جاسازی فعالیتهای معماری در روش توسعه FDD

فرزانه اکبری
دانشگاه آزاد اسلامی واحد نجف آباد
farzaneh.akbari@sco.iaun.ac.ir

سید مهران شرفی
دانشگاه آزاد اسلامی واحد نجف آباد
Mehran_sharafi@iaun.ac.ir

ناصر نعمت بخش
دانشگاه آزاد اسلامی واحد نجف آباد
nemat@eng.ui.ac.ir

چکیده: پیدایش روشهای چابک در مهندسی نرم افزار از یک دهه قبل می‌باشد. علت ظهور این روشها را میتوان در مشکلات روش های پیشین در مواجهه با پروژه های نرم افزاری و پیچیدگی ذاتی تولید نرم افزار پیدا نمود .در متدولوژی‌های سنگین وزن به دلیل حجم بالای مستندات، برنامه ریزی جامع، مستندسازی از ابتدا تا انتها و طراحی کامل و گسترده، سرعت توسعه سیستم تا حد بالایی کاهش می‌یابد ولی در مقابل، روش های توسعه‌ی چابک بر روی تسریع و سادگی در مراحل تحلیل و ساخت نرم افزار تمرکز دارند و جهت توسعه‌ی سیستم های نرم افزاری که ویژگی‌هایی نظیر دوره‌ی تحویل کوتاه و نیازمندیهای نامشخص دارند به عنوان شیوه های محبوب و کارآمد شناخته می‌شوند. آنها بر رضایت مشتری، پاسخ سریع به تغییرات و انتشار در زمان کمتر تاکید می‌کنند. یکی از چالشهای روشهای چابک مربوط به توجه کم این روشها به فعالیتهای معماری نرم افزار است. تاکنون راه حلهای مختلفی برای مقابله با این چالش پیشنهاد شده است، در این مقاله به ارائه رویکردی برای پاسخگویی به این چالش در روش FDD[1] خواهیم پرداخت.

رویکرد پیشنهادی یک طراحی معماری براساس ویژگی‌های کیفی در ابتدای روش FDD ایجاد می‌کند به طوری که توجه لازم را به اصول چابکی و ارزشهای FDD داشته باشد. به عبارت دیگر راه حل پیشنهادی برای دستیابی به مزیتهای معماری نرم افزار برای روش FDD به گونه ای عمل می‌نماید که با ارزشها و اصول چابکی در تناقض قرار نگیرد. در این مقاله یک مطالعه موردی بررسی شده است که هدف آن نشان دادن کاربرد پذیری و ارزیابی رویکرد پیشنهادی به طور عملی بوده است.

واژه های کلیدی: چابکی، روش FDD، معماری نرم افزار، فعالیتهای معماری، ویژگی‌های کیفی.

۱- مقدمه

در دهه های اخیر، تلاشهای متنوعی برای ارائه روشهای توسعه موثر که توان پشتیبانی از محیط های گوناگون توسعه و نیازمندیهای متنوع مشتریان را داشته باشند، صورت پذیرفته است. روشهای چابک[2] به عنوان نسل جدید روشهای توسعه مطرح گردیدند و تلاش نمودند بر مشکلات روشهای گذشته غلبه نمایند. به هر صورت روشهای چابک نیز محدودیتهایی دارند [1,2] که یکی از آنها مربوط به معماری نرم افزار و عدم توجه لازم به اهداف و مزایای آن است [3,4]. به طور کلی مدلهای توسعه‌ی نرم افزار عموماً به دو دسته تقسیم می‌شوند:

۱) مدلهای قدیمی[3] : Incremental، UP، Spiral، RAD، Prototyping،Waterfall

۲) مدلهای چابک[4] : Crystal، ASD، DSDM، Scrum، XP ،FDD

مشکل اصلی مدلهای قدیمی، مستند سازی سنگینی است که در تمام مراحل توسعه‌ی نرم افزار وجود دارد و باعث تاخیر در سرتاسر نرم افزار می‌شود. روشهای چابک فرآیندهای کم وزنی هستند که نیازمند مستندسازیهای کم می‌باشند و باعث کاهش تاخیر در توسعه می‌شوند و همچنین در این روشها ارتباطات تکراری و قوی مابین سهامداران و توسعه دهندگان نرم افزار وجود دارد. روشهای چابک سعی در کنترل سریع تغییرات و کمتر نمودن زمان مورد نیاز و هزینه ها دارند. در این مقاله یک معماری بر اساس ویژگی‌های کیفی در مرحله‌ی اول روش FDD طراحی شده است.

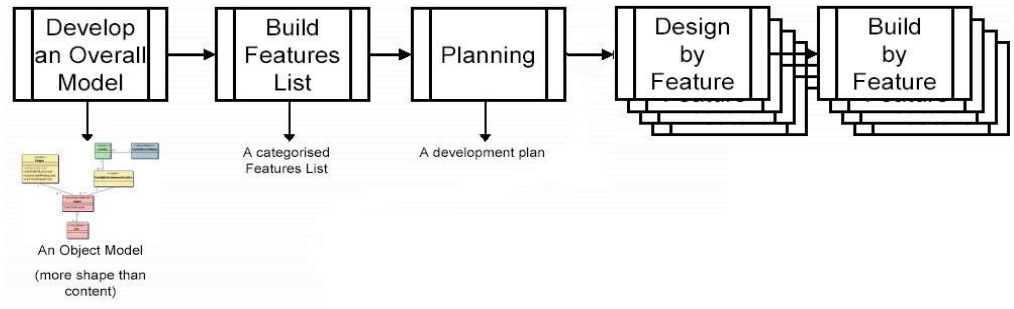
ادامه مقاله به این صورت سازماندهی شده است: در بخش ۲ به معرفی متدولوژی FDD پرداخته ایم، در بخش ۳ رویکرد پیشنهادی برای روش FDD ارائه شده است، در بخش ۴ به برخی از کارهای انجام شده در این زمینه اشاره نموده ایم و بخش ۵ سیستم کلاینت سرور را برای ارزیابی رویکرد پیشنهادی مورد بررسی قرار داده ایم و بخش ۵ نتیجه گیری می‌باشد.

[1] Feature Driven Development
[2] Agile Methods
[3] traditional
[4] agile

۲- معرفی متدولوژی FDD

FDD توسط Jeff De Luca و Peter Coad ، در سال ۱۹۹۷ عرضه شد [5]. FDD از معدود متدهای چابک است که در پروژه های نرم افزاری بزرگ می‌تواند به خوبی به کار گرفته شود و به عنوان یک هسته‌ی قابل افزایش عمل می‌کند و بر جنبه های کیفیتی تاکید دارد و در فواصل زمانی مشخص بخشی از کار را تحویل مشتری می‌دهد [5]. فرآیند آن کل زمان پروژه را در بر نمی‌گیرد و بیشتر روی دو مرحله طراحی و پیاده سازی متمرکز می‌شود.

متدولوژی FDD ترکیب قدرت و سادگی است و در کارهای تجاری مورد نیاز است [5]. همانطور که از نام این متدولوژی بر می‌آید پایه‌ی همه‌ی کارها و توسعه نرم افزار بر مبنای feature ها است. یک feature یک تابع client-valued می‌باشد که می‌تواند در دو هفته یا کم تر پیاده سازی شود.



شکل ۱. مراحل متدولوژی FDD

۲-۱ فرآیندهای متدولوژی FDD

FDD شامل پنج مرحله می‌باشد که دو مرحله‌ی آخر به صورت تکراری می‌باشند(شکل۱)، که به شرح زیر می‌باشند [6]:

۱) توسعه‌ی یک مدل عمومی[5]
۲) ساخت لیست feature ها[6]
۳) برنامه ریزی بر اساس feature ها[7]
۴) طراحی بر اساس feature ها[8]
۵) ساخت بر اساس feature ها[9]

زمان تکرارها در این متدولوژی کم تر از دو هفته است. که در ادامه هر کدام از فرآیندهای FDD را به اختصار شرح می‌دهیم.

۲-۱-۱ توسعه یک مدل عمومی

در این فرآیند، تیم مدل کننده که شامل نماینده‌ی کاربر، سربرنامه نویس ها و متخصصان دامنه می‌باشد، تشکیل می‌شود و تحت سرپرستی سر معمار است و با استفاده از یک شناخت کلی از دامنه و نیازهای سیستم، در قالب یک چرخه مدل سازی انجام می‌شود و مدل موجودیت ها به دست می‌آید. سپس چند زیر گروه از تیم برای مدل کردن یک ناحیه منصوب می‌شوند، مدل های ارائه شده توسط آن ها ادغام شده و مدل دامنه و مدل کلی اشیا بدست آمده با هم ادغام می‌شوند. به طور کلی این مرحله شامل ساخت نمودار کلاس می‌باشد که انواع قابل توجهی از اشیا موجود در یک حوزه‌ی مسئله و روابط مابین آنها را به تصویر می‌کشد که یک شکل از تجزیه‌ی سیستم می‌باشد. مسئله به اشیا مهمی‌شکسته می‌شود، طراحی و پیاده سازی هر شی یا کلاس در این مدل یک مسئله‌ی کوچکتر برای حل می‌باشد، سرانجام پس از اینکه کلاسهای کامل شده با هم ترکیب شدند، آنها راه حلی برای مسئله‌ی بزرگتر را نشان می‌دهند.

۲-۱-۲ تهیه‌ی لیست feature ها

در تهیه‌ی لیست feature ها، تیمی‌با حضور مدیر پروژه، نماینده‌ی کاربر و سربرنامه نویس ها تشکیل می‌شود و اقدام به تهیه‌ی لیستfeature در سه سطح ناحیه، فعالیت و قدم که همانfeature است، می‌کند. این کار به صورت بالا به پایین انجام می‌شود و انواع تکنیک ها مورد استفاده قرار

[5] Develop an Overall Model
[6] Build Feature List
[7] Plan By Feature
[8] Design By Feature(DBF)
[9] Build By Feature(BBF)

می‌گیرد که سطوح انتزاع از سه سطح بیش تر نشود. نتیجه‌ی حاصله یک سلسله مراتب لیست feature طبقه بندی شده است که بیشتر آنها از متدهای کلاس ها در مدل عمومی‌ایجاد می‌شوند، متدها در مدل داخل feature ها انتقال می‌یابند. یک feature یک تابع کوچک مقدار دهی شده توسط کاربر است که در طی ۲ هفته پیاده سازی می‌شود و قالب آن به صورت زیر است [6]:

<action> the <result> <by|for |of | to|> <a(n)> <object>

به طور مثال:

Calculate [action] the total [result] of a sale [object].

۲-۱-۳ برنامه ریزی بر اساس feature ها

در برنامه ریزی با feature ابتدا تیمی‌متشکل از مدیر پروژه، برنامه نویسان ارشد و مدیر توسعه ایجاد می‌شود که زمان تکمیل هرکدام از فعالیت ها و در نتیجه زمان تکمیل برای هر مجموعه feature، مجموعه feature های اصلی و ناحیه ها را مشخص می‌نماید و براساس آن برنامه ریزی را انجام می‌دهد و توالی توسعه مشخص می‌شود، سپس هر مجموعه feature به یک برنامه نویسان ارشد تحویل داده می‌شود.

۲-۱-۴ طراحی براساس feature ها

در طراحی با feature ابتدا تیم feature تشکیل می‌شود سپس کار به این صورت است که برنامه نویس ارشدی که مسئول یک مجموعهfeature است صاحبان کلاس هایی که لازم دارد را دور هم جمع و در صورت لزوم در شناخت محیط به آن ها کمک می‌کند. در این مرحله برای پیاده سازی feature ها نمودار توالی ایجاد می‌شود و مشخص می‌شود که اشیا چگونه باید تعامل کنند تا یک feature پیاده سازی شود. در نهایت ساختار درونی کلاس ها و متدها تهیه و یک بسته‌ی طراحی برای آنها ایجاد می‌شود.

۲-۱-۵ ساخت بر اساس feature ها

در نهایت در مرحله‌ی ساخت با feature ها متدها به همراه آزمون هایشان و بر اساس توصیف سطح طراحی پیاده سازی می‌شوند. کدها بررسی می‌شوند تا مطمئن شویم صحیح هستند و با استانداردها مطابقت دارند. در سطوح مختلف آزمون را انجام می‌دهیم تا مطمئن شویم که نیازمندی‌ها پیاده سازی شده اند و در نهایت در صورت عدم وجود مشکل حاصل کار را به مجموعه‌ی ساخت اضافه می‌کنیم. دو مرحله‌ی آخر، طراحی و ساخت بر اساس feature به صورت تکراری تا پیاده سازی کامل تمام feature انجام می‌شود که هر تکرار بین چند روز تا دو هفته طول می‌کشد که شامل موارد طراحی، بازبینی طراحی، کد نویسی، آزمایش، یکپارچه سازی و بازبینی کد می‌باشد. در پایان هر تکرار و پس از بررسی موفقیت آمیز کد، مجوز ساخته شدن کد داده می‌شود و یک محصول اصلی خواهیم داشت.

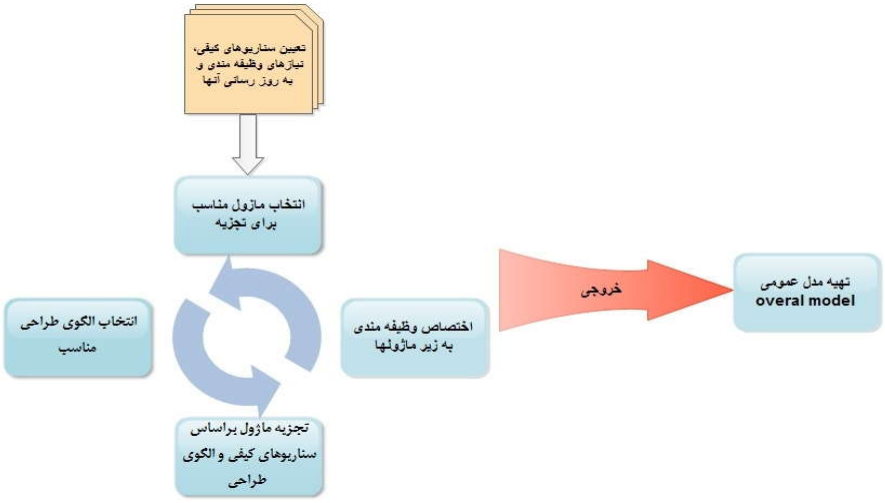
۳- رویکرد پیشنهادی

هدف رویکرد پیشنهادی در این مقاله، دستیابی به مزایای معماری نرم افزار در روش FDD است که این مزایا بطور مشخص دستیابی به ویژگی‌های کیفی، ایجاد مدل مفهومی‌از نرم افزار و ارتباط مناسب میان ذینفعان از طریق زبان مشترکی به نام معماری سیستم است. در این مقاله یک معماری مناسب برای روش FDD طراحی شده تا به ویژگی‌های کیفی مورد نیاز دست یابیم، روش FDD خود از یک فاز ضمنی جهت طراحی معماری برخوردار است ولی برآورنده‌ی ویژگی‌های کیفی نیست و با توجه به اینکه اگر معماری به ویژگی‌های کیفی توجهی نداشته باشد، معماری ضعیفی خواهد بود، بایستی از همان ابتدای توسعه به ویژگی‌های کیفی توجه نمود تا بتوان در طراحی و پیاده سازی سیستم اعمال کرد. در نتیجه در مرحله‌ی اول FDD که ایجاد مدل عمومی‌می‌باشد، ابتدا تیم مدل کننده، تحت سرپرستی معمار ارشد تشکیل می‌شود، تمام اعضای مشارکت کننده در FDD باید از هدف و انگیزه‌ی توسعه‌ی سیستم، آگاه باشند. یعنی محیط تجاری و کل سیستم باید مشخص شود تا بتوان ویژگی‌های کیفی آن را به طور موثر تحلیل کرد به عبارتی باید یک شناخت کلی از دامنه به دست آید، سپس چند زیرگروه از تیم برای مدل کردن هر ناحیه منصوب می‌شوند. به عبارتی سیستم را بر اساس ویژگی‌های کیفی تجزیه می‌نماییم به عبارتی در قالب یک چرخه، سیستم تجزیه و مدل عمومی‌بر اساس ویژگی‌های کیفی ایجاد می‌شود که مراحل آن به شرح زیر است:

۱) ابتدا نیازمندی‌های وظیفه مندی، غیر وظیفه مندی و هدف از توسعه‌ی سیستم با همکاری سهامداران و معمار مشخص می‌شود.
۲) ماژولی را برای تجزیه انتخاب می‌کنیم که ماژول شروع معمولا کل سیستم است.
۳) الگوی طراحی مناسبی برای رسیدن به ویژگی کیفی مورد نظر اتخاذ می‌کنیم.
۴) ماژول انتخاب شده را بر اساس ویژگی‌های کیفی و با انتخاب الگوی طراحی مناسب توسط معمار، تجزیه می‌کنیم.
۴) نیازهای وظیفه مندی را به زیر ماژولها اختصاص می‌دهیم.
۵) برا ی ماژولی که نیاز به تجزیه‌ی بیشتر دارد به مرحله ۳ برمی‌گردیم و مراحل بالا را تکرار می‌کنیم.

۶) در مرحله‌ی آخر از تجزیه، معمار مدلهای ایجاد شده را با هم ادغام نموده و یک مدل عمومی و یک معماری سطح بالا از سیستم بر اساس ویژگی‌های کیفی ایجاد می‌کند. به طور کلی الگوهایی که انتخاب می‌شوند، به تصمیمات معماری کمک خواهند نمود. به عبارتی انتخاب معماری مناسب، برای دستیابی به ویژگی‌های کیفی مورد نظر بسیار مهم است.

با اضافه شدن ویژگی‌های کیفی به مرحله‌ی اول FDD، معماری اولیه سیستم مبتنی بر ویژگی‌های کیفی طراحی می‌شود زیرا ویژگی‌های کیفی ساختار سیستم را تعیین می‌نمایند و بدون در نظر گرفتن آنها هر ساختاری می‌تواند برآورنده نیازهای وظیفه مندی توصیف شده در قالب feature باشد (شکل ۲).



شکل۲. فعالیتهای معماری جاسازی شده در مرحله اول FDD

این مدل نیز مانند دیگر مدلهای اولیه در طول توسعه و با مواجهه با تغییرات نیازمندیها، دچار تغییراتی خواهد شد. این نکته بایستی مدنظر قرار گیرد که در شرایط جدید، زوج برنامه نویس علاوه بر feature ها که حاوی وظیفه مندی مشخصی هستند، از ویژگی‌های کیفی آن نیز مطلع می‌گردد و بایستی به گونه ای طراحی و برنامه نویسی را انجام دهند که برآورنده ویژگیهای کیفی باشند. هیچ ویژگی کیفی کاملا وابسته به یک مرحله خاص مثلا طراحی، اجرا یا توسعه نیست و می‌بایستی در سرتاسر مراحل طراحی، اجرا و توسعه در نظر گرفته شود. خروجی بدست آمده از این مرحله، ایجاد یک معماری مبتنی بر ویژگی کیفی می باشد.

در مرحله‌ی دوم با توجه به مدلی که از مرحله‌ی اول بدست آمده است، اقدام به تهیه‌ی لیستی از feature ها می‌کنند، که به دلیل در نظر گرفتن ویژگیهای کیفی و استفاده از الگوهای طراحی، تعداد featureها نسبت به قبل بیشتر می شود ولی از طرفی به دلیل اینکه در مرحله قبل نیازمندیهای وظیفه مندی مشخص شده اند در نتیجه لیست feature ها سریعتر آماده می‌شود.

در مرحله‌ی سوم، با توجه به feature ها، یک برنامه ریزی زمانی برای اجرای آنها تعیین می‌شود. مدیر پروژه، مدیر توسعه، و برنامه ریز ارشد ترتیبی که feature ها باید پیاده سازی شوند را، بر مبنای وابستگیهای feature ها، بار کاری تیم توسعه، و پیچیدگی feature هایی که پیاده سازی می‌شوند طرح ریزی می‌کنند و با استفاده از توالی توسعه و اولویت feature به عنوان یک راهنما، تیم طرح ریزی، سر برنامه نویسان را به عنوان مالکان مجموعه feature ها انتساب می‌کند.

در مرحله چهارم و با استفاده از feature ها طراحی صورت می‌گیرد و در صورت بروز تغییر در نیازمندیها مراحل بالا مجددا تکرار می‌شوند. برنامه ریز ارشد،feature برای توسعه را از لیست feature اختصاصی اش انتخاب می‌کند. وی ممکن است feature های متعددی را انتخاب نماید، سپس مدل شی را بر مبنای نمودار توالی پالایش می‌کند. در نهایت ساختار درونی کلاس ها و متدها تهیه و یک بسته‌ی طراحی برای آنها ایجاد می‌شود. در این هنگام برنامه نویس اصلی، feature های بعدی را دریافت و کلاسهایی را که احتمالا با آن درگیر می‌باشند را شناسایی نموده و با مالکین کلاس متناظر تماس می‌گیرد.

در مرحله‌ی آخر که ساخت بر اساس feature ها می‌باشد، هر مالک کلاس، متدهایش را برای feature می‌سازد و متدها به همراه آزمون هایشان و بر اساس توصیف سطح طراحی پیاده‌سازی می‌شوند سپس کدها بررسی و با استانداردها مطابقت داده می‌شوند. در سطوح مختلف آزمون هایی برای اطمینان از دستیابی به نیازمندی‌ها و ویژگی‌های کیفی مورد نظر انجام می‌شود و در نهایت در صورت عدم وجود مشکل حاصل کار را به مجموعه‌ی ساخت اضافه می‌کنیم تا یک محصول اصلی داشته باشیم و کار با feature های جدید ادامه پیدا می‌کند و دو مرحله‌ی آخر تکرار می‌شود تا تمام

feature ها پیاده سازی شوند و در نهایت با استفاده از feature ها و ویژگی‌های کیفی تعیین شده، سیستم توسعه می‌یابد. ایجاد مدل معماری از سیستم در روش پیشنهادی، همگام با فرآیند توسعه‌ی FDD پیشرفت می‌کند، در نتیجه چابکی FDD حفظ می شود.

۴- کارهای مرتبط

در [۱] نویسنده بر روی تعبیه فعالیتهای معماری نرم افزاری CAR[10] و RAQ[11] در فرآیند توسعه‌ی XP متمرکز شده است. هر دو فعالیت پیشنهاد شده در پی دستیابی به ویژگی‌های کیفی می‌باشند. روش CAR برای برآوردن ویژگی‌های کیفی مدنظر توسعه دهندگان، برنامه نویسان و آزمون کنندگان مانند قابلیت استفاده مجدد، قابلیت تغییر و قابلیت آزمون‌پذیری طراحی شده است و به توسعه دهندگان کمک می‌کند تا با شناسایی ضعفهای معماری و تشخیص راه حل هایی جهت تجدید نظر در آنها در انتهای هر فرآیند، ضعف های معماری را کاهش دهند و به عنوان یک فعالیت بهبود معماری استفاده می‌شود. روش RAQ، در پی برآوردن ویژگی‌های کیفی از جمله کارآیی و قابلیت استفاده می‌باشد که شیوه ای عملی و مشابه جلسه طوفان ذهنی است و در انتهای تمام تکرارها فعال می‌شود. هدف اصلی آن آزمایش سیستم می‌باشد، به عبارتی بررسی می‌کند که خروجی هر تکرار منطبق بر نیاز مشتری به همراه ویژگی‌های کیفی مورد نظر باشد. یکی از مزایای مهم روش RAQ این است که از سیستم کاری آماده جهت ارزیابی ویژگی‌های کیفی و مدل های معماری استفاده می‌کند. علاوه بر این ، فعالیتهای پیشنهاد شده به صورت موازی با فرآیند توسعه‌ی XP طراحی شده، به گونه ای که چابکی XP را حفظ نماید. همچنین در [۱] نویسنده یک نقش معمار به روش XP اضافه نموده است که باعث افزایش هزینه‌ی توسعه می شود و برنامه نویسان پس از توسعه‌ی وظیفه‌ی محوله، یک مدل ساده از سیستم، به معمار ارسال می‌نمایند، سپس معمار این مدلها را جمع‌آوری نموده و بعد از بررسی آنها و کشف ضعفها راه حلهایی را برای رفع آنها ارائه می‌نماید و مجدداً به صورت وظیفه‌ی جدید به برنامه نویسان محول می شود که این خود مستلزم زمان زیادی می باشد، در صورتی که در روش پیشنهادی برای FDD ابتدا سیستم در یک چرخه‌ی تکراری با در نظر گرفتن ویژگی‌های کیفی و با استفاده از الگوها و تاکتیکهای معماری مناسبی که معمار انتخاب می کند به ماژولهای کوچکتری تجزیه می‌شود (بخش ۳). این تجزیه باعث کاهش پیچیدگی مسئله شده، بعد از اتمام تجزیه یک مدل و معماری از سیستم توسط تیم مدل کننده مبتنی بر ویژگی‌های کیفی، ایجاد و بقیه‌ی مراحل FDD بر اساس این مدل اجرا می‌شود.

در [7]، مولفین بر اصلاح روش ATAM تمرکز می‌کنند تا آن را برای روش کریستال که یکی از متدهای چابک است، برنامه ریزی کنند. این طرح باعث افزایش کیفیت توسعه‌ی محصول با استفاده از مدل فرآیند کریستال چابک می شود. مولفین به صورت خیلی دقیق روش کریستال و ATAM را تحلیل و فازهای صفر، ۳ و مراحل ۱، ۲ و ۹ از ATAM را برای مناسب سازی و نگاشت آن در کریستال حذف نموده اند سپس رهنمودهای تئوری برای آن ارائه کرده اند.

در [8] روش ACRUM[12] بر اساس فرآیند توسعه‌ی SCRUM طراحی شده است، به طور خلاصه سه فعالیت از ACRUM وجود دارد: ابتدا تحلیل ویژگی‌های کیفی[13]، سپس تهیه جدول نقشه‌ی همبستگی بین نیازها[14] و تحلیل آن و در نهایت موفقیت یا شکست ویژگی‌های کیفی در حین نمایش هر مرحله نهایی از طریق فعالیت VAQ[15] تایید می‌شود. این سه فعالیت بدون سربار و همگام با مراحل SCRUM پیش می‌روند. علاوه بر این، همکاری مشتری که هسته‌ی ارزشی روش های چابک است، با استفاده از فعالیت AQUA تقویت می‌شود. روش های RAM و VAQ باعث سرعت پاسخ به تغییرات را افزایش می دهند.

۵- مطالعه موردی: سیستم کلاینت سرور

در این مطالعه‌ی موردی به بررسی یک سیستم کلاینت سرور پرداخته ایم. در این سیستم می خواهیم توسعه بر اساس متدولوژی FDD و با استفاده از روش پیشنهادی انجام گیرد و کارآیی سیستم افزایش یابد. حال با استفاده از روش پیشنهادی به توسعه‌ی سیستم می پردازیم و جزئیات آن را به طور کامل شرح می دهیم:

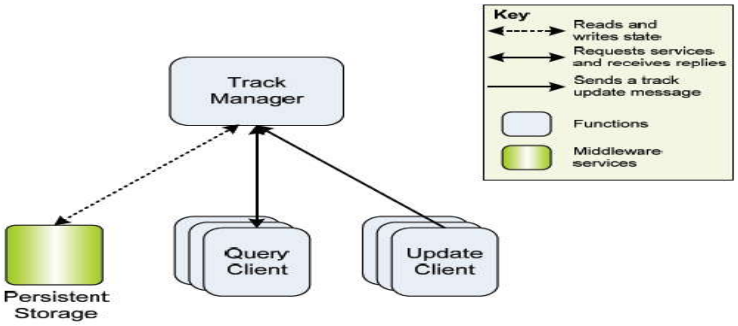
طبق مرحله‌ی اول FDD باید یک مدل عمومی برای سیستم ایجاد نماییم که این مدل را طبق روش پیشنهادی طراحی می کنیم. ابتدا نیازمندی‌های سیستم شامل نیازهای عملیاتی، محدودیتها و ویژگی‌های کیفی را مشخص می کنیم.

نیازمندی‌های عملیاتی شامل موارد زیر می باشند و در شکل ۳ نیز نشان داده شده اند:

- Track Manager: یک سروری است که پیگیری‌های لازم را برای پاسخگویی به دو نوع کاربر Update Client و Query Client مهیا می کند.

[10] Continuous Architectural Refactoring
[11] Real Architecture Qualification
[12] ACRUM(Attribute-driven sCRUM)
[13] AQUA(Analysis of Quality Attributes)
[14] RAM(Requirement Association Matrix)
[15] VAlidation of Quality attribute

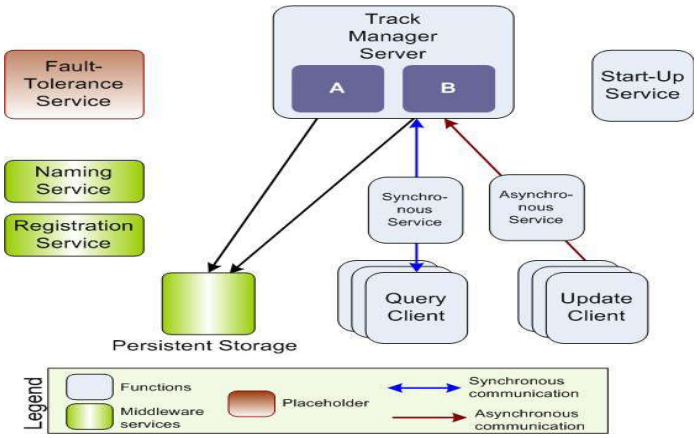
- Update Client: این کلاینت Track Update را به صورت دوره ای به Track manager ارسال می نماید.
- Query Client : این کلاینتها باید جوابی برای query خود دریافت نمایند.



شکل ۳. نیازمندی‌های عملیاتی سیستم کلاینت سرور[9]

محدودیتی که در این سیستم وجود دارد، محدودیت ظرفیت است به طوری که سرور Track manager ظرفیت محدودی در پذیرش درخواست دریافتی از Update Client و Query Client دارد و بایستی در طراحی مورد توجه قرار گیرد. سپس ویژگی‌های کیفی باید مشخص شود که در این سیستم کارآیی و قابلیت دسترس پذیری دو ویژگی کیفی بسیار مهم می باشند.

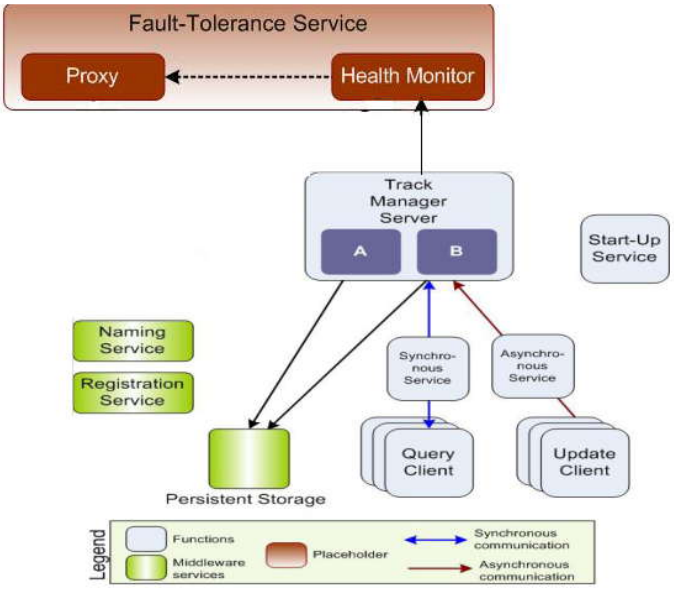
تا این زمان تمام نیازمندی‌های سیستم توسط سهامداران مشخص شده و متخصص دامنه نیازمندی‌های سیستم را کسب کرده و از حوزه‌ی عمل سیستم و محیط کسب و کار آن اطلاعات کافی پیدا کرده و سپس نویسندگان فنی این نیازمندی‌ها را مستندسازی نموده و به معمار ارشد تحویل می دهند، سپس معمار ارشد جلسات طراحی را برگزار می‌نماید و پس از بررسی نیازمندی‌ها و با توجه به ویژگی‌های کیفی مورد نیاز سیستم، ابتدا کل سیستم را به عنوان یک ماژول برای تجزیه انتخاب می کند سپس تاکتیک‌ها و الگوی طراحی مناسبی را با توجه به نیازمندی‌های تعیین شده در مرحله‌ی قبل، انتخاب می کند. به دلیل اینکه در این سیستم می خواهیم به کارآیی بالایی دست یابیم و افزایش کارآیی یک معیار مهم است در نتیجه معمار تاکتیک‌های افزایش منابع قابل دسترس[16] و همروندی[17] را انتخاب نموده و سپس تیم مدلسازی برای پیاده سازی تاکتیک‌های اعلام شده به زیرگروه‌هایی تقسیم شده و هر گروه مدل لازم برای پیاده سازی یکی از این تاکتیک‌ها را ایجاد می کند. به طوری که برای پیاده سازی تاکتیک، افزایش منابع قابل دسترس، سرور Track manager را به دو مولفه‌ی A وB تجزیه نموده که در هریک از این مولفه ها یک پردازنده قرار می گیرد و از حافظه‌ی مشترکی استفاده می کنند، همچنین برای پیاده سازی تاکتیک درجه همروندی، ارتباط مابین کلاینتها و Track manager را با مکانیسم متفاوتی برقرار می کنند، به طوری‌که مابینTrack manager و query client ارتباط همزمان[18] و مابین Track manager و update client ارتباط غیرهمزمان[19] تعریف شده است، سپس هر تیم مدل خود را به معمار ارشد تحویل می دهد و معمار ارشد مدلها را با هم ادغام نموده و اگر موارد تکراری وجود دارد آن‌ها را حذف نموده و اصلاح می نماید سپس مدل شکل ۴ را تهیه می نمایند.



شکل ۴. معماری ایجاد شده پس از اولین مرحله از تجزیه‌ی سیستم [9]

[16] Increase resources
[17] Concurrency
[18] synchronous communication
[19] Asynchronous communication

سپس در مرحله‌ی بعد از تجزیه‌ی سیستم معمار ارشد، با توجه به اینکه ویژگی کیفی قابلیت در دسترس پذیری نیز باید در سیستم وجود داشته باشد و در صورت بروز مشکل و خطایی در Track manager سیستم همواره در دسترس باشد، نیاز است که یک مولفه به نام fault tolerance به سیستم اضافه شود که در آن از تاکتیک Health monitoring برای تشخیص خطایی که در سرور Track manager رخ داده است استفاده شود در نتیجه تیم مدلسازی این مولفه را به مدل سیستم اضافه نموده و معماری سیستم به صورت شکل ۵ بدست می‌آید.



شکل ۵. معماری ایجاد شده پس از دومین مرحله از تجزیه‌ی سیستم

پس از تجزیه‌ی کل سیستم، خروجی این مرحله از FDD مجموعه ای از کلاسها و مولفه ها می باشد که دربردارنده‌ی کلیه‌ی نیازهای سیستم می باشد که در حقیق یک معماری از سیستم به وجود می آید. در مرحله‌ی دوم لیستی از feature ها تعیین می شود و مراحل بعدی FDD انجام می شود تا feature ها کاملا پیاده سازی شده و سیستم توسعه یابد.

۶- نتیجه گیری

در این مقاله بر روی روش FDD که یکی از روشهای چابک است، تمرکز نموده و تلاش شد تا معماری مناسبی در مرحله‌ی اول روش FDD ارائه گردد، به گونه ای که مزایای معماری نرم افزار مخصوصاً ویژگی‌های کیفی را در برداشته باشد و چابکی روش FDD نقض نشود. این متدولوژی یک روش در حال ساخت است و هنوز راه زیادی برای تکامل دارد [10]. از آنجا که هیچ گزارش شکستی درباره استفاده از FDD داده نشده است، پس محققان از هر تحقیق و بهبود کیفیت برای روش FDD استقبال می‌کنند. چون باعث کاربرد بهتر این متدولوژی در موقعیت‌های خاص می شود. برای اینکه فعالیتهای پیشنهادی را مورد ارزیابی قرار دهیم از یک مطالعه موردی که سیستم کلاینت سرور می‌باشد، استفاده نمودیم که نشان می‌دهد روش پیشنهادی بسیار خوب عمل می‌نماید و در مرحله اول FDD یک مدل بر اساس ویژگی‌های کیفی خواهیم داشت و در نتیجه توسعه سیستم بر مبنای این مدل انجام می‌شود. این مدل به گونه ای طراحی گردیده است که سربارکمی را متحمل تیم توسعه نماید.

مراجع

[۱] شریفلو، امیرملزم، **جاسازی فعالیتهای معماری نرم افزار در روش XP**، پایان‌نامه کارشناسی ارشد مهندسی کامپیوتر، دانشگاه شهیدبهشتی، تهران، ۱۳۸۷.

[1] Turk, D., France, R., Rumpe, B., "Limitations of Agile Software Processes", Proceedings of 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering, pp. 43-46, 2002.

[2] Beck, K., Boehm, B., "Agility through Discipline: A Debate", Computer , vol. 36, no. 6, 2003.

[3] Elssamadisy, A. and Schalliol, G., "Recognizing and responding to 'bad smells' in extreme programming", Proceedings of the 24th International Conference on Software Engineering, pp. 617-622, 2002.

[4] Jensen, R. N., Moller, T., Sonder, P. and Tjornehoj, G., "Architecture and design in extreme Programming; Introducing Developer Stories", Proceedings of 7th International Conference on Processes and Extreme Programming in Software Engineering, pp. 164 – 168, 2006.

[5] P.Coad,E.Lefebvre,J.De Luca, "Java Modeling in Color With UML: Enterprise Components and Process", Prentice Hall .1999.
[6] S.R. Palmer, J.M.Felsing, "A Practical Guide to Feature-Driven Development", Upper Saddle River, NJ,Prentice Hall.2002.
[7] S. Farhan, H. Tauseef and M. A. Fahiem, "Adding Agility to Architecture Tradeoff Anlaysis Method for Mapping on Crystal", World Congress on Software Engineering, 2009.
[8] Sanghoon , J., Myungjin, H., Eunseok , L., Keun, L., "Quality Attribute driven Development", 2011 Ninth International Conference on Software Engineering Research, Management and Applications.
[9] William G. Wood, "A Practical Example of Applying Attribute-Driven Design (ADD), Version 2.0", Technical Report, February 2007
[10] P. Abrahamsson, O. Salo, J.ronkainen,J. Warsta, "Agile Software development methods review and analysis".2002.