مدیریت دانش مورد پی[1] در فرایند توسعه نرم افزار

امیر خانلری
دانشکده مدیریت دانشگاه تهران
AmirKh@SystemGroup.Net

واژه‌های کلیدی

مدیریت دانش، چرخه توسعه نرم افزار، استدلال مورد پی، کارخانه تجربه

چکیده

شرکت های نرم افزاری کاملاً متکی به سرمایه های فکری و دانش سازمانی خود می باشند، بنابراین می بایست چنین سرمایه ای به طور منظم برنامه ریزی شده و به شکل مناسبی هدایت گردد. جهت هدایت و مدیریت چنین دانش و سرمایه ای روش های تکنیکی متفاوتی وجود دارد، یکی از روش هایی که در توسعه سیستم های خبره دانش پی مورد استفاده قرار می گیرد، " استدلال مورد پی " است. این روش با مدل های استفاده مجدد دانش نرم افزاری تطابق دارد و می تواند جهت مدیریت دانش چرخه توسعه نرم افزار مناسب باشد. در این مقاله با توجه به خصوصیات این روش ، مدلی جهت مدیریت دانش سازمانی متناسب با فرایندهای تولید و توسعه نرم افزار ارائه می گردد.

[1] Case-Based

مقدمه

دارایی های عمده شرکت های نرم افزاری، کارخانه ها، ساختمانها یا ماشین آلات گران قیمت نیست، بلکه دارایی اصلی یک سازمان نرم افزاری مانند بخش هایی چون مشاوره، حقوق، بانکداری، تبلیغات و.. سرمایه فکری آن است. مشکل اساسی چنین سرمایه ای نرخ بالای ورود و خروج آن است که با نرخ مشابهی افراد بی تجربه وارد سازمان شده و تجارب سازمانی خارج می شود، بنابراین چنین شرکتهایی با چالش حفظ سطح توانمندی مورد نیاز برای غلبه بر رقبایشان مواجه هستند.

[15] از طرفی زمینه فعالیت نرم افزار و نیازهای کاربران به سرعت در حال تغییر است که این امر باعث می شود تا دانش توسعه نرم افزار نیز پویا و متغیر باشد. [10]

توسعه نرم افزار شامل بسیاری از افراد درگیر در مراحل و فعالیتهای مختلف است و نه تنها منابع حاضر همگام با نیازها افزایش نمی یابند، بلکه روزانه از میزان تجربه و شایستگی آنها با خروج از سازمان کاسته می شود. بنابراین استفاده بهتر از این دانش، انگیزه و محرک اصلی مدیریت دانش در مهندسی نرم افزار است که تجزیه و تحلیل عمیق تری را می طلبد. [15]

مطابق نتایج یک تحقیق، افراد در گیر در فرایند توسعه نرم افزار بیش از40% زمانشان زا برای دستیابی ، مستند سازی اطلاعات و استفاده از منابع صرف می کنند. این یافته ها نشان می دهد که توسعه نرم افزار فعالیتی وابسته به دانش است و پیچیدگی این دانش فراتر از ظرفیت افراد است تا بتوانند به تنهایی مشکلات خویش را حل کنند. از آنجایی که دانش چنین نقش حیاتی در توسعه نرم افزار ایفا می کند، ابزارهایی جهت انتشار آن مورد نیاز می باشد.[8]

دانش در شرکتهای نرم افزاری کاملاً وابسته به نوع سازمان، ابزارهای مورد استفاده، سطوح دانش افراد، فرهنگ سازمانی و دیگر اشکال تجربه پی دانش است. بیشترین کار یک توسعه دهنده نرم افزار ایجاد سیستم های کاربردی از طریق مشخصات ناقص و نیازمندی های نوسانی مشتری است و ماهیت پویای دانش در این سازمانها به گونه ای است که مفروضات سنتی دانش در این محیط کارایی ندارد. از آنجایی که دانش در چنین محیط هایی به صورت پویا و وابسته به موقعیت خاص است، اتخاذ روش پشتیبان تصمیم مورد پی می تواند برای کنترل و هدایت دانش مناسب باشد. [8-19]

۱. "استدلال مورد پی"

پارادایم عمومی "استدلال مورد پی" فرض می کند که بیشتر مسائل انسانی از طریق به کارگیری تجربیات گذشته و قیاس با موقعیت حاضر حل می شود. در واقع کنش های ما بیشتر بر پایه تجربیات گذشته است تا بتوانیم شباهتها و تفاوتهای میان آنها را بیابیم.در این صورت می توان کنش های موفق را تکرارنمود، کنش های ناقص را اصلاح کرد و از کنش های ناموفق دوری گزید.[14-11]

"استدلال مورد پی"در موقعیت هایی که هیچ گونه راه حل رسمی و الگوریتمی وجود ندارد و تنها بعضی از مثالها از گذشته موجود می باشد، کاربران را پشتیبانی می کند. پشتیبان تصمیم مورد پی وظیفه تشخیص شباهتها میان موارد و ارزیابی قابلیت کاربرد کنش های پیشنهادی در مساله پیش روی را بر عهده دارد.[10]

"استدلال مورد پی" یک پارادایم حل مشکل است که از نظر بسیاری از جنبه ها با روش های دیگر هوش مصنوعی متفاوت است. این روش به جای تکیه صرف بر دانش عمومی یک دامنه، یا ایجاد ارتباط میان توصیفگرهای مساله و نتایج، قادر است تا از دانش خاصی که قبلاً در موقعیت های واقعی تجربه شده است، بهره برداری نماید. مساله جدید با

یافتن یک یا چند مورد مشابه گذشته حل می شود و از آن در موقعیت جدید مساله استفاده مجدد به عمل می آید. تفاوت مهم دیگر این است که "استدلال مورد پی" یک روش یادگیری تدریجی و پایدار است زیرا هر زمان مساله ای حل می شود و تجربه جدیدی بدست می آید، نگهداری شده و برای حل مسائل آینده آماده می گردد. باید در نظر داشت که واژه حل مساله در این روش به طور گسترده تری به کار می رود که این با زمینه سیستم های دانش پی سازگاری و ارتباط دارد، بدان معنی که حل مساله لزوماً یافتن حل واقعی یک مساله کاربردی نیست، بلکه می تواند هر موردی باشد که کاربر قبل از مواجهه در آن قرار گرفته باشد و به طور مثال برای تصدیق یا نقد یک موقعیت پیشنهاد شده توسط کاربر، تفسیر موقعیت مساله، ایجاد مجموعه ای از راه حل های ممکن یا ایجاد انتظارات در داده قابل مشاهده مورد استفاده واقع می شود.

اولین سیستمی که می تواند به عنوان "استدلالگر مورد پی" نامیده شود سیستمCYRUS است که توسط ژانت کلدنر در دانشگاه Yale توسعه داده شد. بعداً مدل حافظه موردی این سیستم به عنوان اساسی جهت استفاده در دیگر سیستم های "استدلال مورد پی"به کار گرفته شد مثل سیستم هایMediator ، Persuader ، Chef ، Julla وCasey . همچنین اصول و مدل های دیگری از "استدلال مورد پی" توسط بروس پورتر و گروهش در دانشگاه تگزاس توسعه داده شد که منجر به توسعه سیستم Protos شد و نیز افراد دیگری مثل ادوینا ریسلند و گروهش در دانشگاه ماساچوست و ... .[1]

"استدلال مورد پی" در زمینه مهندسی نرم افزار کاربردهای مختلفی دارد که این کاربردها را می توان به دو طبقه اصلی تقسیم بندی نمود: استفاده در مسایل پیش بینی و استفاده مجدد از دانش در حل مسائل[16] که در این مقاله تمرکز بر روی کاربرد آن در زمینه استفاده مجدد از دانش سازمانی می باشد.

۲. استفاده مجدد در مهندسی نرم افزار

یک شیوه یرای کاهش هزینه توسعه نرم افزار استفاده مجدد بخش هایی از نرم افزارهای تولید شده است. در اثر این کاربرد، علاوه بر این که زمان و هزینه توسعه نرم افزار کاهش می یابد، کیفیت محصولات نیز افزایش می یابد، زیرا از اجزایی استفاده می شود که قبلاً تست کیفیت و قابلیت اطمینان را گذراتده اند.[13] مفهوم استفاده مجدد در مهندسی نرم افزار، منبعی مهم و بالقوه برای بهبود بهره وری سازمانی به شمار می رود. این استفاده فراتر از محصول نرم افزاری و کدهای آن است و تقریباً همه مصنوعات مرتبط با توسعه نرم افزار شامل برنامه پروژه، برنامه تست و همچنین علائم، الگوها، مشخصات و فرایندها و تجارب پروژه نرم افزاری را در بر می گیرد. [16] به طور کلی اجزائی که می توانند به طور موثری مجدداً استفاده شوند عبارتند از:

- مشخصات نیازمندی ها
- طراحی
- کد
- موارد تست
- دانش

دانش انتزاعی ترین محصول توسعه نرم افزاری است و هر چند استفاده مجدد از دانش به طور خودکار و بدون تلاش آگاهانه در فرایند توسعه اتفاق می افتد ولی به یادآوری جزئیات دانش قابل استفاده توسعه مشکل می باشد و

استفاده مجدد دانش به صورت برنامه ریزی شده می تواند اثربخشی آن را افزایش دهد، به همین منظور، دانش قابل استفاده باید به طور منظم بازیابی ، مستند سازی و بازنمایی گردد.[13]

مشکل اساسی در استفاده مجدد از دانش نرم افزاری این است که به علت تفاوت بین پروژه های مختلف نرم افزاری تطبیق دقیق بین موارد بسیار مشکل است، بنابر این تنها بازیابی اجزای مشابه مطرح می باشد. کار اولیه در این زمینه توسط مایدن و ساتکلایف انجام شد که فنون استدلال قیاسی را به عنوان پشتیبانی جهت استفاده مجدد از مشخصات نرم افزاری به کار بردند و دامنه را با استفاده از چهار بعد متفاوت (معنایی، ساختاری، عملی و انتزاعی) مقایسه نمودند. همچنین از "استدلال مورد پی" در استفاده مجدد مجموعه توابع برنامه نویسی زبانهایC و Ada با استفاده از معیارهای فاصله ای بر اساس ترکیبی از شبکه های معنایی به کار برده شده است. [16]

۳. استفاده مجدد و "استدلال مورد پی"

بهترین شکل استفاده مجدد از مهندسی نرم افزار که توسط "استدلال مورد پی" پشتیبانی شده است استفاده مجدد از تجربه است. به عبارت دیگر برای یادگیری موثر از پروژه های سابق نرم افزاری و برای اینکه درسهای فراگرفته شده بطور گسترده در اختیار کل سازمان قرار گیرد می توان از این روش استفاده نمود.

مهمترین انگیزه استفاده مجدد از تجربه در حیطه مهندسی نرم افزار از ایده های باسیلی در مورد کارخانه تجربه[۲] ناشی می شود.[6] این ایده ها کاملاً همراستا با تکنولوژی "استدلال مورد پی"است به طوری که بسیاری از محققان یادگیری سازمانی را به عنوان کاربرد طبیعی این نوع استدلال[20-9-7-2] و برتری روش "استدلال مورد پی" بر دیگر روش ها را ترکیب حل مساله با یادگیری پایداری که از تجربه حل مساله بدست می آید، دانسته اند. [3]علاوه بر این پارادایم بهبود کیفیت QIP/EF چارچوبی برای یادگیری مستمر در مورد فعالیتهای مهندسی نرم افزار و روش های آن ارائه می دهد که این پارادایم زیر ساخت سازمانی مناسبی برای اجرای سیستم های "استدلال مورد پی"در محیط های صنعتی است و برای استفاده مجدد از چرخه حیات، فرایندها و محصولات نرم افزاری نیز بسیار مورد ارجاع قرار می گیرد. در واقع جمع آوری و مدیریت تجارب توسعه نرم افزار نقش کلیدی در بهبود کیفیت و فرایند نرم افزار ایفا می کند. [6]

جهت استفاده مجدد از دانشی که در فرایند مذکور شرح گردید، موارد زیر باید قبل از شروع به کار برنامه استفاده مجدد به روشنی مشخص گردد:[13]

- ایجاد جزء[۳] : اجزاء قابل استفاده مجدد باید شناسایی شوند. انتخاب نوع صحیح اجزائی که قابلیت استفاده مجدد دارند بسیار مهم است. در این مورد تحلیل دامنه[۴] می تواند مورد استفاده قرار گیرد.
- شاخص بندی و ذخیره سازی جزء: اجزاء باید طبقه بندی شوند تا به آسانی قابل جستجو باشند، بنابراین باید در یک سیستم پایگاه داده رابطه ای[۵] یا سیستم پایگاه داده شیء گرا[۶] ذخیره سازی شوند.

---

[2] Experience Factory
[3] Component
[4] Domain Analysis

- جستجوی جزء: برای اینکه جستجو به طور کارایی صورت گیرد، توسعه دهندگان نیازمند روش های جستجوی صحیح هستند.
- فهم و درک جزء: توسعه دهندگان برای تشخیص قابل استفاده بودن یک جزء باید درک دقیقی از آنچه جزء انجام می دهد داشته باشند.
- تطبیق جزء: جزء انتخاب شده ممکن است دقیقاً با مشکل مورد نظر منطبق نباشد، لذا قبل از استفاده مجدد باید با شرایط موجود تطبیق داده شود.
- نگهداری انباره: اجزاء جدید باید وارد انباره شوند، اجزاء ناقص پیگیری شوند، کاربردهای جدید جایگزین کاربردهای منسوخ شوند و اجزاء منسوخ از انباره حذف گردد.

برای استفاده از روش استدلال مورد پی برای مدیریت تجارب چرخه توسعه نرم افزار می بایست چرخه "استدلال مورد پی"عمومی را با فرایند های موجود تطبیق داده و هریک از گامهای آن را با توجه به شرایط به کار برد، گامهای این چرخه عبارتند از:[1]

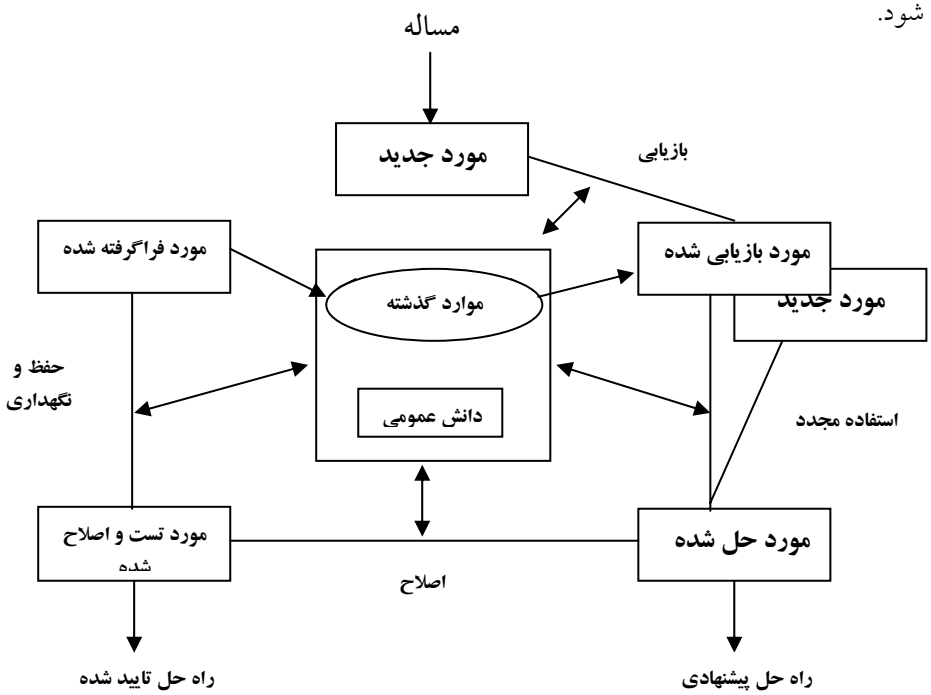
1.بازیابی بیشترین مورد یا موارد مشابه (Retrieve)

2.استفاده مجدد اطلاعات و دانش برای حل مساله (Reuse)

3.اصلاح راه حل پیشنهادی (Revise)

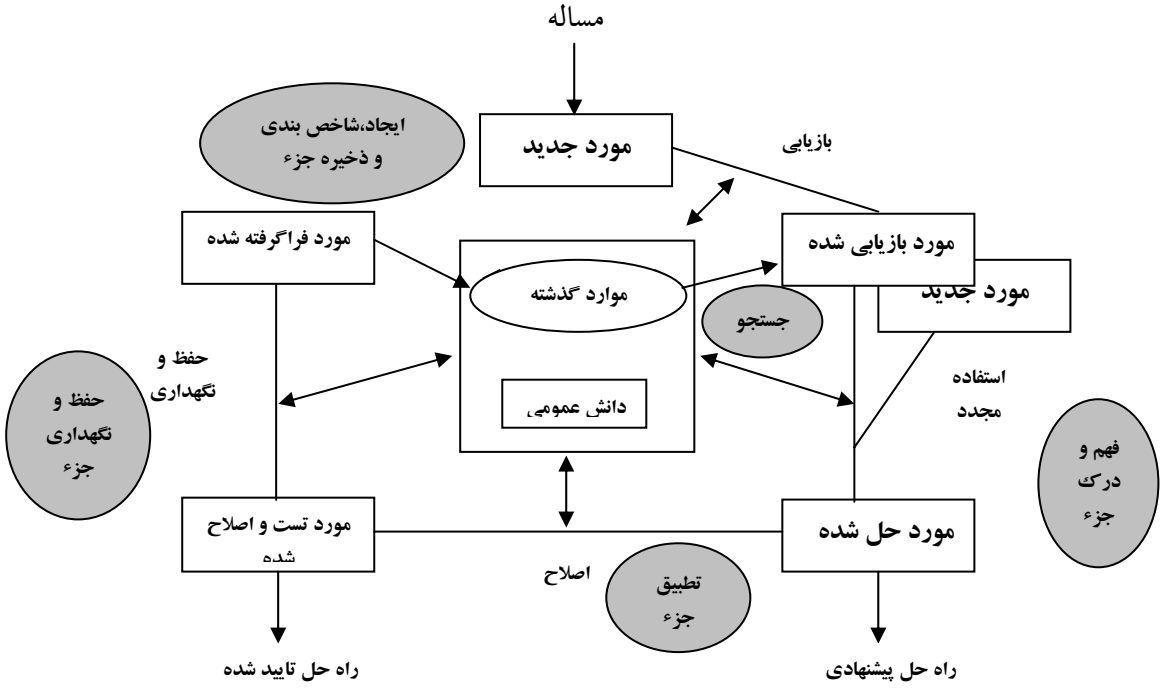
4.حفظ و نگهداری جزئیات تجربه جدید برای استفاده احتمالی در حل مساله (Retain)

در اسن چرخه (شکل شماره 1) مساله جدید با بازیابی یک یا بیشتر از موارد گذشته حل می شود، این موارد مجدداً استفاده شده و راه حلها بر اساس استفاده های جدید اصلاح گردیده و در پایگاه دانش(پایگاه مورد[۷]) موجود ذخیره می شود.



شکل شماره (۱)- چرخه استدلال مورد پی

با اندکی تامل در این چرخه متوجه می شویم که مراحل و گامهای آنها شباهت زیادی به یکدیگر دارند. شاید یکی از دلایلی که تحقیقات گذشته، چرخه"استدلال مورد پی" را مناسب ترین را ه استفاده مجدد از دانش مهندسی نرم افزار دانسته اند، انطباق گامهای استفاده مجدد و چرخه "استدلال مورد پی"باشد. در این دو چرخه فرایند بازیابی چرخه "استدلال مورد پی"با جستجوی جزء فرایند استفاده مجدد، استفاده مجدد با فهم و درک جزء، اصلاح با تطبیق جزء و حفظ و نگهداری با نگهداری جزء که با ایجاد، شاخص بندی و ذخیره جزء انجام می گیرد، تطابق دارند. بنابر این با توجه به مراحل استفاده مجدد از دانش مهندسی نرم افزار می توان چرخه "استدلال مورد پی"را طبق شکل(2) در نظر گرفت.



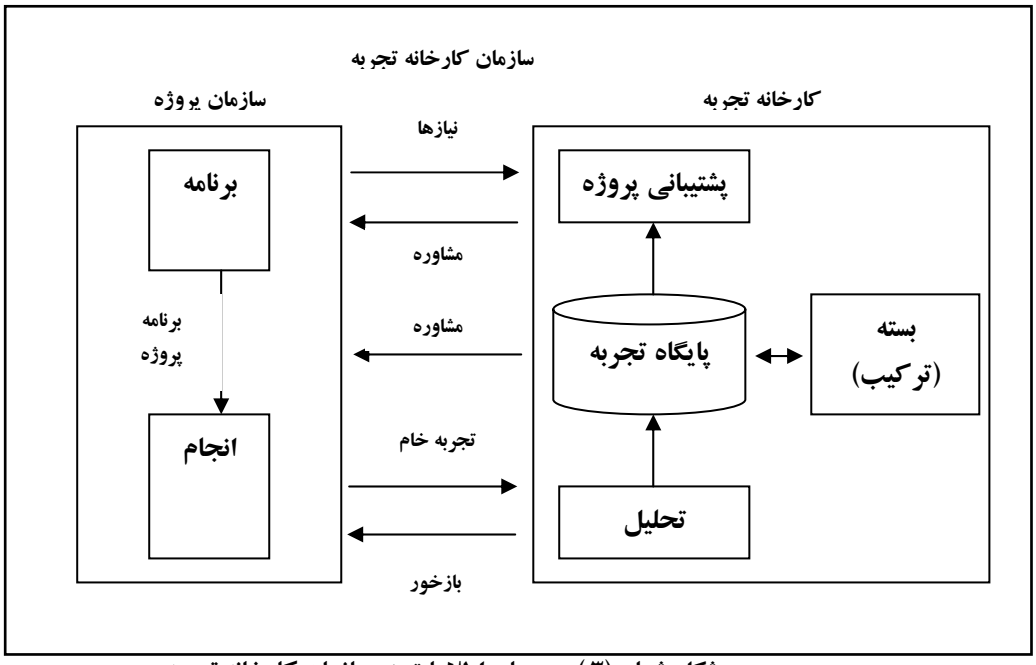
شکل شماره(۲)- چرخه استدلال مورد پی و استفاده مجدد دانش توسعه نرم افزار

۴. کارخانه تجربه

پارادایم کارخانه تجربه روش سازمانی برای کسب تجربیات توسعه از پروژه های خط تولید نرم افزار و عرضه تجربیات به پروژه های آینده است. این پارادایم متکی به جمع آوری تجربیات و ذخیره آنها در یک پایگاه تجربه است. به طور کلی شش دسته تجربه نرم افزاری شناسایی شده است که عبارتند از: محصول، فرایند، رابطه، ابزار، مدیریت و داده.[5]

روش های مختلفی برای مستند سازی این دسته تجربیات معرفی شده است که این روش ها شامل گزارشات غیر رسمی، متن ساختار یافته و زبانهای رسمی می باشد. محتویات این دسته تجربیات از تجربیات در دست تا درسهای فراگرفته شده از قبل متفاوت است. [18] برای پایگاه های تجربه بزرگ تکنولوژی "استدلال مورد پی"پیشنهاد شده است که برای مستند سازی و انتشار بسته های تجربه استفاده می شود. [17] کارخانه تجربه درسهای فراگرفته شده، داده های پروژه و گزارشات تکنولوژیکی را تحلیل و ترکیب نموده و آنها را در انباره تجربه ذخیره می کند. کارخانه تجربه روش های مختلفی را مثل معیارهای طراحی فرایندهای نرم افزاری و ویژگی های محصول نرم افزاری برای بسته بندی تجربه به کار می گیرد و سپس با استفاده از ویژگی هایی که رفتارشان را در زمینه های مختلف توصیف می کند مدل هایی ارائه می دهد. داده های این مدل

ها از پروژه های توسعه و توسط افراد، مستندات و پشتیبانی خودکار فراهم می آید.فعالیت های کارخانه تجربه و سازمان پروژه ای می بایست با یکدیگر یکپارچه شوند به طوری که فعالیت های این کارخانه باید تجربه های استخراج شده را به پروژه ها ارائه دهد و پروژه ها نیز تجارب جدید و نیازهای خود را در اختیار آن قرار دهند.[17] شکل شماره (3) این تعامل و مبادله تجربه را ارائه می دهد.



شکل شماره(۳) – جریان اطلاعات در سازمان کارخانه تجربه

در سیستم هایی مانندBORE که مدیریت دانش توسعه نرم افزار را با استفاده از تکنولوژی "استدلال مورد پی"انجام می دهند، دو نوع اطلاعات عمده وجود دارد:

- موارد تجربه
- توصیفگرهای منابع موجود

موارد تجربه که شامل درسهای فراگرفته شده از قبل است حاوی اطلاعات زیر می باشد:

1.نام پروژه
2.فرایند تجربه شده
3.نوع تجربه (موفق یا ناموفق)
4.مساله (توصیفی از مساله مورد نظر)
5.راه حل
6.زمینه (موقعیتی که مساله در آن رخ داده است)

همچنین توصیفگرهای منابع نیز شامل اطلاعاتی در مورد منبع خاص مورد استفاده نرم افزار است که موارد زیر را در بر می گیرد و اصولاً از نیاز به ارزیابی تناسب ابزارها برای یک مساله خاص ، نشات می گیرد.

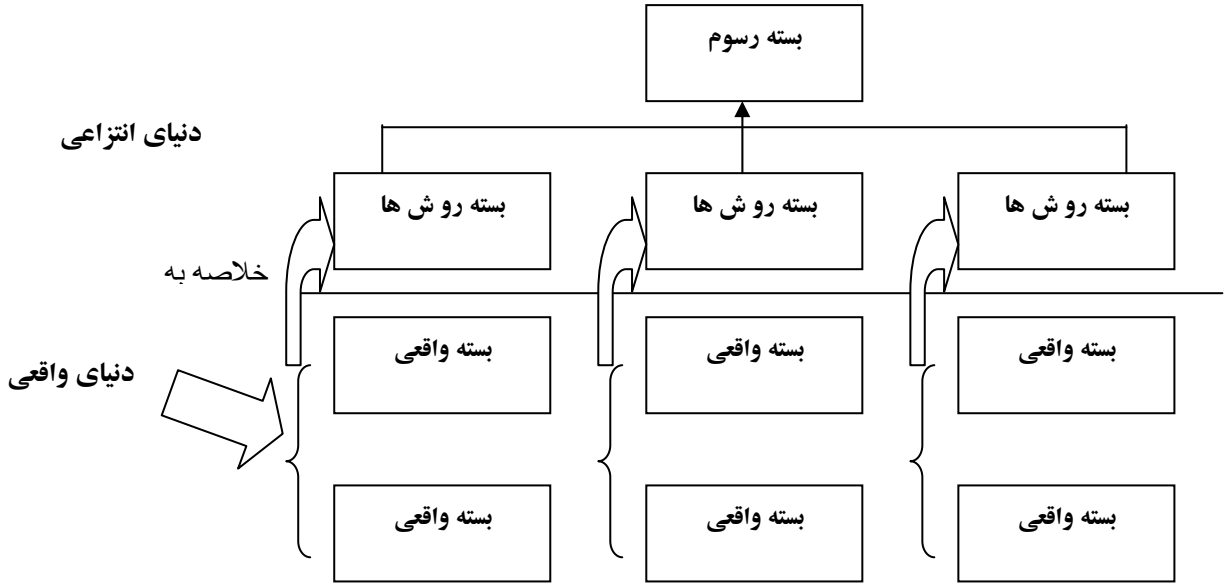
- ابزارها: شامل ابزارهای CASE ، سازنده هایGUI ، کامپایلرها، ابزارهای پشتیبانی پروژه و دیگر برنامه های کاربردی که در مورد چگونگی استفاده از این ابزارها شرح داده می شود.
- پروژه ها: شامل اطلاعاتی در مورد پروژه های توسعه در سازمان است.
- افراد: اطلاعات در مورد توسعه دهندگان نرم افزار در سازمان که شامل اطلاعات تماس و مواردی که شخص در حافظه سازمان به ثبت رسانیده است، می باشد.
- روش های توسعه: روش هایی مثل روش آبشاری، روش توسعه ساختار یافته، روش توسعه سریع و... و چگونگی استفاده و زمان بکارگیری آنها.[10]

دانش توسعه نرم افزار به دو دسته تقسیم می شود: دانش واقعی و دانش انتزاعی. دانش واقعی از تجربیات در دست بدست می آید و دانش انتزاعی فرایند تصمیم گیری یک پروژه را با ارائه راه حلهای بسته بندی شده پشتیبانی می کند. دانش انتزاعی ممکن است از دانش داخل سازمان، نتایج تجربی یا صنعت نرم افزار کسب شود که باید با استفاده از بسته های واقعی مرتبط اصلاح گردد و علیرغم این که دانش واقعی می تواند به عنوان یک نوع خاص بسته بندی شود، دانش انتزاعی نیاز به بسته بندی در انواع متفاوت دارد تا نیاز کاربران مختلف را برآورد سازد. بدین منظور دانش انتزاعی به انواع رسوم[8] و روش ها[9] تقسیم می شود. بنابراین به طور کل سه نوع دانش توسعه عبارتند از:

- **رسوم:** بهترین اعمال[10] یک صنعت را بصورت کلی نشان می دهند. به طور مثال، مستند سازی کارایی یک فرایند، شاخص های ابزار و...
- **روش ها:** بر جزئیات یک فرایند خاص یا بهترین عمل تمرکز دارد. این بسته ممکن است متدولوژی خاص اجرای یک فرایند و یا چگونگی انجام زیر فرایند هایش را مستند سازی کند.
- **دانش واقعی:** این بسته ها کاملاً وابسته به دنیای واقعی هستند و تجربیات در دست را مستند سازی می کنند. یک بسته واقعی چگونگی انجام یک بسته خلاصه را در یک زمینه خاص نشان می دهد و موفقیت یا شکست آن را شرح می دهد.

به عبارتی دیگر رسوم شاخص های بهترین اعمال صنعت را نشان می دهند، روش ها، متدولوژی های انجام این اعمال را ارائه می دهند و بسته های واقعی، تجربه کسب شده در اجرای فرایند را توصیف می کنند. این سه نوع دانش به عنوان تجارب توسعه در سطوح مختلف انتزاع در شکل(4) دیده می شود.

[8] Praxis
[9] Modus
[10] Best Practice



شکل شماره(۴)- سطوح مختلف انتزاع دانش توسعه نرم افزار

در اثر اجرای فرایند توسعه، تجارب به عنوان یک بسته واقعی ذخیره می شوند و با تحلیل یک مجموعه از بسته های واقعی، موارد خاص محیطی خلاصه شده و به صورت بسته های روش ارائه می گردد و پس از جمع بندی، به صورت شاخص ها، ریسک های مرور فنی و .. در بسته های رسوم ارائه می شود.

۵. فرایند توسعه نرم افزار و چرخه مدیریت دانش

فرایند تولید و توسعه نرم افزار با توجه به متدولوژی های مختلف توسعه نرم افزار متفاوت است، لیکن بعضی از فرایندهای اصلی با توجه به نیازهای یک محصول نرم افزاری در تمامی متدولوژی های مورد استفاده مشترک می باشد. عمده این فعالیتها عبارتند از:

1.تحلیل نیازمندی ها

2.طراحی

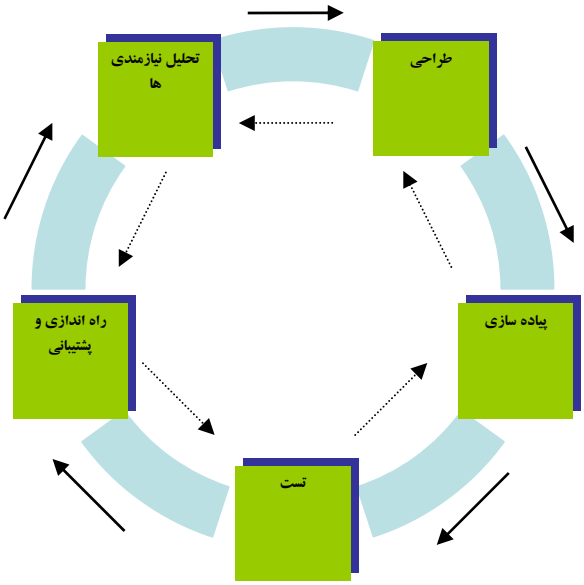
3.برنامه نویسی و پیاده سازی

4.تست

5.راه اندازی و پشتیبانی

این فرایند که به صورت یک چرخه در طول حیات یک محصول نرم افزاری بطور مداوم انجام می شود (شکل5) از فاز تحلیل شروع شده و به پشتیانی از محصول تولیدی می انجامد. چنانچه بخواهیم بر اساس این فرایند، دانش موجود در توسعه نرم افزار را مدیریت نماییم و در اثر آن فرایندها را تسهیل بخشیده و بهره وری سازمانی بهبود یابد، می بایست بر اساس مدل های استفاده مجدد و چرخه "استدلال مورد پی"که بدان اشاره شد این فرایند را پشتیبانی نمود. بر اساس این چرخه، دانش از فاز تحلیل نیازمندی ها با توجه به نیاز مشتری که ناشی از یک کسب و کار واقعی است پس از تحلیل و مستند سازی به فاز طراحی انتقال می یابد، آنگاه این دانش با ساختاری جدید و در قالب خروجی های فاز طراحی در اختیار برنامه نویس قرار می گیرد و به صورت یک برنامه رایانه ای در می آید ودر مرحله تست با توجه به مستندات

تحلیل این برنامه مورد آزمون قرار می گیرد. با توجه به این چرخه، چنانچه این دانش و تجارب به صورت موارد مختلف در پایگاه تجربه ذخیره گردد، می تواند مورد استفاده سیستم های پشتیبانی که در فرایند توسعه نرم افزار شرکت نداشته و این دانش به صورت عمیق و دقیق به آنها انتقال نیافته است، منتقل شود. از طرفی تیم های پشتیبانی، افرادی درگیر با مشتری هستند که به طور مستقیم با نیازهای کاربران سر و کار دارند و می توانند نیاز های جدید – که از بکارگیری برنامه های تولیدی و فهم نیازهای تولید نشده به وجود آمده است– را با دانشی که از برطرف نمودن نواقص[11] برنامه در شرکت های مشتریان کسب کرده اند، در اختیار تحلیل گران قرار دهند . از طرفی آنها می توانند کار تیم تست را چک نموده و در مورد صحت تست بازخور دهند. بنابر چنین مدلی، این دانش به صورت یک چرخه مدام حرکت نموده، بر اثر مرور زمان نکامل یافته و باعث بهبود محصولات نرم افزاری و افزایش کارایی سازمانی می شود. این مدل در شکل(5)نشان داده شده است، در این مدل، فلش های پر رنگ نشان دهنده دانشی است که در فازهای مختلف به مراحل بعدی انتقال می یابد و فلش های نقطه چین بازخوری است که به مراحل قبل داده می شود و در مجموع هر دو جریان به تکامل دانش موجود در پایگاه تجربه منجر می شود.



شکل شماره(۵) – فرایند مدیریت دانش توسعه نرم افزار

نتیجه گیری:

فرایند استفاده مجدد از محصولات و دانش توسعه نرم افزار با چرخه استدلال مورد پی تطابق زیادی دارد، با استفاده از این مدل ها و ترکیب آنها می توان دانش توسعه نرم افزار را به خوبی در پایگاه های تجربه ذخیره نموده و با توجه به نیاز در اختیار مراحل مختلف این فرایند که بصورت یک چرخه مدام در حال گردش است، قرار داد. توسعه دهندگان نرم افزار به راحتی می توانند با استفاده از سیستم دانش پی حاصل، مسائل خود را با توجه به تجربیات گذشته حل نموده، تجارب جدید را وارد پایگاه کرده و موارد گذشته را به روز آوری کنند. در اثر تعامل با این سیستم، بهره وری، کیفیت محصول نرم افزاری، یادگیری و رضایت توسعه دهندگان افزایش یافته و زمان و هزینه تولید نرم افزار کاهش می یابد.

[11] Bug

منابع و مراجع:

[۱] Aamodt, A. & Plaza, E. (۱۹۹۴). Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. AI Communications, vol.۷ , No.۱, pp.۳۹-۵۹
[۲] Althoff, K-D. Birk, A., Wangenheim, C.G.V., Tautz, C., (۱۹۹۸) CBR for experimental software engineering. In Burkhard, Lenz et al., eds., Case-Based Reasoning Technology from foundations to applications, Springer, Berlin
[۳] Althoff, K-D., Aamodt, A.,(۱۹۹۶) relating case-based problem solving and learning methods to task and domain characteristic: towards an analytic framework , vol.۹, No.۳
[۴] Basili, V.R.,(۱۹۸۵) Quantitative evaluation of software engineering methodology, first pan pacific computer conference
[۵] Basili, V.R., Caldiera, G., (۱۹۹۱) Methodological and architectural issues in the experience factory, ۱۶th annual software engineering workshop
[۶] Basili, V.R, G.Caldiera, and H.D.Rombach,(۱۹۹۴) the experience factory, in Encyclopedia of software Engineering, Ed.J.J.Marciniak, John Wiley, pp.476۴۶۹-
[۷] Carlsen, S., Johnsen, H.D., Coll, G.J., Maehle, A., Carlsen, A., Hatling, M., (۱۹۹۹) knowledge re-activation mediated through knowledge carriers, conference on management of information and communication technology
[۸] Curtis, B., Krasner, H., Iscoe, N. (۱۹۸۸), a field study of the software design process for large systems, communications of the ACM,۳۱(۱۱),pp.۱۲۶۸-۱۲۸۷
[۹] Henninger, S. (۱۹۹۷), Capturing and formalizing best practices in a software development organization, ۹th Int. Conference on software engineering and knowledge engineering, Spain
[۱۰] Henninger, S. (۱۹۹۷), Case-Based Knowledge Management Tools for software development, Journal of Automated Software Engineering, vol.۴, No.۱
[۱۱] Kolodner, J.L., (۱۹۹۳) Case-Based Reasoning, Morgan-Kaufman, San Mateo, CA.
[۱۲] Kolodner, J.L., (۱۹۹۱) Improving Human Decision Making through Case-Based Decision Aiding, AI magazine,۱۲ (۱), pp. ۵۲-۶۸
[۱۳] Mall, R.,(۲۰۰۳) fundamentals of software engineering, Prentice-Hall of India,۲nd edition
[۱۴] Pearce, M., Goel, A.K, Kolodner, J.L., Zimring, C., Sentosa, L., Billington, R., Case-Based Design Support: A Case Study in Architectural Design, IEEE Expert,۷ (۵), pp.۱۴-۲۰
[۱۵] Rus, I. and Lindvall, M. (۲۰۰۲), Knowledge Management in Software Engineering, IEEE Software
[۱۶] Shepperd, M. (?) Case-Based Reasoning and Software engineering
[۱۷] Tautz, C., Althoff K-D (۱۹۹۷) using Case-Based Reasoning for reusing software knowledge,۲ nd International Conference on Case-Based Reasoning
[۱۸] Tautz, C., Althoff, K-D, Nick, M., (۲۰۰۰) A Case-Based approach for managing qualitative experience, workshop at ۱۷th national conference on AI
[۱۹] Walz, D.B., Elam, J.J., Curtis, B., Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration, Communications of the ACM, ۳۶(۱۰), pp. ۶۲-۷۷
[۲۰] Wangenheim, C.G.V., Althoff, K-D., Barcia, Tautz, C.,(۱۹۹۸) evaluation of technologies for packaging and reuse of software engineering, Int. Conference on Industrial engineering applications of artificial intelligence and expert systems