدمت‌های مصوب دیگر سامانه‌ها در ارتباط باشد. لذا واسطی نیز برای این کار تعبیه شده است تا خدمت‌های مصوب با کمک آن بتوانند نیازهای خود مانند ردوبدل اطلاعات را مرتفع نمایند.

در اینجا بطور مختصر مدل مرجع توضیح داده شد که برای مطالعه بیشتر می‌توان به [۱۰،۱۱] مراجعه نمود.

۴- بررسی خانواده سامانه‌های حاوی گردش کار

در بخش‌های قبلی، روش تولید نرم‌افزار مبتنی بر خانواده و سامانه مدیریت گردش کار به همراه مدل مرجع آن بطور مستقل مورد بررسی قرار گرفتند. در این بخش با استفاده از روش مطرح شده، خانواده سامانه‌های حاوی گردش کار مورد مطالعه قرار می‌گیرد.

تعداد اعضای این خانواده بسیار زیاد است. زیرا هر سامانه‌ای که به نوعی حاوی گردش کار باشد، عضو این خانواده خواهد بود. لذا بطور نمونه می‌توان به موارد زیر اشاره نمود.

- سامانه‌های مربوط به اتوماسیون عملکرد ادارات، انبارها، شرکت‌های خصوصی و دولتی و نهادهای اجتماعی دیگر مانند تعاونی‌ها و انجمن‌ها
- سامانه‌های مربوط به امور مزایده و مناقصه
- سامانه‌های مدیریت فعالیت‌ها در کارخانه‌های متفاوت
- سامانه‌های آموزش در دانشگاه‌ها و سایر مراکز آموزشی غیر آموزش عالی
- سامانه‌های بیمارستان‌ها

با توجه به این‌که فقط به تعدادی از سامانه‌های گردش کار اشاره شد، در حالیکه همین موارد تعداد زیادی از سازمان‌ها را تحت پوشش قرار می‌دهد، لذا تعداد اعضای این خانواده بسیار زیاد است. بنابراین با توجه به توضیحاتی که در مورد روش تولید نرم‌افزار مبتنی بر خانواده ارائه گردید، این روش می‌تواند اعضای این خانواده را با زمان و هزینه اندک ایجاد نماید.

از این رو با توجه به روش مذکور، مهندس دامنه پس از شناخت اجمالی دامنه، یک مدل اقتصادی تهیه می‌کند تا تصمیمات بعدی طبق آن اتخاذ شود. ولی در اینجا بدلیل وضوح اقتصادی بودن این کار، از تهیه مدل اقتصادی دامنه صرف‌نظر می‌گردد. گام بعدی تحلیل مشترکات و تمایزات است که مهندس دامنه باید آن را انجام دهد. پس از مطالعه روی سامانه‌های متفاوت و بررسی استانداردهای در دست تدوین برای سامانه‌های گردش کار، مشترکات و تمایزات بدست آمده به تفکیک از قرار زیر می‌باشد:

❖ **مشترکات**

- وجود پردازه‌های مختلف
- وجود وظایف و فعالیت‌های متفاوت
- وجود کنشگرهای سازمانی
- وجود اقلام گردش کار
- وجود تصمیم‌گیری مبتنی بر قاعده
- کنترل و نمونه‌سازی پردازه‌ها
- کنترل و نمونه‌سازی فعالیت‌ها
- وجود لیست کارها و وظایف
- تخصیص‌دهی وظایف
- وجود ابزارهای تعریف و تفسیر پردازه [۱۲] و فعالیت‌های آن
- وجود ابزارهای مربوط به تعریف و تفسیر منطق مسیرگزینی اقلام گردش کار
- ایجاد امنیت [۱۳] با هویت‌شناسی و کنترل دسترسی [۱۴] برای انجام وظایف محوله بصورت نقش محور [۱۵] یا سیاست‌های دیگر

❖ **تمایزات**

- تعداد و نحوه عملکرد پردازه‌ها
- تعداد و نحوه عملکرد فعالیت‌ها
- تعداد و ویژگی‌های کنشگرها
- اقلام گردش‌کار با صفات متفاوت
- منطق مسیرگزینی[۱۶]
- منطق اتخاذ تصمیم
- گزارش‌گیری‌های متنوع
- انجام تارنگاری
- وجود و شکل الگوهای رفتاری [۱۷،۱۸،۱۹] متفاوت از جمله And-Join, Or-Join [۲۰]

موارد ذکرشده که نتایج تحلیل مشترکات و تمایزات می‌باشد به عنوان ورودی گام بعدی روش مورد بحث استفاده می‌شود. مشترکاتی که بین سامانه‌های گردش کار وجود داشت، به عنوان قسمت پایه‌ای و الزامی همه اعضا در تولید آن‌ها نقش کلیدی ایفا می‌نماید. از این رو باید موارد مشترک پیاده‌سازی گردیده، در هنگام تولید یک عضو از خانواده به آن متصل گردد. اکنون باید زبانی برای تولید تمایزات اعضا طراحی گردد. پس از طراحی زبان، مترجم آن و محیط تولید، مهندس کاربرد می‌تواند با استفاده از آن، هر یک از ویژگی‌های مورد نظر را برای سامانه مطلوب مشتری، تولید نماید. لذا طراحی زبان کاری بسیار مهم است که باید با دقت انجام شود. در قسمت بعد جزییات مربوط به زبان گردش کار توضیح داده می‌شود.

۵- طراحی زبان جهت مدل‌سازی دامنه گردش کار

زبان مدل‌سازی کاربرد مبتنی بر نتایج گام تحلیل مشترکات و تمایزات طراحی شده و در صورت لزوم اصلاحاتی نیز روی آن انجام می‌شود. با توجه به دامنه سامانه‌های حاوی گردش کار و نتایج تحلیل، زبان مذکور دارای اجزاء و قواعد زیر است.

- تعریف مدل سازمان

هر سازمان دارای یک نام است و از چندین بخش تشکیل شده است که در آن بخش‌ها کارمندانی به ایفای نقش می‌پردازند که هر یک از آن‌ها نیز شرح وظایفی دارند. با انجام کارها و وظایف توسط کارمندان هدف سازمان محقق می‌گردد. بنابراین در این میان انتقال اسناد، اطلاعات بین کارمندان از مهمترین مباحث گردش کار در سازمان است. البته این انتقالات باید در قالب فرآیندهای سازمانی و ارضای شرایط خاصی صورت پذیرد تا در زمان مقتضی اطلاعات لازم در دسترس فرد مجاز و مطلوب قرار گیرد. از این رو در تعریف مدل سازمانی اقلام زیر باید تعریف گردد.

۱) نام سازمان
۲) تعریف کنش‌گرها و موارد مرتبط با آن‌ها
۳) تعریف نرم‌افزارهای سازمان
۴) تعریف داده‌های مرتبط
۵) تعریف موجودیت‌های سازمانی
۶) تعریف توابع لازم برای انجام کارها
۷) تعریف موارد لازم برای ثبت عملکردها
۸) تعریف گزارشات متنوع
۹) تعریف پردازه‌ها، فعالیت‌های آن‌ها و انتقالات بین فعالیت‌ها

OrganizationModel ::=	< ORGANIZATIONNAME >
WorkflowActorSpecification	**O**rganizationChartDefinition
DepartmentHasRole	**G**roupHasMember
HumanHasRole	**W**orkflowApplicationDefinition
WorkflowRelevantData	**O**bjectDefinition
LibraryDeclaration	**L**ogginDefinition
ReportDefinition	**W**orkflowProcessDefinition

شکل ۵- تعریف مدل سازمان

- تعریف کنش‌گرها

بطور کلی کنش‌گرها به پنج گروه تقسیم می‌شوند که عبارتند از بخش سازمانی، نقش سازمانی، کارمند، سیستم و گروه. به دلیل اینکه در سازمان‌های متفاوت، کنش‌گرها دارای خصوصیاتی هستند که منحصر به آن سازمان است، لذا ابتدا باید خصوصیات کنش‌گرها بطور کلی مشخص گردد (شکل ۶). سپس همه کنش‌گرهای موجود در سازمان با توجه به خصوصیات مطلوب که قبلاً تعریف گردید، توسط قواعد موجود در شکل ۷ مشخص شوند.

ذکر این نکته حائز اهمیت است که ممکن است در میان پردازه‌های سازمان لازم باشد تا جزیی از کار توسط یک نرم‌افزار خاص یا یک سیستم موجود انجام شود، بنابراین باید این سیستم به عنوان یک کنش‌گر معرفی گردد. به همین علت یک از انواع کنشگرها، سیستم در نظر گرفته شده است. در این میان کنش‌گر گروه نیز فقط به عنوان یک دسته‌بندی کننده برای سایر کنش‌گرهاست و هویت مستقلی ندارد. بنابراین‌می‌توان کنش‌گرهای سازمان را گروه‌بندی نمود. البته همانطور که قبلاً ذکر شد، در رابطه با کلیات کنش‌گرهای سازمان ذکر مواردی مانند تعریف نمودار سازمانی، انتساب نقش‌ها به بخش‌های سازمان، تعریف کارمندان و انتساب نقش‌ها به آنان نیز دارای اهمیت است. به همین دلیل قواعدی نیز برای این امر تعبیه گردیده که در شکل ۸ تا شکل ۱۱ قابل مشاهده می‌باشند.

```
WorkflowActorSpecification ::=
        ExtendedAttributeDefinition
                ACTORS{ WorkflowActorDeclaration}
ExtendedAttributeDefinition ::=
        ExtendedAttributeItemDefinition
                ExtendedAttributeDefinitionList | λ
ExtendedAttributeItemDefinition ::=
        < AttributeName : AttributeType OtherProperties >
OtherProperties ::= DefaultValue AllowNull
DefaultValue ::=  { DEFAULT : Value } | λ
AllowNull  ::= , ALLOWNULL | λ
ExtendedAttributeDefinitionList::=
        ExtendedAttributeItemDefinition
                ExtendedAttributeDefinitionList |λ
```

شکل ۶- تعریف نوع کنش‌گر

```
WorkflowActorDeclaration  ::=
WorkflowActorItem    WorkflowActorList
WorkflowActorItem ::= <Name, Description, ActorType,
                        UserName, Password
                        ExtendedAttributeDeclarationList>
ActorType   ::= HUMAN | ROLE | ACTORSYSTEM |
                ORGANIZATIONDEPARTMENT | GROUP
ExtendedAttributeDeclarationList ::= ,
                        ExtendedAttributeDeclarationItem
                        ExtendedAttributeDeclarationList | λ
ExtendedAttributeDeclarationItem  ::= Value | NULL
WorkflowActorList ::= WorkflowActorDeclaration  | λ
```

شکل ۷- تعریف کنش‌گر

```
OrganizationChartDefinition  ::=
        ORGANIZATIONCHART {
                OrganizationDepartmentItem
                OrganizationDepartmentList } | λ
OrganizationDepartmentItem ::=
        < DepartmentName, DepartmentFatherName >
OrganizationDepartmentList  ::=
        OrganizationDepartmentItem
                OrganizationDepartmentList | λ
DepartmentFatherName        ::=
                DepartmentName | NULL
```

شکل ۸- تعریف نمودار سازمانی

```
DepartmentHasRole ::=
        DEPARTMENTROLE {
                DepartmentHasRoleItem
                DepartmentHasRoleList } | λ
DepartmentHasRoleItem ::=
        <DepartmentName , RoleName Manager>
Manager  ::= , MANAGER
DepartmentHasRoleList  ::=
        DepartmentHasRoleItem DepartmentHasRoleList | λ
```

شکل ۹- تعریف نقش‌های سازمانی

```
HumanHasRole ::=
                HUMANROLE {
                        HumanHasRoleItem
                        HumanHasRoleList } | λ
HumanHasRoleItem ::= < HumanName , RoleName >
HumanHasRoleList  ::=
                        HumanHasRoleItem
                        HumanHasRoleList | λ
```

شکل ۱۰- تعریف تناظر فرد به نقش

```
GroupHasMember  ::=
                GROUPMEMBERS {
                        GroupHasMemberItem
                        GroupHasMemberList } | λ
GroupHasMemberItem  ::=
                < GroupName , MemberItem MemberList >
MemberItem   ::= ( MemberName , MemberType )
MemberList   ::= MemberItem MemberList | λ
MemberType  ::=
                HUMAN | ROLE | ACTORSYSTEM |
                        ORGANIZATIONDEPARTMENT
GroupHasMemberList    ::=
                GroupHasMemberItem
                GroupHasMemberList | λ
```

شکل ۱۱- تعریف اعضای گروه

```
WorkflowApplicationDefinition  ::=
                WORKFLOWAPPLICATION {
                        WorkflowApplicationItem
                        WorkflowApplicationList } | λ
WorkflowApplicationItem  ::=
                < ApplicationName , ApplicationDescription ,
                ToolName , FormalParameters ,
                ExtendedAttributes >
WorkflowApplicationList ::=
                WorkflowApplicationItem
                        WorkflowApplicationList | λ
```

شکل ۱۲- تعریف نرم‌افزارهای فراخوانی‌شونده

- تعریف نرم‌افزارهای فراخوانی‌شونده

در پاره‌ای از موارد برای انجام اعمال، احتیاج به فراخوانی یک یا چند نرم‌افزار می‌باشد. ولی با توجه به این‌که برای فراخوانی، نرم‌افزارهای مذکور باید به موتور گردش کار شناسانده شوند تا وی در هنگام نیاز از آن‌ها استفاده نماید، بنابراین با استفاده از قواعد موجود در زبان که برای این کار تعبیه شده‌اند، نرم‌افزارهای مورد نظر به همراه نوع ورودی‌ها و خروجی‌ها به سیستم موتور گردش کار معرفی می‌گردند. سپس موتور گردش‌کار در موقع نیاز با ایجاد پارامترهای لازم برای ورود و خروج اطلاعات فراخوانی را بطور کامل انجام می‌دهد که قواعد مربوط به این مطلب را می‌توان در شکل ۱۲ مشاهده نمود.

- تعریف صفات خاص

در سازمان‌های متفاوت بسته به نیاز، احتیاج به تعریف صفاتی خاص است که در هنگام طراحی زبان بطور مشخص نمی‌توان آن‌ها را در نظر گرفت، ولی باید قابلیتی به زبان اضافه نمود که هر سازمان بتواند صفات خاص کاری مورد نظر خویش را تعریف نماید. بنابراین قسمتی از قواعد شکل ۱۳به این امر اختصاص داده شده است.

```
ExtendedAttributes ::=
                EXTENDEDATTRIBUTES {
                        ExtendedAttributeItem
                        ExtendedAttributeList } | λ
ExtendedAttributeItem  ::=
                < AttributeName , AttributeType ,
                  AttributeValue,AttributeDescription >
ExtendedAttributeList    ::=
                ExtendedAttributeItem ExtendedAttributeList | λ
```

شکل ۱۳- تعریف صفات خاص

- تعریف پردازه‌ها و فعالیت‌ها

مهمترین موجودیت در دستورزبان پردازه است. پردازه‌ها بیان کننده روال‌های کاری موجود در سازمان می‌باشند که با تعریف و اجرای آن‌ها می‌توان تمام اهداف سازمانی را محقق نمود. به عبارت دیگر، به ازای هر روال موجود در سازمان، پردازه‌ای در دستورزبان ایجاد می‌گردد که در تعریف آن باید اطلاعات متفاوتی مشخص شود. البته با توجه به این‌که هر پردازه از تعدادی فعالیت تشکیل شده است، لذا باید فعالیت‌ها و تمام اطلاعات مربوط به آن‌ها نیز تعریف گردد. بطور کلی فعالیت‌ها به دو دسته فعالیت‌های غیرزائد و فعالیت‌های زائد تقسیم می‌شوند. فعالیت‌های زائد فعالیت‌هایی هستند که برای اتخاذ تصمیم‌های مربوط به شیوه گردش کار از جمله ایجاد حلقه، کنترل‌گر انتقالات ورودی یا انتقالات خروجی استفاده می‌شوند و این در حالی است که فعالیت‌های غیرزائد همان کارهای واقعی می‌باشند که برای تحقق گردش‌کار توسط کنش‌گرها صورت می‌پذیرند. قواعد مربوط به این قسمت در شکل ۱۴ نشان داده شده است.

- تعریف انتقالات

برای تحقق پردازه، باید فعالیت‌ها به ترتیب تعریف‌شده‌ای اجرا گردند. در این میان بعد از تکمیل یک فعالیت، فعالیت بعدی اجرا می‌گردد. البته این‌که فعالیت بعدی کدام فعالیت است و با حصول چه شرایطی باید اجرا شود، توسط گرافی جهت‌دار مشخص می‌گردد که در این دستور زبان گراف مذکور با استفاده از قواعد تعبیه شده قابل توصیف است که این توصیفات به نام انتقالات شناخته می‌شوند. با مراجعه به شکل ۱۵ می‌توان دستورات لازم را مشاهده نمود.

- تعریف عبارت

برای بیان شروط در تعریف انتقالات باید بتوان عبارات منطقی مطلوب را تعریف نمود که در بعضی از موارد لازم است تا با استفاده از عبارت‌های حسابی، تاریخی یا رشته‌ای، عبارت‌های منطقی ترکیبی ایجاد نمود که در شکل ۱۶ این قواعد نمایش داده شده‌اند.

- تعریف داده‌های مرتبط

برای متغیرهایی که در تعریف عبارات استفاده می‌شوند و در زمان اجرا مقادیر آنان مشخص می‌گردد، از دستورات مربوط به تعریف داده مرتبط استفاده می‌شود که در این میان با ذکر نام، شرح، نوع، طول و مقدار اولیه می‌توان به تعریف این متغیرها مبادرت نمود (شکل ۱۷).

- تعریف انواع داده‌ای ساده و مرکب

با توجه به اینکه برای تعریف داده‌های مرتبط و تعریف پارامترهای توابع احتیاج به ذکر نوع آنان است. لذا قواعد موجود در شکل ۱۸ برای این امر تعبیه گردیده است. همانطور که در شکل مذکور دیده می‌شود، پنج نوع داده‌ای ساده و نیز سه نوع داده‌ای ترکیبی که شامل رکورد، آرایه و مجموعه می‌باشد در نظر گرفته شده است.

```
WorkflowProcessDefinition ::=
            WorkflowProcessItem WorkflowProcessList
WorkflowProcessItem ::=
             PROCESS{
                ProcessInformation,ProcessActivityList,
                TransitionDefinition,
                ProcessTransitionList }
ProcessInformation    ::=
              < Name , Description ,
                ProcessResponsibleName , Priority ,
                Duration (DurationUnit) , FromDate ,
                ToDate ,Documentation , Icon >
DurationUnit ::= YEAR | MONTH | DAY | HOUR | MINUTE
              | SECOND
ProcessActivityList    ::=
               ACTIVITIES { ActivityItem ActivityList }
ActivityItem   ::=
              <Name,Description,ActivityPerformerName,
               ObjectName , Duration (DurationUnit) ,
               FromDate , ToDate, Kind StartMode
               FinishMode Priority Icon Documentation
               ExtendedAttributes >
Kind        ::= Dummy | NoDummy
Dummy     ::= LoopControler | TransitionPolicy
LoopControler::= , LOOP ( LoopKind < LoopCondition > ) | λ
LoopKind          ::= REPEATUNTIL | WHILE
LoopCondition    ::= LogicalExpression
TransitionPolicy  ::= InputKind OutputKind
InputKind          ::= , INPUT ( ActionOnInputTransition )
ActionOnInputTransition  ::= ANDJOIN | XORJOIN
OutputKind  ::= , OUTPUT ( ActionOnOutputTransition )
ActionOnOutputTransition  ::= ANDSPLIT | XORSPLIT
                              TransitionNameList
TransitionNameList ::= { TransitionName
                         OtherTransitionsName } | λ
OtherTransitionName ::= , TransitionName
                          OtherTransitionsName | λ
StartMode    ::= AUTOMATIC | MANUAL
FinishMode  ::= AUTOMATIC | MANUAL
NoDummy    ::= APPLICATION ( ApplicationName ) |
                SUBFLOW( ProcesName ) | FunctionAccess
                | REPORT ( ReportName ) | λ
WorkflowProcessList  ::=
              WorkflowProcessItem WorkflowProcessList | λ
```

شکل ۱۴- تعریف پردازه و فعالیت

```
TransitionDefinition ::=
        TRANSITIONS { TransitionItem TransitionList } | λ
TransitionItem  ::=
         < TansitionName , TransitionDescription ,
           TransitionKind ExtendedAttributes >
TransitionKind ::=
            FROM < ActivityName > TO < ActivityName >
                                    TransitionCondition
          | FROMLOOP < ActivityName > TO
                                    < ActivityName >
          | FROM < ActivityName > TOLOOP
                                    < ActivityName >
TransitionCondition  ::= [ LogicalExpression ] | λ
```

شکل ۱۵- تعریف انتقالات

- تعریف توابع

با توجه به این‌که این زبان برای دامنه بزرگی از نرم‌افزارهای گردش‌کار طراحی شده است و باید بتواند قابلیت‌های متنوع و متفاوتی از قبیل چرخش انواع سند یا سایر موجودیت‌های الکترونیکی قابل چرخش، تخصیص کارها به همکاران و سایر موارد مربوط را دارا باشد. بنابراین برای ایجاد انعطاف‌پذیری در تولید اعضای دامنه مذکور، به کاربر اجازه داده شده است تا بتواند توابعی تعریف نماید و با استفاده از Apiهایی که تعبیه شده است، بتواند نیازهای خود را مرتفع نماید. از این رو کاربر می‌تواند با قواعد شکل ۱۹ به این امر مبادرت ورزد.

- تعریف پارامترها

در تعریف توابع لازم است، پارامترهای ورودی و خروجی توابع با جزییات کافی بطور مثال نوع، تعداد و مقدار پیش‌فرض مشخص شوند که با استفاده از قواعد تعبیه شده در شکل ۲۰ می‌توان تمامی پارامترهای ورودی یا خروجی توابع را بصورت جزیی تعریف نمود.

- تعریف شی

در سیستم‌های گردش‌کار اغلب موجودیتی در حال گردش است که این موجودیت می‌تواند سند، شرح یک دستور یا هر موجودیت الکترونیکی دیگری باشد. برای این‌که بتوان بطور مناسب با این پدیده برخورد کرد، عنصری به نام شی، مسئولیت تعریف موجودیت گردش‌کننده را برعهده دارد و با تعریف آن می‌توان به موتور گردش‌کار شی مورد نظر خود را شناساند. لذا با استفاده از دستورات شکل ۲۱ می‌توان عناصر مورد نظر را تعریف نمود.

```
LogicalExpression  ::=
    ( LogicalExpression BooleanOperator LogicalExpression )
    | ( StringExpression BOprator StringExpression )
    | ( DateExpression BOperator DateExpression )
    | ( ArithmaticExpression BOperator ArithmaticExpression )
    | ( SetMemberName IN SetName )
    | UnaryBooleanOprator ( LogicalExpression )
    | ( BooleanConstant )    | ( DataAccess )
ArithmaticExpression  ::=
   ( ArithmaticExpression AOperator ArithmaticExpression )
   | ( DateExpression - DateExpression )
   | - ArithmaticExpression
   | NumericConstant   | DataAccess
DateExpression  ::=
   ( DateExpression + DateExpression )
   | ArithmaticExpression + DateExpression   | DateConstant
   | DataAccess
StringExpression  ::=
   StringExpression + StringExpression   | StringConstant
   | DataAccess
DataAccess  ::=
   FunctionAccess   | RecordName.RecordMemberName
   | ArrayName [ ArrayIndex ]
FunctionAccess  ::= ( FNCACS : FunctionName )
BooleanOperator  ::= && | || | EqualityOperator
BOperator  ::= < | > | <= | >= | EqualityOperator
EqualityOperator  ::=    = = | !=
AOperator  ::= + | - | * | / | MOD | DIV | ^
BooleanConstant   ::= TRUE | FALSE
```

شکل ۱۶- تعریف عبارات، شیوه دسترسی به داده‌ها و عملگرها

- ثبت وقایع

جهت رفع مطلوب مشکلات احتمالی در هنگام بروز خطا، لازم است که اطلاعات مربوط به نام کنش‌گرها، نوع آن‌ها، زمان انجام کار، نام پردازه، نام فعالیت، تاریخ یا یکسری دیگر از صفات که به دل‌خواه کاربر از قبل تعریف شده است نگهداری گردد و برای ردیابی عوامل خطا و ساده‌تر نمودن پردازه نگهداری از آن بهره‌برداری شود (شکل ۲۲).

```
WorkflowRelevantData ::=
        WORKFLOWRELEVANTDATA {
                RelevantDataItem  RelevantDataList } | λ
RelevantDataItem ::= < RelevantDataName ,
                       RelevantDataDescription ,
                       RelevantDataType,
                       RelevantDataLength , DefaultValue >
RelevantDataType  ::= ComplexDataTypeDefinition
RelevantDataList   ::= RelevantDataItem RelevantDataList | λ
```

شکل ۱۷- تعریف داده‌های مرتبط

```
BasicDataTypeDefinition  ::=
        STRING | FLOAT | REFRENCE | DATE | BOOLEAN
ComplexDataTypeDefinition  ::=
          BasicDataTypeDefinition
          | RECORD RecordName { RecordMemberItem
                                RecordMemberList }
        | ARRAY ArrayName( BeginNumber .. EndNumber)
                                : BasicDataTypeDefinition
          | SET SetName Of { DataConstantItem
                                DataConstantList }
RecordMemberItem  ::=
                RecordMemberName : RecordMemberType
RecordMemberType  ::=  BasicDataTypeDefinition
RecordMemberList  ::=
                , RecordMemberItem RecordMemberList | λ
DataConstantItem   ::= Constant
DataConstantList   ::= DataConstantItem DataConstantList | λ
```

شکل ۱۸- تعریف انواع داده‌ای ساده و ترکیبی

```
LibraryDeclaration  ::= LIBRARY { Libraries } | λ
Libraries  ::= LibraryItem LibraryList
LibraryList  ::= , LibraryItem LibraryList | λ
LibraryItem ::= FunctionItem
FunctionItem  ::=
          FUNCTION < FunctionName , FunctionDescription ,
          ResultType , ContentFile , FormalParameters
          ,ExtendedAttributes >
ResultType  ::= BasicDataTypeDefinition
```

شکل ۱۹- تعریف توابع

```
FormalParameters ::=
                FORMALPARAMETER { Parameters } | λ
Parameters  ::= InputParameters OutputParameters
InputParameters  ::=
   INPUTPARAMETERS ( ParameterItem ParameterList ) | λ
OutputParameters  ::=
 OUTPUTPARAMETERS ( ParameterItem ParameterList ) | λ
ParameterItem  ::= RelevantDataName j ObjectName
ParameterList  ::= , ParameterItem ParameterList | λ
```

شکل ۲۰- تعریف پارامترها

```
ObjectDefinition  ::= OBJECTS { ObjectItem ObjectList } | λ
ObjectItem  ::=
       < ObjectName , ObjectDescription ExtendedAttribute >
ObjectList  ::= ObjectItem ObjectList | λ
```

شکل ۲۱- تعریف شیء

```
LoggingDefinition  ::=
     LOG { LogAttributeNameItem LogAttributeNameList } |
λ
LogAttributeNameItem  ::=
     ACTORNAME | ACTORTYPE | TIME
     | PROCESSNAME | DATE
     | ACTIVITYNAME | RELDATA <RelevantDataName>
     | OBJ<ObjectName>
     | AEAN<ActorExtendedAttributeName>
LogAttributeNameList  ::=
              , LogAttributeNameItem LogAttributeNameList |
λ
```

شکل ۲۲- تعریف Log

- گزارش‌ها

در بیشتر سازمان‌ها مدیران به وجود گزارش‌های متفاوت علاقه‌مند هستند و آنان یا دیگر همکاران سازمان با استفاده از گزارش‌ها می‌توانند عملکردها را از لحاظ کمی و کیفی ارزیابی نمایند تا برای برنامه‌ریزی آتی سازمان تصمیمات درستی اتخاذ نمایند. با استفاده از قواعد دستور زبان که در شکل ۲۳ نشان داده شده است می‌توان گزارش‌های متفاوت و متنوعی را ایجاد نمود.

```
ReportDefinition  ::= ReportItem ReportList
ReportItem  ::=
          REPORT ReportName { ReportAttributeItem
                                ReportAttributeList } | λ
ReportAttributeItem  ::=
          ACTORNAME | ACTORTYPE | TIME
          | PROCESSNAME | ACTIVITYNAME | DATE
          | RELDATA <RelevantDataName>
          | OBJ <ObjectName>
          | AEAN <ActorExtendedAttributeName>
ReportAttributeList  ::=
                , ReportAttributeIem ReportAttributeList | λ
ReportList   ::= ReportItem ReportList | λ
```

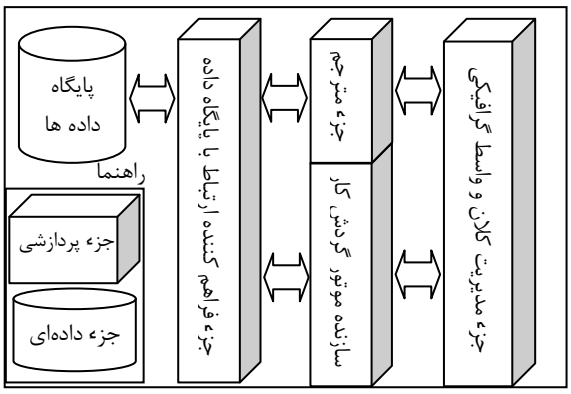
شکل ۲۳- تعریف گزارش

۶- معماری مولد موتور گردش‌کار SWEG[5]

پس از طراحی زبان برای تولید موتور گردش‌کار، باید محیطی مناسب ایجاد نمود تا با استفاده از آن مهندس کاربرد بتواند موتور دل‌خواه خود را توصیف و سپس آن‌را با استفاده از مترجم زبان، تولید نماید. به همین دلیل پیاده‌سازی محیط مورد ذکر از اهمیت ویژه‌ای برخوردار است. بنابراین معماری کلان نرم‌افزار سه لایه در نظر گرفته شده است که آن را می‌توان در شکل ۲۴ مشاهده نمود. معماری مورد نظر به دلایل متفاوتی از قبیل استقلال اجزا، قابلیت اطمینان بالا، انعطاف‌پذیری و استقلال از سیستم مدیریت پایگاه‌داده‌ها بسیار مناسب می‌باشد. به طور کلی هر یک از لایه‌ها وظایف خاصی را عهده‌دار هستندکه به شرح زیر است:

- لایه مدیریت کلان و واسط گرافیکی کاربر

یکی از اعمال مهمی که این لایه انجام می‌دهد، مدیریت کلان سایر لایه‌ها و اجزای آن‌ها است. البته وظایف دیگری را نیز بر عهده دارد که از آن جمله می‌توان به فراخوانی جزء مترجم و جزء سازنده موتور گردش کار جهت اجرا و نیز نظارت بر صحت عملکرد آن دو جزء اشاره نمود. همچنین این لایه وظیفه تعامل با کاربر، فراهم نمودن ابزارهای مدل‌سازی گرافیکی مناسب، چک کردن صحت منطق مدل‌سازی، تولید مدل متنی و ذخیره‌سازی و بارگذاری مجدد را انجام می‌دهد.

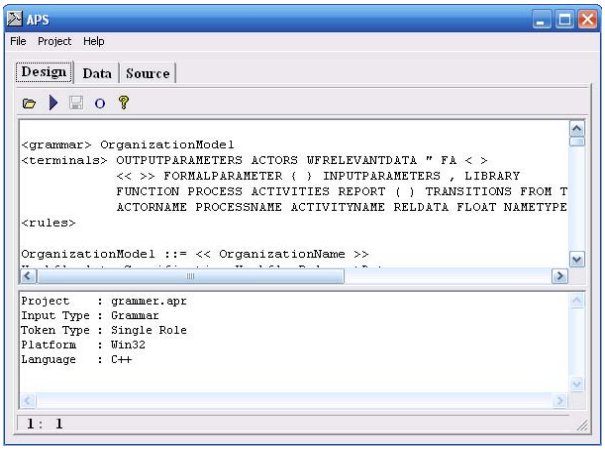


شکل ۲۴- معماری کلان نرم‌افزار تولیدکننده موتور گردش کار

- جزء مترجم از لایه دوم

این جزء یکی از مهمترین اجزاء این معماری است که وظیفه ترجمه مدل متنی را برعهده دارد و اطلاعات لازم و مفید را از مدل متنی بیرون کشیده، با استفاده از لایه سوم در پایگاه داده‌ها ذخیره می‌نماید تا در هنگام تولید موتور گردش کار از آن استفاده گردد. برای تولید مترجمی که بتواند زبان طراحی شده در بخش قبلی را ترجمه نماید، می‌توان از مترجم‌سازهایی مانند APS[۶]، PreCC، GBP یا Yacc استفاده نمود.

در این معماری برای تولید مترجم از APS استفاده شده است. این نرم‌افزار می‌تواند با استفاده از دستورزبان یا گراف نحوی زبان مورد نظر، مترجم آن را ایجاد نماید. در این میان مشخص کردن نحوه عملکرد روال‌های مفهومی بر عهده طراح زبان خواهد بود. در شکل ۲۵ نمایی از APS نشان داده شده است.



شکل ۲۵- نمایی از نرم‌افزار APS

- جزء سازنده موتور گردش کار از لایه دوم

پس از اتمام کار جزء مترجم، این بخش با استفاده از اطلاعات موجود در پایگاه داده‌ها، موتور گردش کار مناسبی که منطبق با نیازهای کاربر و مدل منطقی ترسیم‌شده وی باشد، ایجاد می‌نماید. این قسمت موتوری ایجاد می‌نماید که با موتور گردش کاری که در بخش‌های قبلی بررسی شد، سازگار باشد. بنابراین موتور تولید شده دارای اجزایی مانند جزء اجراکننده نرم‌افزارها، جزء فراهم‌کننده توابع برنامه‌نویسی، جزء ردیاب فعالیت‌ها و جزء شروع‌کننده پردازه است.

- جزء فراهم‌کننده ارتباط با پایگاه داده‌ها

این جزء تمام کارهای مربوط به ذخیره‌سازی، بروزرسانی و بازیابی اطلاعات را برعهده دارد. در این لایه توابع اساسی کار با پایگاه داده‌ها وجود دارد و هر جزء دیگر برای انجام اعمال مورد نظر خود از این اعمال استفاده می‌کند. به همین دلیل در سایر اجزا هیچ نمونه‌ای از توابعی که مربوط به سیستم مدیریت پایگاه داده‌های خاصی باشد، دیده نمی‌شود.

۷- بررسی مثالی از گردش کار

در این مثال شرکتی مورد بررسی قرار می‌گیرد که الگوهای رفتاری And-join و And-split در آن استفاده شده است. در این شرکت تجاری پنج نفر کار می‌کنند که نقش‌های مدیرعامل، مدیرمالی و کارشناس را بر عهده دارند. از آن پنج نفر، یک نفر مدیر عامل، یک نفر مدیرمالی و سه نفرکارشناس هستند. در ابتدا سازمان مذکور توسط دستورزبان توصیف می‌شود که این دستورات در شکل ۲۶ قابل مشاهده هستند. فرض کنید در این شرکت پردازه تجاری زیر وجود دارد.

```
< Corporation >
ACTORS {
   < DirectManager, "Main Role Of Corporation", ROLE,
     DirectMan, ManPass >
   < FinancialManager, "Financial Manager", ROLE,
     FinanMan, FiPass >
   < FirstEngineer, "Computer Engineer ", ROLE, FEng,
     Eng1Pass >
   < SecondEngineer, "Computer Engineer ", ROLE, SEng,
     Eng2Pass >
   < ThirdEngineer, "Computer Engineer ", ROLE, TEng,
     Eng3Pass >
   < EngineerGroup, "Group Of Engineers ", GROUP, -, - >
   < Word, "Software For Creating Document ",
     ACTORSYSTEM, None, None >
}
GROUPMEMBERS {
   <EngineerGroup, (FEng, ROLE)>  <EngineerGroup,
    (SEng, ROLE)>  <EngineerGroup, (TEng, ROLE)>
 }
WORKFLOWAPPLICATION {
  <  MsWord,  "MicrosoftWord  ",  msword.exe,
FORMALPARAMETER { OUTPUTPARAMETER (Path)} >
 }
WORKFLOWRELEVANTDATA {
   < Result, "Result", BOOLEAN, 1, TRUE >
   < Path, "Path Of Request File", STRING, 500, "" >
   < AppName, "Word Application File Name", STRING, 10,
     "MsWord.exe" >
 }
LIBRARY {
  FUNCTION < GivingIdea, "This Function Show Request
          And Giving Result And Set Result Relevant Data",
          BOOLEAN, "func.dll",
          FORMALPARAMETER {
                INPUTPARAMETERS ( AppName , Path )
                OUTPUTPARAMETER (Result)
          }
}
```

شکل ۲۶- تعریف مدل سازمان

۷-۱- پردازه درخواست مساعده

برای این‌که کارشناس مساعده دریافت کند، ابتدا باید فرم مربوط به تقاضای مساعده را پرنماید. سپس فرم مذکور را به مدیرعامل و مدیرمالی ارسال کند. مدیرعامل و مدیرمالی درخواست مذکور را بررسی می‌نمایند و نظر خود را نیز اعلام

می‌کنند. در صورتی‌که هر دو موافق باشند، مساعده مذکور پرداخت خواهد شد ولی اگر یکی از آن‌ها با درخواست مخالفت نماید، به کارشناس هیچ مبلغی پرداخت نخواهد شد. همچنین موافقت یا عدم موافقت نیز به اطلاع وی خواهد رسید. در این پردازه دو الگوی And-split و And-join مشاهده می‌شود که دستورات لازم برای تعریف پردازه مورد نظر در شکل ۲۷ ذکر گردیده است.

```
PROCESS {
        < FinancialHelp, "Financial Help Description",
          DirectMan >
        ACTIVITIES {
        < FillingRequest, "None", EngineerGroup,
          APPLICATION(MsWord, Path), MANUAL,
          MANUAL >
        < FinMIdea, "Getting Financial Manager Idea",
          Financial Manager, FUNCTION ( GivingIdea ),
          MANUAL, MANUAL >
        < DirMIdea, "Getting Direct Manager Idea",
          Direct Manager, FUNCTION ( GivingIdea ),
          MANUAL, MANUAL >
        < NegativeResult, "Your Request Not Accepted",
          EngineerGroup, MANUAL, MANUAL>
        < ToFMDM, "Sendig To F. and D. Managers",
          OUTPUT ( ANDSPLIT ) >
        < FromFMDM, "Both Managers Accepted", INPUT
          ( ANDJOIN ) >
        < PositiveResult, "Your Request Accepted",
          EngineerGroup,  MANUAL, MANUAL>
        }
        TRANSITIONS {
        < SendToManagers0, "Sending to DommyActivity",
          FROM < FillingRequest > TO < ToFMDM > >
        < SendingToManager1, " Sending From Dummy
          Activity To Direct Manager for Giving
          Idea",FROM < ToFMDM > TO < DirMIdea >>
        < SendingToManager2, " Sending From Dummy
          Activity To Financial Manager for Giving
          Idea",FROM < ToFMDM > TO < FinMIdea >>
        < IdeaResult0, "Direct Managers Not Accepted",
          FROM < DirMIdea > TO < NegativeResult >
          [ Result == FALSE ] >
        < IdeaResult1,"FinancialManagers Not
          Accepted",FROM<FinMIdea>
          TO < NegativeResult > [ Result == FALSE ] >
        < IdeaResult2, "Direct Managers Accepted",
          FROM < DirMIdea >
          TO < FromFMDM > [ Result == TRUE ] >
        < IdeaResult3, "Financial Managers Accepted",
          FROM < FinMIdea >
          TO < FromFMDM > [ Result == TRUE ] >
        < PosResult,"Sending Positive Answer ",
          From<FromFMDM >   TO < PositiveResult > >
}
```

شکل ۲۷- تعریف پردازه درخواست مساعده

۸- نتیجه‌گیری

با توجه به گسترش روزافزون استفاده از سامانه‌های گردش کار در سازمان‌ها، جهت ارتقاء سطح کمی و کیفی در کارها، در این مقاله به بررسی روش تولید نرم‌افزار مبتنی بر خانواده، موتور گردش کار استاندارد و استفاده از روش مذکور جهت ایجاد موتورهای گردش کار پرداخته شد و در نهایت معماری کلان سامانه‌ای که بتواند موتور گردش کار دل‌خواه را ایجاد نماید مطرح گردید. ولی در این میان پیاده‌سازی کامل سامانه، مبتنی بر وب بودن آن، ایجاد مدل‌های مبتنی بر XML برای ایجاد قابلیت حمل و تولید موتورهای گردش کار توزیع شده، ایجاد عامل‌ها در انجام فعالیت‌ها [۱۶] از موارد مهمی هستند که باید روی آن کار شود.

مراجع

[1] K. C. Kang, V. Sugumaran, and S. Park, *Applied software product-line engineering*, Auerbach Publications, 2009.

[2] C. Kim, H. S. Chung, E. S. Cho, *Micro and macro workflow variability design techniques of component*, Information and Software Technology, In Press, Corrected Proof, Available online 31 January 2007.

[3] Biglever Software Inc., *Software Product Lines*, http://www.softwareproductlines.com, 2004.

[4] D. M. Weiss, and C. R. Lai, *Software Product-Line Engineering: A Family-Based Software Developement Process*, Addison Wesley Professional, 1999.

[5] M. Harsu, *FAST product-line architecture process*, Software Systems Laboratory, Tampere University of Technology, Technical Reports, pp.45, 2001.

[6] M. Wang, H. Wang, and D. Xu, "The design of intelligent workflow monitoring with agent technology," *Knowledge-Based Systems*, vol. 18, no. 6, pp. 257-266, 2005.

[7] W. Tan and Y. Fan, "Dynamic workflow model fragmentation for distributed execution," *Computers in Industry*, vol. 58, no. 5, pp. 381-391, 2007.

[8] J. Jung, H. Kim, and S. Kang, "Standards-based approaches to B2B workflow integration," *Computers and Industrial Engineering*, vol. 51, no. 2, pp. 321-334, 2006.

[9] M. D. R-Moreno, D. Borrajo, A. Cesta, and A. Oddi, "Integrating planning and scheduling in workflow domains, Expert Systems with Applications," vol. 33, no. 2, pp. 389-406, 2007.

[10] D. Hollingsworth, *The Workflow Reference Model - Issue 1.1*, Technical Report Document Number TC00-1003, Workflow Management Coalition, 1995.

[11] Workflow Management Coalition, *Workflow Management Coalition Terminology & Glossary*, Technical Report Document Number WFMC-TC-1011, Workflow Management Coalition, 1999.

[12] S. Li and D. Coleman, "Modeling distributed GIS data production workflow, Computers," *Environment and Urban Systems*, vol. 29, no. 4, pp. 401-424, 2005.

[13] R. A. Botha and J. H. P. Ellof, "A security interpretation of the workflow reference model," *Information Security - from Small Systems to Management of Secure Infrastructures*, pp. 43–51, 1998.

[14] D. G. Cholewka, R. A. Botha, and J. H. P. Ellof, "A Context-sensitive Access Control Model and Prototype

implementation," *Proc. of the IFIP 15th AnnualWorking Conference on Information Security for Global Information Infrastructures*, pp. 341-350, 2000.

[15] J. Wainer, A. Kumar, and P. Barthelmess, "DW-RBAC: A formal security model of delegation and revocation in workflow systems," *Information Systems*, vol. 32, no. 3, pp. 365-384, 2007.

[16] P. Senkul and I. H. Toroslu, "An architecture for workflow scheduling under resource allocation constraints," *Information Systems*, vol. 30, no. 5, pp. 399-422, 2005.

[17] W. M. P. van der Aalst and A. H. M. Ter Hofstede, "YAWL: Yet Another Workflow Language," *Information Systems*, vol. 30, no. 4, pp. 245-275, 2005.

[18] R. W. Weidel, *Workflow White Paper*, SoftBrands Manufacturing, 2002.

[19] C. Ouyang, M. Adams, and A. H. M. Ter Hofstede, "Yet Another Workflow Language: Concepts, Tool Support, and Application," *Handbook of Research on Business Process Modeling*, pp. 92-121, 2009.

[20] H. Lee and H. Suh, "Workflow structuring and reengineering method for design process," *Computers and Industrial Engineering*, vol. 51, no. 4, pp. 698-714, 2006.

مرتضی یوسف صنعتی مدرک کارشناسی و کارشناسی‌ارشد را از دانشگاه صنعتی شریف به ترتیب در سال‌های ۸۱ و ۸۳ در رشته مهندسی نرم‌افزار اخذ نمود. سپس از سال ۸۳ در دانشگاه بوعلی سینا در گروه کامپیوتر مشغول به تدریس گردید. همچنین ایشان از سال ۸۹ در مقطع دکترا در دانشگاه مک مستر در کشور کانادا مشغول به تحصیل است و زمینه تحقیقاتی ایشان مدیریت جریان گردش کار در پزشکی می‌باشد.

آدرس پست‌الکترونیکی ایشان عبارت است از:

mysanati@basu.ac.ir

سید حسن میریان حسین‌آبادی مدرک کارشناسی را از دانشگاه شهید بهشتی در سال ۶۳ و مدرک کارشناسی‌ارشد را از دانشگاه صنعتی شریف در سال ۶۶ در رشته مهندسی نرم‌افزار اخذ نموده است. ایشان در سال ۷۱ در مقطع دکترا در دانشگاه اسکس انگلستان ادامه تحصیل داده و در سال ۷۶ پس از فراغت از تحصیل با اخذ مدرک دکترا در علوم کامپیوتر (روش‌های صوری) به عنوان استادیار در دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف مشغول به کار شده‌اند. زمینه‌های تحقیقاتی ایشان کاربرد روش‌های صوری در توصیف و تولید نرم‌افزار به ویژه با استفاده از تئوری انواع و ریاضیات ساختی، معیارها و اندازه‌گیری نرم‌افزار، معماری نرم‌افزارهای با قابلیت پیکربندی مجدد، توصیف و درستی‌یابی صوری معماری نرم‌افزار و پایگاه داده رابطه‌ای و اکس- ام- ال است.

آدرس پست‌الکترونیکی ایشان عبارت است از:

hmirian@sharif.edu

اطلاعات بررسی مقاله:

تاریخ ارسال: ۸۸/۳/۲۵

تاریخ اصلاح: ۹۰/۵/۳

تاریخ قبول شدن: ۹۰/۵/۱۶

نویسنده مرتبط: دکتر سیدحسن میریان حسین‌آبادی، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران.

[1] Product Line Engineering
[2] Family Based Approach
[3] Rational Unified Process
[4] Structural System Analysis and Design Method
[5] Sharif Workflow Engine Generator
[6] Advanced Parsing System