راهبرد متدولوژی اسکرام

مهندسی مدرن توسعه نرم افزار

مولف: احمد عادلی

فهرست

پیشگفتار

کتاب حاضر راهنمای آموزشی متدولوژی اسکرام با رویکردی کاربردی می باشد. متدولوژی اسکرام یکی از قدرتمندترین راهبردهای توسعه نرم افزار با بیشترین انعطاف ممکن می باشد. این شیوه مدیریتی در حال حاضر در بسیاری از سازمان های دنیا شیوه ای مرسوم و آزمایش شده ای می باشد. اسکرام از زمان حضور خود توانست انقلابی بزرگ در صنعت نرم افزار بسیاری از کشور ها ایجاد کند. متاسفانه با این که بیش از یک دهه از عمر این متدولوژی پر قدرت می گذرد، هنوز در کشور ما بسیار ناشناخته و مجهور مانده است. نگارنده وظیفه خود دانسته که این دانش نوین را در قالب نوشتار به صنعت نرم افزار ایران پیشکش کند.

هدف از این عمل پر کردن بزرگترین خلأ پروژه های نرم افزاری در ایران یعنی مدیریت کار گروهی می باشد. از آنجایی که در حال حاضر از شیوه های بسیار ناکارآمدی برای این مهم استفاده می شود و با توجه به میل سرکش توسعه دهندگان به اصلاح این شرایط، امید است که این متدولوژی نوین مورد اقبال عمومی قرار گیرد.

کتاب حاضر اثری تألیفی که بدون استفاده مستقیم از منبعی بر اساس دانش و تجربه نگارنده نوشته شده است. البته استفاده از کتاب ها و منابعی را که نگارنده برای آموزش طی سال ها اخیر استفاده کرده انکار ناشدنی است. این کتاب دارای دو بخش اصلی است. بخش اول آموزش اسکرام و تشریح بیانیه آجیل به صورت محض و تئوریک می باشد. این بخش با استفاده از اشکال و چارت هایی که بر اساس ایده های نگارنده می باشد تغدیه شده است. بخش دوم رویکردی کاملاً کاربردی دارد و اجرای واقعی اسکرام را در حالتی عملی با پوشش کامل نرم افزار مدیریت پروژه قدرتمند Team Foundation Server (برای اسکرام) در بر دارد. این کتاب آنچه نیاز است که یک توسعه دهنده ای راجع به مدیریت پروژه های نرم افزاری بداند، با رعایت اصل اختصار در خود جای داده است. مخاطب این کتاب همه توسعه دهندگان و مدیران پروژه های نرم افزاری می باشند که به نحوی در پی افزایش کیفیت خروجی کار خود هستند. آگاهی برخی از اعضای یک تیم از این متدولوژی چندان چاره ساز نخواهد بود و اجرای بی نقص آن مستلزم تسلط همه اعضا بر متدولوژی اسکرام می باشد.

 لطفاً برای اخذ هرگونه مشاوره، ارائه سوال، پیشنهاد و یا انتقاد از کانال ارتباطی ahmadadeli@outlook.com استفاده نمایید.

احمد عادلی

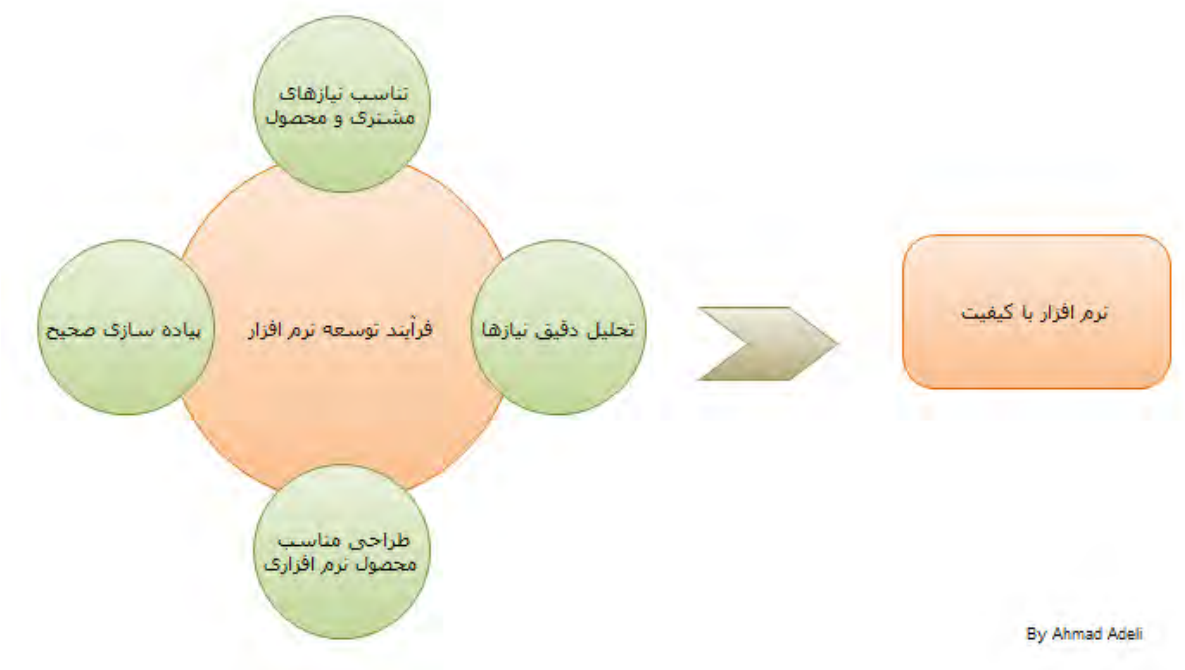
بهار 92

فصل اول: مروری بر فرآینده های توسعه نرم افزار

در دهه 60 میلادی کامپیوتر ها بسیار ساده بودند و به طبع نرم افزارهای ساده ای نیز داشتند. برنامه نویسان در آن زمان از فرآینده های خاصی جهت توسعه نرم افزار استفاده نمی کردند. یک پروژه پس از شروع آنقدر ادامه می یافت تا کامل شود. در کارهای گروهی هر فرد- هر بار قسمتی را برعهده می گرفت. در صورت مواجهه با مشکل، روند توسعه برای حل آن مشکل متوقف می شد. نا گفته پیداست که این شیوه تولید نرم افزار مشکلات زیادی را ایجاد می کرد. خروجی این فرآیند، معمولاً نرم افزاری بود که کاربرد پذیری پایینی داشت(تا حدود کمی کار می کرد). بدیهی است که نرم افزار تهیه شده ناکارآمد – ناپایدار و پر هزینه بود و به معنای واقعی کلمه بی کیفیت بود. با وجود این این گونه بی کیفیت بودن این فرآیندهای نرم افزاری در آن زمان (به دلیل عدم تنوع فرآیندها) بسیار پر کاربرد بودند. این در حالی است که بقای این نرم افزارها در صنعت کنونی نرم افزار، امری محال است.

اما سوال اینجاست که آیا مفهوم واژه *استفاده* (به صورت عام) در طول تاریخ تغییر کرده است؟ به طور کلی استقبال از یک محصول منوط به ارضای انتظارات مشتری است. در دهه 50 و 60 میلادی مفهوم کامپیوتر برای عموم هنوز به طور کامل جا نیافتاده بود و کاربران کامپیوتر اغلب افرادی متخصص بودند و نیازهایی چون کاربری آسان و متنوع چندان مورد نظر نبود. از طرفی افراد متخصص به راحتی می توانستند مشکلات نرم افزارهای تولید شده را مرتفع سازند. ناگفته نماند در آن زمان کاربران به طرز عجیبی با باگ های نرم افزاری کنار می آمدند و این گونه نرم افزارها هرچند بی کیفیت مورد اقبال قرار می گرفتند.

فاکتورهای زیادی در ساخت یک محصول نرم افزاری با کیفیت دخیل هستند. فاکتورهایی چون تناسب نیاز مشتری و محصول، تحلیل دقیق نیازها، طراحی مناسب محصول نرم افزاری، پیاده سازی صحیح و ... می توانند نمونه از از این فاکتورهای موثر باشند. هر یک از این فاکتورها به نحوی سهم خود را در تضمین کیفیت یک نرم افزار ایفا می کنند. اما آنچه بیشتر از همه روی کیفیت یک محصول موثر است *فرآیند توسعه نرم افزار* می باشد(شکل 1-1).

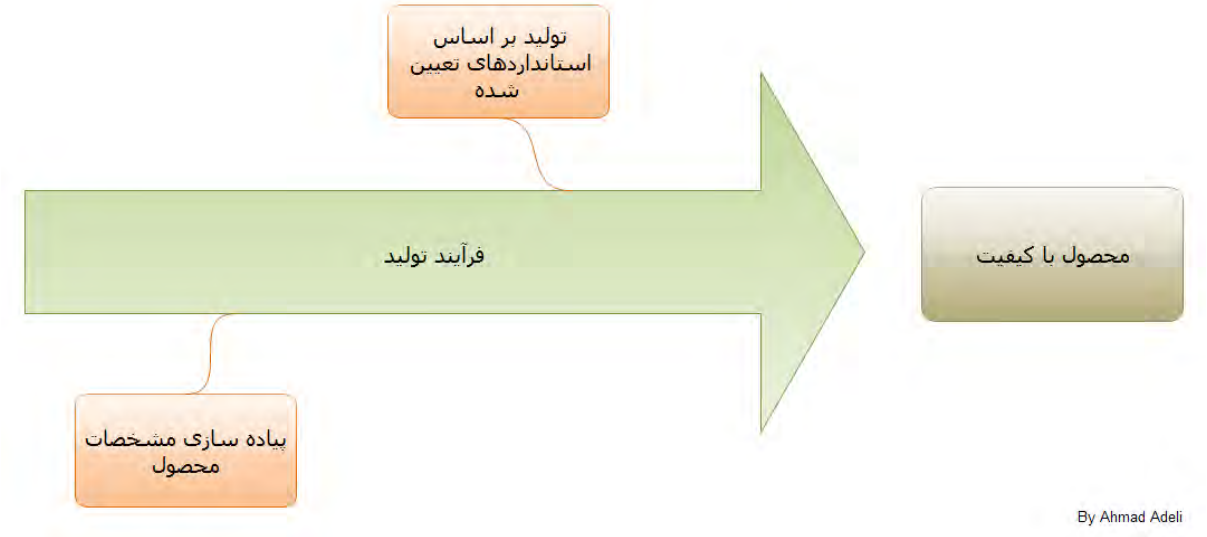


شکل 1-1 :فرآیند توسعه نرم افزار

کیفیت محصولات نرم افزاری مانند دیگر صنایع به طور مستقیم به *فرآیند تولید* وابسته است. فرآیند تولید در اصلاح به مجموعه ای فعالیت هایی اطلاق می شود که منجر به تولید محصول و یا ارئه خدمتی می گردد.

استفاده کردن از اصطلاح فرآیند تولید برای محصولات نرم افزاری اشتباه است! چرا که نرم افزار های تولید نمی شوند؛ بلکه توسعه می یابند. تولید با توسعه متفاوت است. برای عرضه محصولات تولیدی ابتدا طراحان شروع به طراحی فنی آن محصول می کنند. پس از طراحی، نمونه از آن ساخته می شود. در صورت موفقیت آمیز بودن نمونه اولیه طرح را تحت تست های زیادی قرار می دهند. پس از آن طراحی از نظر هزینه بهینه می شود. سپس استاندارد های کیفیتی روی آن اعمال می شود تا طراح کامل شود. فرض شود که طراحی کاملی در دسترس است. از این رو خط تولیدی برای محصول جدید آماده می شود. دستگاه های تولید تنظیم و پیکربندی می شوند. آنچه روی کیفیت محصول نهایی اثر گذار است چگونگی تولید محصولات بر اساس آن طراحی است؛ که این همان فرآیند تولید است. فرآیندی که در بیشتر صنایع به جز صنعت نرم افزار از آن(و یا مشابه آن) استفاده می شود. بدیهی است اگر از فرآیند تولید ناکآرامد و غیر استانداردی استفاده شود تضمین چندانی برای کیفیت محصول نهایی وجود نخواهد داشت.

فرآیند تولیدی مشخص و تعریف شده است. کافی است که سیستم های تولیدی بر اساس استانداردهای تعیین شده عمل کنند و مشخصات محصول را به خوبی پیاده سازی کنند. در این صورت به سطح بالایی از کیفیت فرآیند تولید خواهیم رسید که نتیجه آن محصولی با کیفیت است(شکل 2-1).



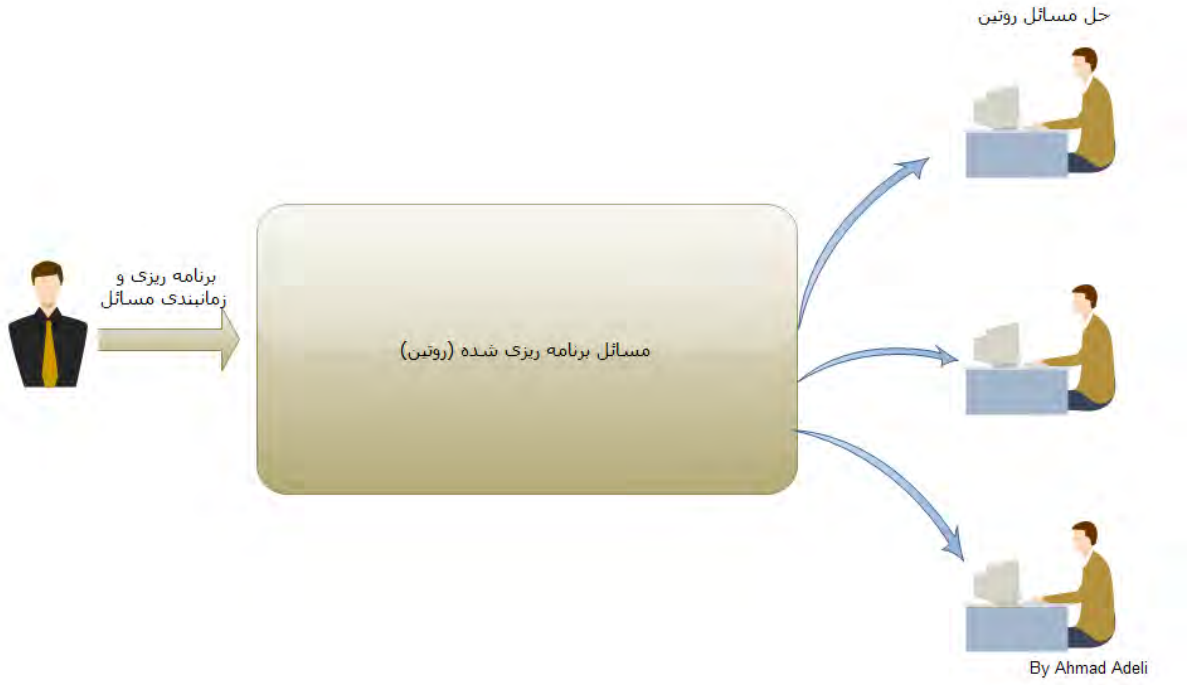
شکل 2-1: فرآیند تولید

در طرف دیگر، توسعه قرار دارد. توسعه با خلق یک ایده شروع می شود.آن ایده بسط پیدا می کند تا در نهایت به طرحی تبدیل شود. آن طرح همان *محصول* ماست. توسعه یک محصول عملی است انتزاعی مثل خلق یک موسیقی. فرآیند توسعه به مراتب پیچیده تر از فرآیند تولید است. فرآیند تهیه نرم افزار، فرآیندی توسعه پذیر است. ابتدا نیازهایی تشخیص داده می شوند. سپس مشخصات آن با استفاده از تحلیل دقیق استخراج می شود. پس از آن اجزا طراحی می شوند و به کمک تکنولوی ها و تکنیک هایی پیاده سازی می شوند. بعد از مراحل بالا در صورت نیاز، پیکربندی هایی در سیستم میزبان انجام می شود و در نهایت محصول مورد استفاده قرار می گیرد. برای نسخه های جدیدتر فرآیند فوق الذکر بارها و بارها به کار گرفته می شود.

فرآیند توسعه نرم افزار یک فرآیند خلاق و انتزاعی است و نه یک فرآیند خودکار و مکانیکی. توسعه نرم افزار فرآیندی است که کیفیت آن به عوامل زیادی از جمله: عوامل انسانی، محیطی و تکنولوژیکی بستگی دارد. این فرآیند برای تیم های توسعه مختلف می تواند متفاوت اجرا شود. گاهی نیز تاثیر سیاست های حاکم بر یک سازمان و یا یک منطقه می تواند اجرای یک فرآیند را دست خوش تغییر کند.

توسعه نرم افزار به مانند حل هزاران معما است. در توسعه نرم افزار همه روزه توسعه دهندگان با معماهای زیادی روبرو می شوند که باید آنها را حل نمایند. حل هر کدام از این مسائل نیازمند خلاقیت است. توسعه دهندگان بر اساس مهارت و خلاقیت خود بر چگونگی حل این مسائل آگاه اند. آنچه آنان مایلند بدانند این است که هر بار کدام مسئله را باید برگزینند. آن ها نمی خواهند که انتخاب مسائل را نیز خلاقانه انجام دهند(کما این

که این روش به علت تعدد افراد تیم غیر ممکن به نظر می رسد). آنچه توسعه دهندگان ترجیح می دهند اجرای فرآیندی روتین در راستای حل خلاق مسئله است. تکرار منظم و با برنامه این فعالیت روتین و البته نظارت بر اجرای می تواند ما را به سوی یک محصول با کیفیت سوق دهد(شکل 3-1).



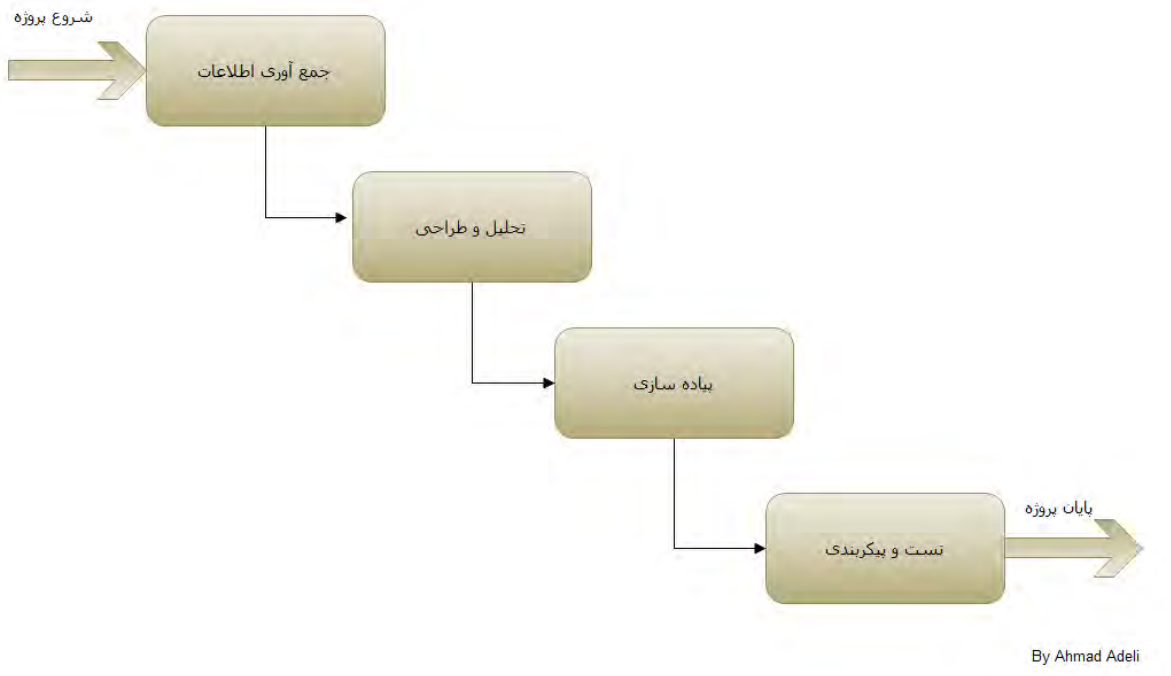
شکل 3-1: گزینش روتین مسائل

فرآیند های توسعه نرم افزار

از سال 1970 تاکنون فرآیند های توسعه نرم افزار دست خوش تغیرات بسیاری شدند. این تغیرات را می توان در سه دوره به وصف کشید:

فرآیند ترتیبی یا آبشاری

این فرآیند دقیقاً بر اساس فازهای طراحی و ساخت یک نرم افزار اجرا می شود که چهار فاز اصلی را شامل می شوند. در فاز اول نیازهای یک نرم افزار به طور کامل باید تشخیص داده شده!(با فرض محال) و جمع آوری گردد.سپس این نیازهای تحلیل می شوند و این تحلیل به طراحی اجزای نرم افزار منتج می شود. پس از دو فاز قبلی برنامه نویسی و پیاده سازی فنی نرم افزار وجود دارد. در انتها نیز نرم افزار تست و پیکربندی می شود(شکل 4-1).



شکل 4-1: فرآیند آبشاری

فاز هایی شرح داده شدند به صورت کلی و فاقد جزئیات بودند. این فازها هر کدام به نحوی در فرآیندهای مختلف حضور دارند و ترتیب و تکرار آنهاست که تفاوت فرآیندهای مختلف را رقم می زند. در فرآیند آبشاری این فازها به ساده ترین شکل ممکن وجود خود هستند. این فرآیند ریشه در تفکر سنتی توسعه دهندگان راجع به توسعه نرم افزار دارد.

کلیدی ترین مشکل این فرآیند نیازهای نرم افزار است. بر اساس تفکر این فرآیند کل نیازها به یکباره باید در ابتدای توسعه تشخیص داده شوند. به تجربه می توان گفت که این عمل غیر ممکن است. نیازهای یک نرم افزار به مرور در طول توسعه و بر اساس تجربه کاربر در کار با نرم افزار بدست خواهد آمد.در اغلب نرم افزارها، توسعه نرم افزار پایان ناپذیر است به این معنی که هر وقت نیاز جدیدی کشف شود توسعه جدیدی بر اساس آن، به سلسله توسعه های نرم افزار اضافه خواهد شد.

فرآیند آبشاری(ترتیبی) در زمان خود فرآیند پر استفاده ای بود. نرم افزار های زیادی بر اساس این فرآیند توسعه به بازار عرضه شدند. از طرفی نیز گفته شد این فرآیند دارای مشکلاتی اساسی در شناسایی نیازمندی های یک نرم افزار است. اما چگونه ممکن است به وسیله فرآیندی این چنینی نرم افزاری را توسعه داد؟! پاسخ این تناقض از دو حالت خارج نیست: حالت اول این است که مشخصات نرم افزار برای توسعه به دلیل ضعف در استخراج نیازمندی ها نا کامل و یا حتی اشتباه تهیه می شد. این موضوع باعث می شد که نرم افزار هایی که در نهایت تحویل داده می شدند نا کارآمد باشند. از این رو کاربرپذیری یک نرم افزار به شدت تنزل می یافت. حالت دیگر این است که مشکلات موجود در فازهای اولیه در فاز های بعدی خود نشان می دادند که باعث تکرار فازهای ابتدایی در تیم توسعه در راستای حل مشکلات به وجود آمده می شد. ناگفته پیداست که این عمل بخش مهمی از زمان توسعه را هدر می دهد به خصوص که اگر برای آن نیز برنامه ریزی ای صورت نگرفته باشد. این حالت ممکن بود با تشخیص مجدد اشکالاتی تازه تر در فازهای جلوتر به کابوس بدتری برای توسعه دهندگان بدل شود.

از حیث ارتباط گروهی نیز این فرآیند چندان خوب عمل نمی کرد. معمولاً در این نوع توسعه تخصص ها فاز محور بودند. بنابراین یا هر فردی چند نقش را بر عهده داشت(تحلیل گر، برنامه نویس و ...) و در بیش از یک فاز حضور داشت و یا تک نقش بود و در برهه از زمان توسعه ایفای نقش می کرد. چنین حالتی ارتباطات ضعیفی را در بین گروه رقم می زد. از طرفی به این دلیل که افراد صرفاً در محدوده فاز خود در توسعه حاضر بودند، درک ضعیفی از پروژه به عنوان یک کل داشتند. که خود باعث ضعیف بودن فعل و انفعالات فنی در بین فازها می شد و همچنین بعضاً مقدمات ایجاد ناهماهنگی هایی را در پروسه توسعه ایجاد می کرد.

همه مشکلات مذکور باعث هدر رفتن زمان می شوند. زمان در گذر است و تیمی که نتواند زمان توسعه را با زمان همگام سازد محکوم به شکست است. برای برخی پروژه ها زمان عاملی است حیاتی. اگر این گونه پروژه ها به موقع تحویل داده نشود دیگر قابل استفاده نیستند. زمان همواره در پروژه ها ارتباط مستقیمی با قیمت تمام شده محصول دارد. هر کدام از اعضای تیم به ازای زمانی که برای تیم تخصیص می دهند درآمد دریافت می کنند. تصور کنید که زمان پیش بینی شده یک پروژه نوعی یک سال بوده اما در نهایت یک سال به طول بیانجامد. این بدین معنی است که اعضای تیم دو برابر هزینه پیش بینی شده حقوق دریافت نموده اند.

برخی قرارداد ها به گونه ای تنظیم می شوند که عدم تحویل به موقع نرم افزار هزینه ای را بابت پرداخت غرامت به تیم تحمیل می کند. در برخی اوقات نیز خارج شدن پروژه از زمان تعیین شده باعث شکست کل پروژه و تحمیل ضرر به مراتب بیشتری خواهد شد.در فرآیند آبشاری به علت عدم ذخیره سازی مناسب زمان قیمت تمام شده نرم افزارها بسیار بالا بود. این در حالی است که کیفیت نرم افزار تمام شده نسبت به قیمت بالای آن به هیچ وجه توجیه پذیر نبود.

فرآیند آبشاری فرآیندی است قدیمی و تقریبا منسوخ شده. متاسفانه همچنان نیز تیم هایی وجود دارند که از این فرآیند استفاده کنند. شاید دلیل آن سادگی اجرا و آموزش این فرآیند باشد. اغلب این تیم ها را تیم های تازه کار تشکیل می دهند که از تجربه کافی در توسعه بی بهره هستند.

فرآیند آبشاری علارغم مشکلاتی که دارد گاهی مفید و مناسب واقع می شود. این فرآیند بیشتر برای تیم ها و پروژه های کوچک اثر گذار است اما تاریخ اثبات کرده که پروژه های بزرگی توسط تیم های بزرگی به کمک این فرآیند به صورت موفق اجرا شدند. به عنوان یک قانون کلی می توان گفت: اگر دامنه کاربرد شناخته شده باشد، نیازمندی های ثابتی وجود داشته باشد و بتوان از آنها به مشخصات رسمی نسبتاً جامعی دست یافت با بودجه و زمان مناسب (معمولا زمان و بودجه واقعی بیش آن چه برآورد می شود در نظر گرفته می شود) می توان پروژه ای موفق را با این فرآیند(آبشاری) تجربه نمود(شکل 5-1).

شکل 4-1: شروط استفاده از فرآیند آبشاری

فرآیند تکراری افزایشی

توسعه نرم افزار پیچیده و پر ابهام است و برای آن روند مشخص و صریحی وجود ندارد. توسعه نرم افزار به معنای واقعی کلمه، *توسعه* می باشد. به این معنی که ابتدا نسخه کوچکی از برنامه ساخته می شود و به مرور بسط پیدا می کند تا کامل شود. سال هاست که دو کلمه تکرار و افزایش در فرهنگ توسعه دهندگان نرم افزار نهادینه شده است. یک فاز در توسعه نرم افزار صرفاً، یک بار صورت نمی پذیرد. بلکه هر فاز دارای تکراری پریودیک و منظمی است. یکی از علل این اجرای تکراری توسعه، می تواند عدم شناخت کافی به آن چه که باید ساخته شود باشد. به بیان ساده تر نه مشتری و نه توسعه دهنده اطلاعات جزئی و کاملی در لحظه، از نرم افزاری که باید ساخته شود ندارند. این عدم اطلاعات ناشی از دو دلیل است: نرم افزار محصولی است انتزاعی و برای عموم افراد ناملموس است و به سختی درک می شود(گرچه برخی مواقع برای خود توسعه دهندگان نیز این چنین است). دلیل دوم می تواند جدید و ناشناخته بود یک نرم افزار نوعی باشد. هرچه این شناخت بیشتر باشد و پیچیدگی توسعه بیشتر می شود.

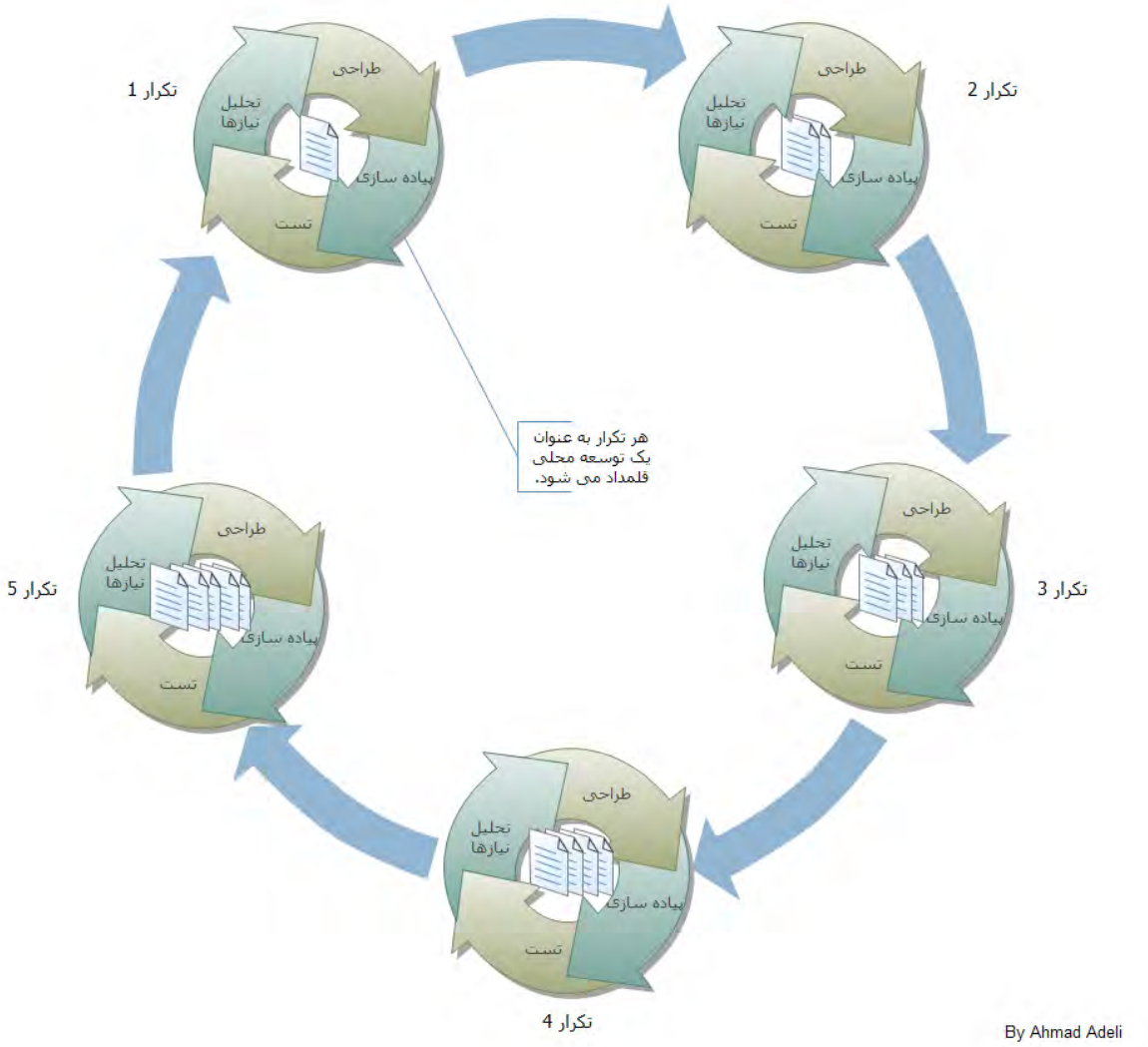
در اغلب مواقع به علت طبیعت تکثیر پذیر نرم افزار، پروژه هایی که شروع می شوند پروژه های جدید می باشند. سوالی در اینجا مطرح می شود: از آنجایی که اطلاعات کافی در رابطه با نرم افزار مذکور از ابتدا وجود ندارد(یاحداقل به صورت کاملی وجود ندارد)، شناخت روی نرم افزار چگونه باید صورت گیرد؟ پاسخ آن در یک کلمه

خلاصه می شود، *تکرار*. اگر به پروسه کشفیات علمی نگاهی بیاندازیم در خواهیم یافت که کشفیات یکباره و به صورت یکجا صورت نگرفته اند بلکه به مرور و بر پایه کشفیات قبلی به دست آمده اند و در هر برهه از زمان قسمتی از آن پیش رفته است. شناخت در توسعه یک نرم افزار نیز به مانند یک تحقیق علمی است و با تکرار به دست می آید. اما این موضوع به این معنی نیست که از آنچه در شرف ساخت آن هستیم شناختی نداشته باشیم؛ اطلاعات ابتدای پروژه اغلب کلی است و برای ادامه توسعه نیاز به اطلاعاتی با جزئیات بیشتر داریم.

نتیجه هر بار تکرار، ساختن بخشی جدیدی از نرم افزار است. این نتیجه از مفهومی پرده بر می دارد که به آن *افزایش* می گوییم. افزایش به این معنی است که بر اساس تکرارها با هر گام توسعه، در راستای رسیدن به کمال مطلوب نرم افزار (آنچه که در نهایت نرم افزار در حال توسعه باید به آن تبدیل شود) قابلیت جدیدی به نرم افزار اضافه می شود(شکل 6-1).

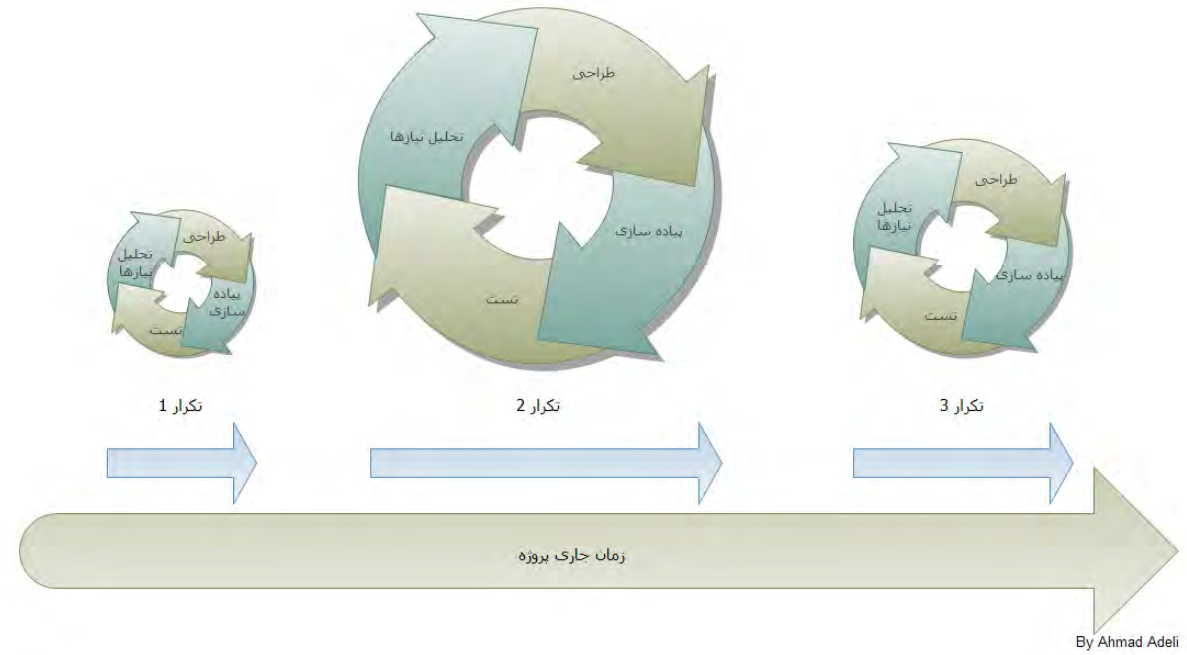
شکل 6-1: پروسه تکراری – افزایشی

در دهه 80 میلادی پس از اثبات ناکارآمد بودن فرآیند ترتیبی(آبشاری) میل بیشتری به مدل توسعه تکراری-افزایشی صورت گرفت. در زمان مذکور مدل های فرآیند زیادی بر اساس مدل تکراری-افزایشی ظهور کردند. قدرت و محبوبیت این مدل از آن جایی است که توسعه کلان را به صورت توسعه محلی تبدیل می کند. توسعه محلی به این معنی است هر تکرار مانند توسعه یک نرم افزار کوچک می شود. این تکرارها آنقدر ادامه می یابد تا نرم افزار به توسعه مناسبی رسد. تعداد دقیق تکرارها برای هر پروژه چندان مشخص نیست. تعداد تکرارها به عواملی چون دامنه کاربرد و مهارت اعضای تیم بستگی دارد. عدم توانایی در تعیین دقیق تعداد تکرارها دال بر عدم توانایی در تخمین آنها نیست. تخمین تعداد تکرارها به دیدگاهی تخمین کل زمان پروژه می باشد(شکل 6-7).

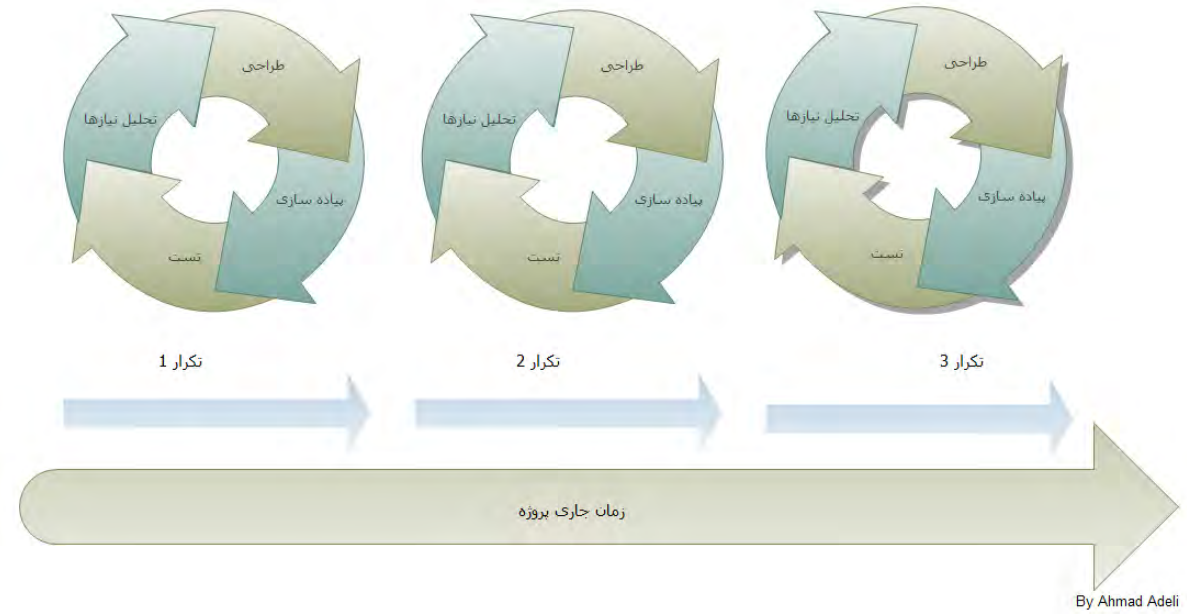


شکل 6-1: فرآیند توسعه تکراری – افزایشی

در این بین مفهوم دیگری وجود به نام زمان تکرارها، که به زمان مورد نیاز برای انجام هر تکرار اطلاق می شود. در برخی از گروه ها اعتقاد بر این است که باید تعداد تکرارها ثابت نگه داشته شود.بدیهی است در چنین حالتی اگر به علت وقایع غیر قابل پیش بینی زمان پروژه بیشتر شود، این زمان تکرار است که باید افزایش یابد. اما برخی دیگر از گروه ها زمان تکرارها ثابت نگه می دارند و از آن طرف تعداد تکرار ها را متغیر می کنند(شکل 8-1).



شکل 8-1 الف: تکرار های متغییر



شکل 8-1 ب: تکرار های ثابت

در هر مدل های فرآیند صرف نظر از نوع آنها تعدادی فاز ثابت قرار وجود دارد. این فازها درون تکرار ها واقع شدند اما نه در همه آنها. در مطالعه مدل های فرآیند مفهومی وجود دارد به نام فرکانس فازها. فرکانس فازها یعنی هر فاز در چند تکرار وجود خواهد داشت(تعداد تکرار فازها در توسعه نرم افزار). ناگفته واضح است که در هر تکرار تعداد یکسانی فاز وجود ندارد. به عنوان مثال تحلیل مشخصات در ابتدای پروژه بخش وسیعی از یک تکرار را به خود اختصاص می دهد و این در حالی است که در این مرحله فازی چون تست و اعتبار سنجی اصلا وجود ندارد.از آن سو در انتهای پروژه تحلیل مشخصات به کمترین میزان خود می رسد ولی فاز تست و اعتبار سنجی یکی پر کارترین فازها بدل می شود(شکل9-1).

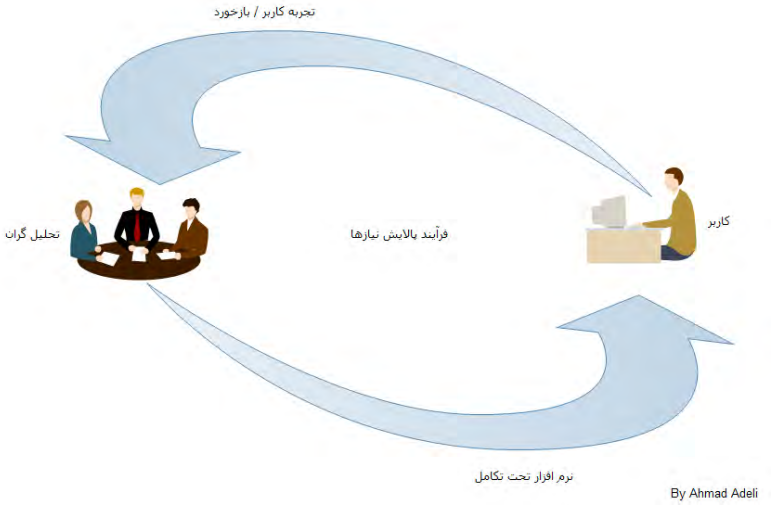
شکل 9-1: فرکانس نوعی فازها در توسعه بر اساس تکرار

بنابرآنچه گفته شد بسیار مهم است که در برنامه ریزی پروژه برای برآورد هزینه و زمان تعیین شود که چه فازی در چه مرحله قرار دارد. این مهم در فاکتور فرکانس فازها نیز می تواند مفید واقع شود. به عنوام مثال اگر تشخیص داده شود که یک فاز داری فرکانس ثابتی است، هزینه و نیروی انسانی ثابتی نیز به آن تخصیص داده می شود. و یا اگر تشخیص داده شود که فرکانس یک فاز در انتهای پروژه با افزایش همراه است، پیش از رسیدن به انتهای پروژه برای آن نیروی انسانی لازم استخدام می شود.

در برخی پروژه ها در میانه پروژه شناخت تیم از نرم افزار در حال توسعه به کمال می رسد و همه مشخصات پروژه کشف می شود. این وسوسه ممکن است در اعضای تیم به وجود آید که فراتر از حد یک فاز در یک تکرار بر روی آن کار شود. این مسئله می تواند بسیار بغرنج شود و تیم را وارد ریسکی بزرگی نماید. هیچ گاه به درستی نمی توان فهمید درکی که تاکنون از نرم افزار به دست آمده درک درستی است یا خیر! گاهی تصور می شود که مشخصات استخراج شده برای نرم افزار درست هستند. بر اساس همان اطلاعات به ظاهر درست، نرم افزار ساخته می شود. در نهایت در پایان یا با کمی خوش شانسی در میانه پروژه تیم متوجه اشتباه خود خواهد شد. در این صورت دو راه برای انتخاب وجود دارد: شکست یا دوباره کاری. به بیانی دیگر گاهی بر اساس همین شناختِ کامل شده و وسوسه متعاقب آن، یک تکرار بسیار به طول می انجامد و عملاً تبدیل به یک مدل آبشاری در دل یک مدل تکراری-افزایشی می شود.

کلید توسعه تکراری انجام کارها در اندازه کم و مناسب می باشد به نحوی که همه فازها در یک تکرار دخالت داده شوند. به عنوان مثال اگر که در یک فاز نیازی تشخیص داده شد باید در همان تکرار در صورت توان تحلیل و مدل سازی شود. گرچه این حضور همه فازها برای یک آیتم در یک تکرار ضروری نمی باشد و می توان مثلا تست واعتبار سنجی در تکرار دیگری به انجام رسانید. پس نیازی به پیاده سازی همه نیازهای تشخیص داده شده در یک تکرار نیست و کافی در هر بار تکرار صرفاً بخشی از نرم افزار را کامل کرد. اما علت چنین عملی چیست و چرا باید این گونه عمل کرد؟ پاسخ تجربه کابر است. توجه شود که بخش بندی کردن توسعه به تکرارها صرفاً برای شناخت و کشف مشخصات توسط تیم توسعه نیست. تجربه کاربر دستآورد مهمی از این بخش بندی می باشد. هر بار که قسمتی از نرم افزار تکمیل می شود سعی بر آن است که کاربر با آن کار کرده و کسب تجربه نماید. کاربران عموماً متخصص کامپیوتر نیستند و نمی تواند به تیم بگویند که دقیقاً چه می

خواهند. بر همین اساس است که همیشه از واژه کشف مشخصات و نیازها استفاده می شود. آنچه در ذهن کاربر است (کاربر در این جا همان مشتری سفارش دهنده نرم افزار است) برای تیم توسعه یک معماست که باید کشف شود. تیم باید درک کند که کاربر دقیقاً چه می خواهد و چه انتظاراتی از نرم افزار نهایی داردکه این فهم فقط از تجربه کاربر با نرم افزار درحال ساخت میسر می شود. هر بار که کاربران با نرم افزار در حال ساخت(در واقع بخش هایی از آن که ساخت شده است) کار می کنند درک تازه از آنچه می خواهند پیدا می کنند. تیم بر اساس این درک کاربر از نیازهایش نیازمندی هایی اصلاح و نیازمندی جدیدی به نرم افزار اضافه می کند. این عمل پالایش نیازها نام دارد. نیازها در هر بار تکرار پالایش می شوند تا در نهایت به مناسب ترین مشخصات بدل شوند. به عبارتی بهتر شناخت تیم توسعه از نرم افزاری که باید ساخته شود و تجربه کاربری دو کفه ترازو هستند که باید توازنی بین آنها ایجاد کرد(شکل 10-1).



شکل 10-1: فرآیند پالایش نیازها

گاهی نیازهایی وجود دارد که برای کاربر دارای ضرب العجل هستند. اما متاسفانه با وجود حساسیت بالایی که دارند به صورت ضمنی بیان می شوند که نتیجه آن انتخاب اولویت پایین برای آن نیازها می باشد. با تجربه کردن نرم افزار در دست ساخت، پس از هر تکرار توسط کاربر، به موقع می توان این گونه نا هماهنگی ها را(که به طبیعت توسعه نرم افزار باز می گردد) تشخیص داد.

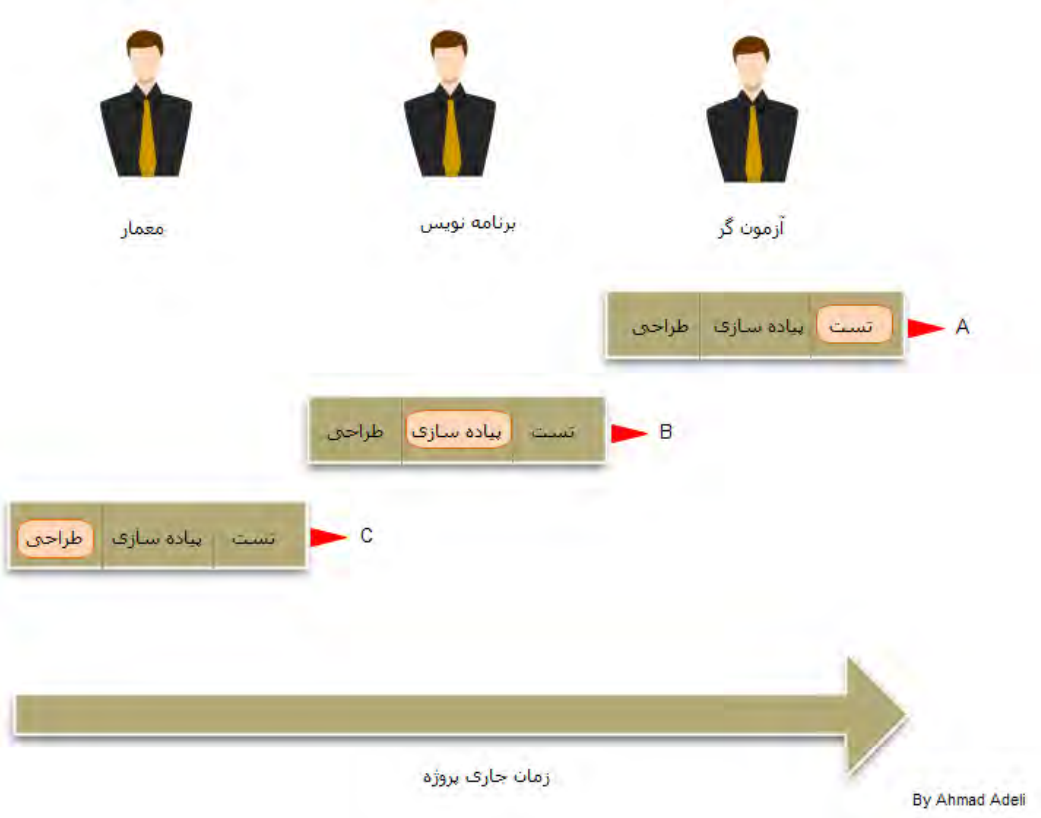
هنمانطور که گفته شد تکرارها تنها برای شناخت نیازها استفاده نمی شوند از این رو پس از کسب این شناخت متوقف نخواهند شد. گاهی فشارهایی از سوی بازار، تجارت و کارفرما تیم توسعه را مجبور به تحویل نسخه های کوچک اما به روزی از نرم افزار می کند. تیم بخشی از نیازمندی هایی که می تواند تحویل دهد اخذ کرده و نرم افزار به روز شده ای(نرم افزار به علاوه قابلیت های جدید) را تحویل می دهد.

کار بر روی پروژه های بلند مدت ذاتاً امری است طاقت فرسا. بدیهی است افرادی که در کوتاه مدت نتیجه کارشان را دریافت می کنند نسبت به افرادی که این نتیجه را در مدت زمان بیشتری دریافت می کنند دارای روحیه مضاعف تری هستند. روحیه بالای افراد یک تیم با کیفیت کار آن ها رابطه ای مستقیم دارد. این موضوع را نیز می توان یکی از مزایای توسعه بر اساس تکرار برشمرد.

مدیریت نیروی انسانی در این مدل فرآیند(تکراری-افزایشی) از نظر تفکیک وظایف تفاوت چندانی با مدل آبشاری ندارد(البته مدیریت منابع انسانی در گروه ها و سازمان های متفاوت، مختلف است). اغلب توسعه دهندگان به صورت تک وظیفه ای کار می کنند. اما مزیت این روش در اینجاست که به علت قرار داشتن فازهای مختلف در هر تکرار، اغلب توسعه دهندگان همراه در طول توسعه حضور دارند که این خود به انسجام گروه و تمرکز بیشتر اعضا بر توسعه کمک شایانی می کند.

اغلب در هر تکرار روی چند نیازمندی کار می شود که هر کدام از آنها ممکن است فازهای توسعه مجزایی را طلب کنند. برای استفاده بهینه تر از منابع انسانی بهتر است در آن واحد چند فاز همزمان به صورت پایپ لاین اجرا شوند به شکلی که هر فاز روی یک نیاز مندی کار کند. به عنوان مثال حالتی فرض شود که تستر(آزمون گر) روی تست کردن یک قسمت از نرم افزار وقت صرف کند که مربوط به نیازمندی A باشد. برنامه نویسی ممکن

است در حال پیاده سازی بخش B نرم افزار باشد. و همچنین در این بین معمار نرم افزار در حال بسط معماری بر اساس نیازمندی C باشد.(توجه شود که هر بخش یا وظیفه معادل یک فاز است). در مرحله بعد هر کدام از اعضا(نقش ها) متقابلاً روی نیازمندی های یکدیگر کار خواهند کرد.(مثلاً روی بخش B و برنامه نویس روی بخش C کار کند). بدین ترتیب هیچ گاه هیچ کدام از اعضا بیکار نخواهند ماند(و یا موقتاً از گروه خارج نخواهند شد) که به خودی خود باعث بهینه سازی هزینه ها می شود(شکل 11-1).



شکل 11-1: اجرای پایپ لاین توسعه

در مدل تکراری-افزایشی متدهای زیادی در جهان معرفی شدند که از جمله آنها می توان به متد مارپیچی، MSF و RUP اشاره کرد. هر کدام از این متدها معایب و مزایای خاص خود را دارند. برخی برای پروژه های کوچک مناسبند و برخی برای پروژه های سازمانی. برخی نیز مختص پلت فرم های خاصی هستند مانند متد توسعه MSF(این متد توسط شرکت مایکروسافت و برای توسعه سیستم های مبتنی بر محصولات این شرکت عرضه شده است). مدل فرآیند تکراری افزایشی بیش از 30 سال است که مورد استفاده قرار می گیرد و هم اکنون نیز بسیاری از سیستم ها و نرم افزارها بر اساس اسن فرآیند ساخته می شوند.

مدل فرآیند آجیل

از دهه 90 میلادی به بعد توسعه دهندگان مجدداً به فکر بهبود توسعه نرم افزار افتادند. در این دهه از مدل فرآیند تکراری-افزایشی به کرات استفاده می شد و متد هایی که بر اساس این مدل فرآیند بودند از محبوبیت بسیاری برخودار بودند(کما اینکه در حال حاضر نیز همچنان محبوب باقی ماندند).

همواره دغدغه اصلی توسعه دهندگان نرم افزار، ساخت نرم افزارهایی با کیفیت بیشتر بوده است. در آن دهه به صورت پراکنده متدهایی معرفی شدند که ریشه در تفکری واحد داشتند. توسعه نرم افزاری با کیفیت به علاوه ی اجتناب از پیچیدگی ها و تشکیل ارتباطاتی قوی هدف واحد این متدها بود. این متدها اغلب تفکرهای سنتی راجع به توسعه نرم افزار را در هم شکستند، گرچه که برخی از آنها که مربوط به طبیعت توسعه می باشند را تغییر نمی دهند به عنوان مثال تفکر تکراری-افزایشی که در بیشتر این متد ها(و شاید همه آنها) به چشم می خورد.

برای یک دهه متدهای مذکور به عنوان رقیب یکدیگر محسوب می شدند اما در سال 2001 بیانیه ای منتشر شد با عنوان بیانیه *آجیل* که هدف آن اتحاد این متد های متشابه بود. این بیانیه توسط تعدادی از سازندگان متدها

چون اسکرام، XP، Kanban، کریستال و ... نوشته شد. بیانیه آجیل متدهای هم خانواده ای را که در دهه 90 ظهر کردند را متحد کرده و در قالب چهار ارزش خلاصه می کند:

- افراد و تعاملات برتر از فرآیندها و ابزارها
- نرم افزار کار کننده برتر از مستند سازی جامع
- همکاری مشتری برتر از قرارداد معامله
- پاسخ به تغییرات برتر از پیروی از طرح

نکته: در حقیقت 8 ارزش وجود دارد که به دو دسته تقسیم شده است. در آجیل ارزش های سمت راست به ارزش های سمت چپ ترجیح داده می شوند اما به این معنی نیست که ارزش های سمت چپ نادیده گرفته خواهند شد.

اساس آجیل چهار ارزش فوق است. در بخش های قبلی راجع به دو مدل فرآیند آبشاری و تکراری-افزایشی توضیح داده شد. گفته شد که مدل تکراری-افزایشی نسبت به مدل آبشاری بهبود بسیاری یافته و کاراتر است. اما خود این مدل نیز مشکلات زیادی دارد. در اصل همین مشکلات بود که توسعه دهندگان را به فکر کار بر روی فرآیندهای بهتر انداخت. آنها به دنبال کارایی بهتر و کیفیت بالاتر توسعه نرم افزار بودند. از این رو به دنبال سدها و موانعی گشتند که در مدل های قبلی با آنها روبرو شده بودند.

یکی از بزرگترین مشکلاتی که در مدل های قبلی موجود بود عدم تطبیق پذیری فرآیند با گروه، شرایط و محیط توسعه بود. قوانین این فرآیندها اغلب تغییر ناپذیر بودند. فرآیند دارای اجرایی، خشک و نامنعطف بود و برای همه اعضای توسعه مناسب نبودند. همچنین قدرت مانور اعضای تیم نیز بسیار پایین بود. مدل های فرآیند در عمل راهنماهایی هستند برای توسعه بهتر. اگر یک مدل نتواند با تیم توسعه مطابقت داده شود این تیم توسعه است که باید خود را با مدل مطابقت دهد که همین موضوع می تواند ریشه بسیاری از مشکلات مدیریت منابع انسانی باشد. در مدل های پیشین تعاملات اعضا به شکل ضعیفی صورت می گرفت. اعضای تیم به زبان فرآیند با یکدیگر ارتباط بر قرار می کردند. به عبارتی بهتر توسعه دهندگان، فرآیند توسعه را به دست ماشین خودکاری به نام مدل فرآیند می سپارند تا توسعه نرم افزار را کنترل کند. گرچه گفته شد که توسعه دهنگان مایل هستند که فرآیند توسعه را به صورت روتین انجام دهند اما با توجه به این که توسعه نرم افزار امری است ذاتاً خلاق افراط در این امر نیز می تواند مشلات بیشتری را به تیم توسعه تحمیل کند. توازن بین توسعه خلاقانه و فرآیند روتین فقط از یک مدل فرآیند انعطاف پذیر بر خواهد آمد، توازنی که چندان در مدل های فرآیند قبلی رویت نشد. به عقیده بسیاری این توازن همان آجیل می باشد.

مدل فرآیند آجیل قابلیت تطبیق بالایی دارد. قابلیت تطبیق در بسیاری از متدهای پیاده سازی این مدل وجود دارد. از این رو برای هر پروژه ای با هر سطحی قابل استفاده است چرا که در شرایط مناسب می توان آن را سفارشی نمود.

CASE در آجیل

سال هاست که توسعه دهندگان از CASE(مهندسی نرم افزار به کمک کامپیوتر) برای تسهیل توسعه کمک می گیرند. ابزارها به توسعه دهندگان اجازه مانور بیشتر را می دهند، کارها را ساده تر می کنند، مدیریت را ارتقاع می دهند و به توسعه سرعت می بخشند. اگر مجدداً به اصل اول آجیل دقت شود مشاهده می شود که ابزارها با وجود مفید بودن در سمت چپ قرار گرفتند(افراد و تعاملات به آنها ترجیح داده شده است). علت مجدداً به فرآیندها باز می گردد. ابزارها ذاتاً بر اساس فرآیندها طراحی شده اند و قوانین آن ها را به ارث می برند. ممکن است حالتی پیش بیاید که ابزار از تغییر فرآیندی که تیم توسعه برای تطبیق پذیری با شرایط می دهد حمایت نکند(در اینجا فرض می شود که ابزار جایگزینی وجود ندارد و یا استفاده از آن امکان پذیر نیست). در این صورت این تیم توسعه است که مجبور با انطباق پذیری با ابزار موجود می شود. ارزش اول صراحتاً این حالت را نقض می کند. چرا که نه تنها دست تیم را برای یک توسعه ایده آل می بندد بلکه احتمالاً موجب خارج شدن فرآیند از لیست کاندیدای تیم خواهد شد. قرار گرفتن تیم توسعه در حصار فرآیندها و حتی ابزارهایی خاص و غیر قابل انعطاف کیفیت فرآیند را به شدت تنزل خواهد داد. این نکته مجدداً یادآوری می شود که افزایش کیفیت فرآیند برابر است با افزایش کیفیت محصول است.

نکته: حالات بحث شده در این بخش فقط یک حالت از حالات مختلف استنتاجی را بررسی می کند و هدف، شفافیت بهتر مطالب است.

مستند سازی در آجیل

مستند سازی در هر فرآیندی یکی از مصنوعات مهم هر پروژه می باشد. هر چه که در توسعه اتفاق می افتد باید به نحوی ثبت شود. آنچه به عنوان مستند سازی ثبت می شود آیتم های ساخته شده، چرایی ساخت و چگونگی ساخت آن ها است. حتی در برخی موارد نام توسعه دهنده ای که آیتم را می سازد نیز در مستند سازی ذکر می شود. همچنین مشخصات فنی(رسمی) را نیز باید به موارد بالا اضافه کرد. به طور خلاصه مستند سازی با این تفکر نوشته می شود اگر توسعه دهنده ای مایل بود مجدداً آیتمی را بسازد که قبلاً ساخته شده یا بخواهد آیتم ساخته شده را تغییر دهد بتواند با رجوع به مستند سازی به راحتی به جزئیات آیتم مذکور اشراف پیدا کند. مستند سازی برای برخی فرآیندها آیتمی حیاتی است و زمان زیادی به آن اختصاص داده می شود. اما صرف این همه وقت و انرژی به مستند سازی برای چنین کاری چندان توجیهه پذیر نیست. دلایل زیاد دیگری برای انجام مستند سازی وجود دارد. مهمترین دلیل آن قابلیت ردگیری است! به جرأت می توان گفت تبدیل کردن نیازمندی ها به یک نرم افزار مشکل ترین کاری است که می توان در توسعه نرم افزار صورت داد. سوالی که همیشه ذهن توسعه دهندگان را به خود مشغول کرده است این است که چگونه می توان تضمین کرد که یک نیاز به طور واقعی پیاده سازی است یا خیر؟ پاسخ این سوال ردگیری (Tracking) می باشد.

ردگیری به این معناست که بتوان یک نیازمندی را به صورت سلسله وقایع از ابتدا(هنگامی که وارد توسعه می شود) تا انتها(هنگامی مورد استفاده کاربر قرار می گیرد)دنبال نمود. به عبارتی دیگر آماده سازی یک نیازمندی برای ردگیری پاسخ به این سوالات است:

- نیازمندی مطرح شده چیست؟
- چگونه کشف شده است؟
- چگونه توسط کاربر نهایی تایید می شود؟
- چگونه در معماری کلی نرم افزار جای می گیرید؟
- چه ساختمان داده هایی در ارتباط با آن وجود دارد؟
- از چه الگوریتمی برای پیاده سازی آن استفاده شده است؟
- پیاده سازی آن چگونه است؟
- چه تست هایی برای تخمین عملکرد آن وجود دارد؟
- با چه رابط کاربری هایی در نرم افزار در ارتباط است؟

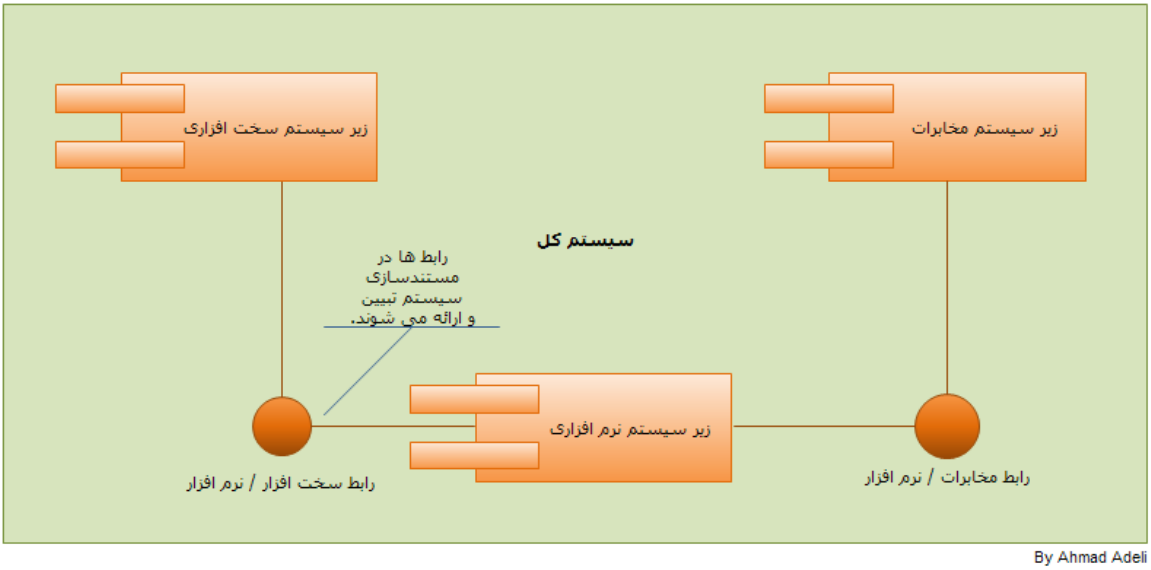
یک مستند سازی جامع (پاسخ مناسب به سوالات مطرح شده) می تواند قابلیت درگیری را تضمین نماید. قابلیت ردگیری به خصوص برای سیستم های حیاتی-ایمنی و حیاتی-امنیتی بسیار پر اهمیت است. بدون مستنداتی جامع، قابلیت ردگیری عملاً غیر ممکن است. این قابلیت به توسعه دهندگان کمک می کند تا بدانند که چه نیاز هایی پیاده سازی شدند، پیاده سازی کدام نیازها صحیح نبود است و نیازهایی که باید پیاده سازی شوند کدامند. اگر مستند به اندازه کافی کامل باشند تیم توسعه می تواند هر مشکلی را با ردگیری دو سویه ریشه یابی کند. ردگیری دو سویه به این معنی است که می توان هم از یک نیاز به یک بخش پیاده سازی شده رسید و هم بالعکس از یک بخش پیاده سازی شده شروع کرد تا به نیاز رسید. حرکت بالعکس برای مواقعی مناسب است که به عنوان مثال کارفرما می خواهد بداند دلیل ساخت بخشی از نرم افزار چیست(ممکن است به عقیده او توجیهه مالی نداشته باشد)؟ در چنین حالتی می توان با استفاده از قابلیت ردگیری برای هر قسمت از نرم افزار نیاز معادلش را استخراج کرد.

فرض کنید توسعه یک نرم افزار نوعی به اتمام رسیده و پروژه وارد فاز نگهداری شده است. پس از مدت ها استفاده نیاز است که بخش هایی از نرم افزار تغییر کند. در چنین حالتی است که وجود یک مستند سازی قوی به شدت نیاز می شود. نرم افزار های تجاری ای که بیش از سورس آن ها 10000 خط کد دارد آن قدر پیچیده هستند که گاهی می توانند خود توسعه دهندگان شان را نیز به مخاطره بیاندازند. از طرفی در پروژه های بزرگ و دراز مدت ممکن است توسعه دهندگانی از خارج از گروه جایگزینی اعضای درون گروه شوند. کلید همگام سازی این توسعه دهندگان جدید با گروه، فقط می تواند مستند سازی آن سیستم باشد.

برخی از تیم های توسعه از سند مشخصات رسمی (بخشی از مستند سازی) به عنوان قرارداد با کارفرما استفاده می کنند. البته این پروژه، پروژه هایی هستند که دارای مشخصات شناخته شده ای می باشند. اگر قراردادی به این شکل تنظیم شود باید بخش اعظم مشخصات رسمی قبل از توسعه مهیا شود.

بعضی از نرم افزار ها شکل مستقلی ندارند و برای تعبیه در دیگر سیستم ها نوشته می شوند. به عبارت دیگر این گونه نرم افزار ها زیر سیستم هایی برای سیستم های بزرگتر هستند. به این زیر سیستم، زیر سیستم

های دیگری نیز در سیستم اصلی وجود دارند. نمونه از این زیر سیستم ها می تواند سخت افزاری باشد که نرم افزاری را که توسعه می دهیم اجرا می کند و یا زیر سیستم مخابراتی که به نرم افزار توسعه یافته قدرت برقراری ارتباط را می بخشد. طبق تعریف سیستم، زیر سیستم ها با یکدیگر همکاری می کنند تا متحداً هدف اصلی سیستم بزرگ تر را برآورده سازند. بنابراین نرم افزار توسعه یافته شده نیز بر اساس این همکاری و به عنوان یک زیر سیستم باید با دیگر زیر سیستم ها ارتباط برقرار کند. این ارتباط توسط رابط هایی تعریف شده برقرار می شود. هر زیر سیستمی که بخواهد با دیگری ارتباط برقرار کند مجبور است از رابط آن زیر سیستم شناخت حاصل کند و گاهی حتی باید بداند که یک زیر سیستم چگونه ساخته می شود. این اطلاعات رابط ها و هر نوع اطلاعات دیگری راجع به زیر سیستم ها بصورت یکپارچه در مستند سازی سیستم درج می شود. به عنوان مثال زیر سیستم سخت افزاری می تواند از طریق مستندات اطلاع رسانی لازم را برای مطابقت با مشخصات فنی نرم افزاراها انجام دهد(مثلاً با توجه به پیچیدگی زمانی الگوریتم ها پردازنده ای مناسب پیشنهاد دهد یا بر اساس قطعات قابل بارگذاری در نرم افزارها میزان حافظه مناسب را برآورد کند) و همچنین متخصصین زیر سیستم مخابراتی می توانند تکنولوژی ارتباطی همخوان با نرم افزار را به عنوان رابط تعیین کنند(شکل 1-12).



شکل 12-1: همکاری زیر سیستم ها در سیستمی بزرگتر

یکی دیگر از موارد استفاده مستند سازی ها، برنامه ریزی و تحلیل جریان کار می باشد. در ابتدای پروژه برنامه ریزی و زمان بندی بر اساس تخمین صورت می گیرد. اما با پیشرفت توسعه به چیزی ارزنده تر از تخمین دست خواهیم یافت به نام سابقه توسعه. سابقه توسعه همه چیزی است که برنامه ریزی به آن نیاز دارد اما به دلیل اینکه در ابتدای توسعه از وجود آن بی بهره ایم به ناچار به تخمین روی می آوریم. اما در میانه راه با استفاده از سابقه توسعه می توانیم برآوردهای نسبتاً دقیق تری در برنامه ریزی و به خصوص زمان بندی انجام دهیم.

از طریق مستند سازی می توان تحلیل جریان کار را نیز صورت داد. با استفاده از تحلیل جریان کار می توان توسعه جاری را با استانداردهای کیفیتی مقایسه کرد و در صورت لزوم اقدام به بهبودهایی برای ارتقاع کیفیت آن نمود.(ازجمله این استانداردهای کیفیتی می توان به استاندارد کیفیت فرآیند و استاندارد کیفیت محصول اشاره کرد).

آنچه تاکنون گفته شده تنها قسمتی از مزایای وجود مستند سازی در فرآیند توسعه است. با این وجود مستند سازی نیز یکی از آیتم های توسعه به شمار می رود. این آیتم مانند دیگر آیتم های توسعه از زمان توسعه تغذیه می کند. برای تیم های توسعه زمان *سرمایه* است. در نتیجه می توان به این نکته اشاره کرد که مستند سازی هزینه بر است. تجربه نشان داده علارغم فواید بسیاری که مستند سازی می تواند ایجاد کند برای ضرر مالی زیادی را نیز می تواند در اثر افراط در انجام آن به تیم توسعه تحمیل کند که این موضوع خواسته یا نا خواسته باعث تاثیر در افزایش قیمت محصول نهایی خواهد شد. البته باید این نکته یادآوری شود که مستند سازی جزو

مصنوعات پروژه می باشد و در دراز مدت سرمایه تخصیص داده شده به آن باز می گردد. اما از طرفی این بازگشت سرمایه برای مصرف کننده محسوس نیست و ارتباط مستقیمی با او برقرار نمی کند.

مستند سازی باعث کند شدن سرعت توسعه می شود. این موضوع برای تیم های توسعه ای که از سوی کارفرما و یا کاربران با ضرب العجل مواجهه هستند ممکن است ایجاد مخاطره کند. برخی سازمان ها سیاست تعریف شده خاصی در رابطه با مستند سازی دارند که این خود باعث تاثیر منفی در زمان توسعه خواهد شد. علاوه براین ها با تغییرات نرم افزار مستندات نیز متعاقباً باید تغییر کند در صورت نیز پاسخ سریع به تغییرات تیم را با چالش هایی بدتری روبرو می کند.

اصلی ترین مشکل مستندات سنگین بودن آن ها است که پس از هفته ها توسعه خروجی پروژه انبوهی مستند به علاوه نرم افزاری ناقص است که قادر به سرویس دهی نیست. این در حالی است که بهانه شروع یک پروژه ساخت نرم افزاری است که سرویس هایی را ارائه کند. دلیل این مشکل این است که تیم وقت و انرژی خود را به جای کار روی نرم افزار، صرف مستند سازی نرم افزاری می کند که به درستی کار نمی کند. البته این گونه نیست که نرم افزار عملاً کار نکند، تنها کارنکردن یک سروریس نیز می تواند کاربرد پذیری نرم افزار را مخاطره بیاندازد.

توزان کلید حل این مشکل است. مستند سازی به دور از تفریط و افراط باید به صورت متوازنی انجام شود. اما به نظر توسعه دهندگان متدهای خانواده آجیل کمی کفه ترازو نرم افزار کار کننده نسبت به مستند سازی جامع بیشتر سنگینی می کند.

نکته: متاسفانه در برخی ترجمه های فارسی بیانه آجیل برای واژه لاتین Working معادل *کارا* استفاده شده است در حالی که معادل فارسی این کلمه عبارت *کارکننده* می باشد (توجه شود که کارا معادل کلماتی چون Efficiency و یا Proficiency می باشد). کارکننده اصطلاحی است در توسعه نرم افزار که در جهت ارجاع به صفت کاربرد پذیری استفاده می شود.(نرم افزاری صفت کاربرد پذیری یدک می کشد که به معنای واقعی در عمل کار کند و سرویس هایی را که برای آن طراحی شده ارائه دهد. این صفت اصلی ترین صفت اندازه گیری یک نرم افزار توسعه یافته از نظر درجه کیفیت می باشد).

نیازمندی ها در آجیل

در اکثر صنایع هنگامی که کارفرمایی بخواهد پروژه ای را انجام دهد با شرکت یا گروهی قرارداد می بندد. ابتدا نشست هایی انجام می شود تا پروژه امکان سنجی شود و توافقات لازم صورت پذیرد. پس از آن قراردادی امضاء می شود که در آن مشخص شده کارفرما چه خروجی ای انتظار دارد. معمولا پس از این مرحله تیم کار را شروع می کند و پروژه با چند نشست دیگر و شاید نظارت کارفرما ادامه می یابد. متاسفانه سال هاست که شرکت های بسیاری در صنعت نرم افزار از فرآیند یاد شده پیروی کرده و توسعه نرم افزار می پردازند. نرم افزار یک ساختمان یا یک قطعه صنعتی نیست که صرفاً بر اساس یک پیمان ساخته شود. این موضوع به ویژه برای نرم افزارهای جدیدی صحت دارد که نه کارفرما و نه تیم توسعه هیچ ایده دقیقی برای چگونگی ساخت آن ها ندارند. از این رو برای ساخت به چیزی فراتر از یک قرارداد صرف نیاز است و چیزی جزء همکاری مستقیم مشتری(یا کارفرما) نخواهد بود. (البته استثنائاتی نیز در این بین وجود دراد. به عنوان مثال اگر نرم افزار در دست توسعه، نسخه متفاوتی از یک نرم افزار قدیمی باشد توسعه بر اساس قرارداد می تواند مناسب باشد.

اغلب دلایلی که سبب ضعف در درک نیازمندی های نرم افزار می شود ناشی از عدم همکاری مشتری است. در اصل سوم از بیانیه آجیل قرارداد معامله در سمت چپ قرار داده شده است. این ارزش به این نکته اشاره می کند که همکاری مشتری به قرارداد معامله ارجحیت دارد. در توسعه نرم افزار بهتر است که مشتری ارتباط مستقیمی با تیم توسعه بر قرار کند. این ارتباط صرفاً مبتنی بر نظارت نیست و اغلب بر اساس همکاری است. چنین ارتباطی را می توان به دو بخش تقسیم کرد: در طول یک تکرار(اغلب متدهای خانواده آجیل بر این اساس کار می کنند) و در پایان یک تکرار. حضور مشتری در پایان یک تکرار از طریق جلسه ای با ارائه بخش های اضافه شده صورت می گیرد از بازخوردهای تجربه کاری مشتری با نرم افزار (به خصوص قسمت های جدید) استفاده شود. اما اگر ارتباط مشتری در طول توسعه انجام شود باید حضور پیوسته و فعال مشتری به نحوی تضمین شود. مزیت این روش آن است که به دلیل حضور همیشگی مشتری احتمال کج روی در توسعه به حداقل خواهد رسید(درباره این روش و چگونگی انجام آن در فصول آینده مفصلاً توضیح داده خواهد شد).

برای هر پروژه ای در هر صنعتی برنامه ریزی لازم است. در اغلب صنایع (و نه در همه آنها) برنامه ریزی حالتی ایستا دارد. در این نوع برنامه ریزی اغلب سعی می شود بر اساس برنامه از پیش تعیین شده عمل شود. در

مقام مقایسه صنعت نرم افزار با صنایع دیکر باز هم تاکید می شود که این صنعت شباهت چندانی با دیگر صنایع نداشته و در حالاتی دقیقاً نقطه مقابل آن ها است. توسعه نرم افزار عکس حالت پیش برنامه ریزی ای پویا دارد. در متد های سنتی توسعه نرم افزار (که اغلب چندان موفق نبوده) با کپی برداری از شیوه برنامه ریزی صنایع غیر نرم افزاری از یک طرح کلی پیروی می نمودند. این شیوده برنامه ریزی در آن دوره به دو دلیل می توانست صورت پذیرد: یا خبر از هیچ تغییری در توسعه نرم افزار نبود! یا به تغییرات اعتنایی نمی شد. در طول توسعه نرم افزار نیازها دائماً تغییر می کنند و به عبارتی تغییرات عضو لاینفک توسعه نرم افزار هستند. اگر صرفاً تا انتهای توسعه بر اساس طرح اولیه عمل شود بدیهی است نتیجه غیر قابل استفاده خواهد شد. تغییری که باید اعمال شود گاهی نهان است و این وظیفه تیم توسعه است که تغییر را کشف و اعمال کند. برای بهتر شدن فرآیند توسعه باید ترجیحاتی در آن اعمال شود. در این مورد ترجیحی است که در ارزش چهارم از بیانیه آجیل این گونه بیان شده است: "پاسخ به تغییرات برتر از پیروی از طرح".

در بیانیه آجیل هیچ گاه بخش سمت چپ ارزش ها نقض نمی شوند. این ارزش ها به نوعی تقابل ارزش های متدهای ناکآرمد سنتی در برابر ارزش ترجیح داده شده در متدهای نوین هستند. تجربه ثابت کرده است که اگر این ترجیحات به شیوه مناسبی اعمال شوند کیفیت فرآیند توسعه به طرز شگفت آوری ارتقاء خواهد یافت.

نرم افزار همواره عضو لاینفک اغلب صنایع(و شاید بتوان گفت همه صنایع) بوده است. بزرگ ترین مشکلی که توسعه دهندگان همواره با آن روبرو بودند عدم درک درست صاحبان صنایع از صنعت نرم افزار است. آن ها محصولات نرم افزاری را مانند محصولات دیگر صنایع و فرآیند توسعه نرم افزار را مشابه دیگر فرآیندها تصور می کنند. از این رو بهتر است برای کاهش مشکلات ارتباطی ناشی از این درک ضعیف، پیش از توسعه نسبت به شناخت مشتریان از نرم افزار به عنوان محصولی انتزاعی اطمینان حاصل کرد و در صورت توان شناخت مناسبی را به آن ها القا نمود.

اصول دوازده گانه آجیل

علاوه بر چهار ارزش ذکر شده، در بیانیه آجیل از دوازده اصل در توسعه نرم افزار نیز پرده برداشته شده است. این دوازده اصل در واقع راهنماهایی برای محقق سازی ارزش ها است. در ادامه این دوازده بررسی خواهد شد.

1. بالاترین اولویت ما رضایت مشتری از طریق تحویل زودهنگام و پیوسته نرم افزار با ارزش است.

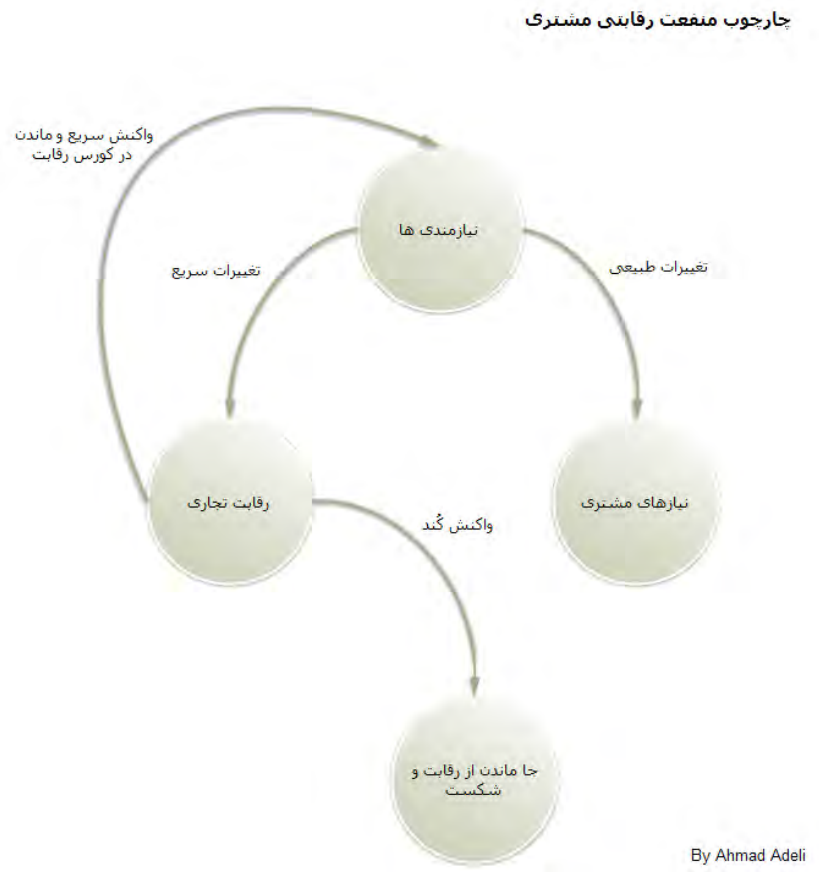
جمله فوق صرفاً یک راهنمایی ساده نیست بلکه می تواند کتاب راهنما باشد که چهار مفهوم کلیدی توسعه نرم افزار را شرح می دهد: رضایت مشتری، نرم افزار با ارزش، تحویل رودهنگام و تحویل پیوسته. پس از ناکامی متدهای خانواده ترتیبی مثل آبشاری (از نسل های اولیه متد های توسعه نرم افزار) یک مفهوم در قلب همه فرآیندهای بعدی نهادینه شد؛ توسعه بر اساس تکرار. متدهای نوین توسعه نیز بر همین اساس بنا شدند(این مفهوم در فصل های آتی به طور مفصل توضیح داده شده است در نتیجه از ذکر مکررات در این بخش خود داری می شود) . یکی دیگر از مفاهیمی که در اصل اول عنوان شد تحویل زودهنگام نرم افزار است. تحویل زود هنگام به این معنی است که بخشی از نرم افزار که در یک تکرار کامل شود باید آماده استفاده باشد. "زود هنگام" در شرایط مختلف تفسیر متفاوتی دارد و ممکن است به معنی هر هفته و هر ماه باشد. مفهوم مهمی دیگری که همراه زود هنگام معرفی شده مفهوم پیوسته است. اگر تکرارها پریودیک و منظم باشد تحویل نیز پیوسته و منظم صورت می گیرد.

گفته شد که پس از هر تکرار باید قسمتی از نرم افزار تحویل داده شود. قسمت های قابل تحویل قسمت هایی هستند که اصطلاحاً "با ارزش" نام دارند(با ارزش معادل واژه لاتین Valued می باشد). قسمت های با ارزش، به صورت یک کل، نرم افزار با ارزش را خواهند ساخت. نرم افزار با ارزش نرم افزاری است که از پس وظایف محوله به خوبی برآید، نیازهای کاربر را بر طرف سازد و همچنین سرویس های سودمندی را ارائه کند. به دیگر سخن نرم افزار صرفاً کامل شده را نمی توان نرم افزار با ارزش نامید بلکه نرم افزاری با ارزش است که سرویس های ارائه شده توسط آن صحت عملکرد داشته باشد حتی اگر این نرم افزار ناقص و در دست توسعه باشد. هیچ تضمینی وجود ندارد که مشخصات تکمیل شده یک نرم افزار همان هایی هستند نیاز کاربران را بر طرف می کنند. یک نرم افزار باید ارزش ایجاد کند. در اینجا تعریف به معنای برطرف کردن نیاز واقعی کاربرانی است که نرم افزار برای آن ها ساخته می شود. برخی افراد با ارزش بودن یک نرم افزار را به "با بها" بودن آن تعبیر می کنند که این تعبیر از دیدگاهی نیز می تواند درست باشد چرا که اگر نرم افزاری سودمند نباشد و نیازی را بر طرف نکند بهایی برای آن پرداخت نخواهد شد. نرم افزاری دارای بها است که رضایت مشتری را همراه خود به یدک بکشد.

2. استقبال از نیازمندی ها، حتی در انتهای توسعه. فرآیندهای آجیل از تغییرات برای منفعت رقابتی مشتری حمایت می کنند.

در انتهای جمله فوق ار عبارتی نامأنوس استفاده شده است: منفعت رقابتی مشتری(البته لازم به ذکر است این عبارت در ادبیات فنی ما چندان شناخته شده نیست). این عبارت در حقیقت مفهومی است که فرهنگ "همه چیز به سود مشتری" را انتقال می دهد.

در نظام های اقتصادی پیشرفته در جهت حمایت از حقوق مصرف کننده، تولید کنندگان در محیط هایی کاملاً رقابتی به سر می برند. رقابیت بین تولید کنندگان همواره به سود مشتری بوده است. در این نظام های اقتصادی قوانین با اولویت منافع مصرف کنندگان وضع می شود. به عنوان مثال اگر دو شرکت از تکنولوژی یکدیگر استفاده کنند به طوری که سطح رقابتی بین این دو شرکت کاهش یابد؛ مراجع قانونی برای جلوگیری از زائل شدن حقوق مصرف کننده از این همکاری ممانعت به عمل آورده و حتی ممکن است هر دو شرکت را مجبور به پرداخت غرامت کند. از طرف دیگر تولید کنندگان نیز برای جلب رضایت و توجه مصرف کننده، پیشرفت فنی خود و متعاقباً کسب سود بیشتر، این نوع مشتری مداری را در دستور کار خود قرار می دهند. یکی از راه های دستیابی به این مهم پوشش دادن فرآیندهاست. متدهای خانواده آجیل نیز در فرآیندهایشان منفعت رقابتی مشتری را پوشش می دهند. چگونگی این عمل در خود اصل نهفته است: "استقبال از نیازمندی ها، حتی در انتهای توسعه". توسعه به هر کجا قدم بگذارد تغییر مانند سایه همراه اوست. برخی تغییرات بر اساس نیازهای کاربران است، اما برخی تغییرات به رقابت تجاری ارتباط دارد. تغییراتی که به رقابت های تجاری مرتبط هستند را اصطلاحاً تغییرات سریع می نامند. عکس العمل کُند در مواجهه با این تغییرات ممکن است باعث جا ماندن از رقابت و حتی شکست شود(شکل 13-1).



شکل 13-1: چارچوب رقابت منفعتی مشتری

به رأی بسیاری از توسعه دهندگان بدترین نوع تغییر، تغییرات انتهای توسعه می باشد. اِعمال تغییر در ابتدای توسعه آسان است اما به عکس درانتهای توسعه به دلیل سطح وسیع آن امری بسیار مشکل می باشد. یکی از کلیدهای حل این مشکل پیشگیری است؛ به این معنی که اگر از فرآیندهای مناسبی برای توسعه استفاده

شود، توسعه در راستا ارضاء نیازها با روند بهتری همراه خواهد بود و در ادامه نیاز به اِعمال تغییرات به حداقل می رسد. فرآیندهای آجیل به همه تغییرات به ویژه آن هایی که در توسعه ظاهر می شوند خوش آمد می گویند.

3. تحویل نرم افزار کارکننده غالباً بین دو هفته تا دو ماه می باشد، زمان های کوتاه تر ارجحیت دارند.

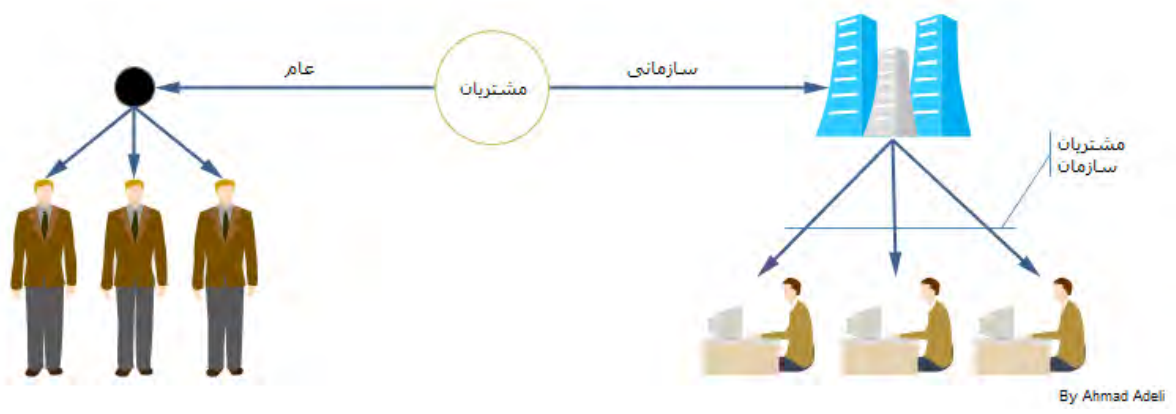
تجربه ثابت کرده تکرارهایی با زمان زیر دو هفته تکرار های ناقصی هستند و معمولاً نمی توان از این تکرارها نرم افزار با ارزشی استخراج کرد. از طرفی هر تکرار مستلزم جلساتی قبل و بعد از خود می باشد که طبیعاتاً باعث اتلاف وقت خواهند شد. توسعه های بیش از دو ماه نیز چندان نمی توانند خوب عمل کنند زیرا همان طور که قبلاً ذکر شد هر تکرار را شبیه به یک پروژه کوچک با متدی ترتیبی بدل خواهد کرد که قاعدتاً مشکلات این متد منسوخ شده را نیز به همراه خواهد داشت. از طرفی تکرارهای بلند مدت باعث کاهش ارتباط با مشتری می شود که خود باعث نقض ارزش اول ار بیانیه آجیل می گردد. همچنین با تحویل دیر هنگام نرم افزار و اعمال تغییرات کمتر (به سبب تأخیر در شناسایی آن ها) به ترتیب اصل اول و دوم بیانیه نقض خواهد شد.

به چند دلیل تکراها با زمان کوتاه مناسب اند: مهمترین دلیل آن شناسایی تغییرات است. در نظام اقتصادی عصر حاضر، پاسخ سریع به تغییرات سودآوری را تضمین می کند. با طولانی شدن زمان تکرارها ارتباط تیم با مشتری کم رنگ خواهد شد که خود باعث می شود تغییرات لازم دیرتر کشف شوند و متعاقب آن دیرتر اِعمال گردند. دلیل بعدی می تواند کوتاه شدن زمان سازماندهی مجدد تیم باشد. همانطور که می دانید در توسعه عوامل انسانی جزو عناصر مهم تلقی می شود. خطا در عمل انسانی امری است اجتناب ناپذیر به خصوص اگر عمل، خلاقانه و غیر روتین باشد. پس از هر تکرار این فرصت وجود دارد که تیم اشکالات خود را بررسی کند و در پس رفع آن در تکرارهای بعدی نماید(این اشکالات حتی می توانند مهارتی و تکنیکی باشند). برای برخی نرم افزارها تحویل سریع نسخه های جدید نرم افزار (نرم افزار به علاوه بخش های ایجاد شده در تکرار جاری) حیاتی است. این مورد هم دلیلی بر تکرارهایی با زمان کوتاه تر تلقی می شود.

4. افراد تجاری(مشتری ها) و توسعه دهندگان باید روزانه در طول پروژه با یکدیگر کار کنند.

سابق بر این در این فصل درباره این مهم سخن گفته شد. اما با نگاهی دقیق تر به جمله فوق کلمه ابهام بر انگیزی دیده خواهد شد: *روزانه*. بر طبق این اصل افراد تیم توسعه و افراد تجاری(مشتریان) همکار هستند و هر روز با هم در ارتباط اند. قبل از بررسی مفهوم عبارت فوق اشاره ای به انواع مشتری خواهد شد.

از منظر تیم هایی که در صدد توسعه نرم افزار هستند مشتری ها به دو دسته تقسیم می شوند: مشتری های سازمانی و مشتری های عام. مشتریان سازمانی مشتری هایی هستند که در نوع خود سازمان یا شرکتی می باشند که با افراد دیگر(مشتریانی دیگر) سرو کار دارند. این سازمان ها اغلب نرم افزارها را برای مصارف داخلی خود استفاده می کنند مثل سیستم انتخاب واحد در یک دانشگاه. دسته دوم مشتریانی هستندکه به طور مستقیم، یا نرم افزار را می خرند و یا از آن سرویس می گیرند. این افراد نرم افزار را برای مصارف شخصی خود استفاده می کنند مثل بازی ها کامپیوتری(شکل 14-1).



شکل 14-1: انوع مشتری

اصل فوق الذکر (اصل 4) بیشتر در خصوص مشتریان سازمانی صدق می کند. در توسعه نرم افزار تغییرات به صورت دوره ای شناخته نمی شوند(که بتوان در تکرارهای آنها را شناخت) بلکه تغییرات لحظه ای شناسایی می شوند. برای جلوگیری از دوباره کاری باید همواره به تغییرات احتمالی گوش فرا داد. سوالی که ممکن است هم اکنون ذهن شما را به خود مشغول کرده باشد این است که چگونه چنین همکاری نزدیکی قابل پیاده سازی است؟ پاسخ به این سوال ساده است، کافی است در خود سازمان کار شود. کار کردن تیم توسعه در سازمان آن ها را در نزدیک به مشتری شان(افراد تجاری) قرار می دهد. این عمل باعث آشنا شدن با مشکلات سازمان، کشف بهتر نیازها، تشخیص مناسب تر تغییرات و در نهایت افزایش کیفیت نرم افزار خواهد شد. ممکن است به نظر رسد این پاسخ کمی غیر واقع بینانه است. اگر این سازمان ها در جغرافیایی قرار داشته باشند که فرهنگ کسب و کار در آن ضعف داشته باشد می تواند کمی بعید به نظر برسد. اما باید توجه داشت هیچ فرآیند یا متدی نمی تواند معجزه کند. نمی توان صرفاً داخل یک فرآیند را بهبود بخشید. تغییرات داخلی بر محیط بیرون نیز اثر گذار خواهد بود. ارتباط نزدیک مشتری موجب ارتقاء کیفیت نرم افزار خواهد شد؛ این مفهوم نظریه ای ثابت شده است. پس اگر کیفیت نرم افزار افزایش یابد(تغییر کند)، پیرو آن فاصله ارتباطی مشتری و تیم توسعه نیز تغییر می کند(کم می شود). پس اگر سازمانی خواهان نرم افزاری با کیفیت باشد، طبیعتاً تغییراتی این چنینی را نیز باید پذیرد.(لازم به ذکر است این پاسخ صرفاً یک راه خل برای سوال فوق می باشد. در فصول بعد روش های بیشتری شرح داده می شود).

اصل جاری در رابطه با برخی مشتریان عام نیز صادق است. شکی نیست که ارتباط روزانه با مشتریان عام کاری است بسی دشوار(و شاید غیر ممکن). اما نرم افزارهایی تجاری ای وجود دارند که از تکنیک بازخورد آنلاین بهره می جویند. تا ارتباط خود را با مشتریان دائماً حفظ نمایند.

5. پروژه را حول افراد با انگیزیه بنا کنید. محیط کار به آن ها بدهید و از نیازهای آن ها پشتیبانی نمایید. و به آن ها اعتماد نمایید تا کارها را انجام دهند.

نکاتی که در این اصل نهفته است در حیطه روانشناسی کار گنجانده می شوند که در این جا به صورت مختصر به آن ها اشاره می شود. همانطور که می دانید انگیزه افراد برای انجام کارها با راندمان کاری آن ها رابطه ای مستقیم دارد(راندمان معیاری است برای برآورد کیفیت کار که از فرمول تعداد کار صحیح تقسیم بر تعداد کل کارها بدست می آید). افراد بی انگیزه در یک پروژه یا بیکار هستند یا بسیار کُند عمل می کنند. در صورت انجام کاری نیز به احتمال بسیار زیاد کاری بی کیفیت(و حتی نادرست) ارائه خواهند داد.

منابع انسانی جزو مهمترین سرمایه های یک سازمان محسوب می شوند. همانطور که یک سرمایه گذاری مالی اشتباه ممکن است یک سازمان را به ورطه نابودی بکشاند؛ سرمایه های انسانی نادرست نیز موجب شکست یک پروژه خواهند شد.

افراد از نظر ضعف انگیزشی به سه نوع تقسیم می شوند: نوع اول آنهایی هستند در رابطه با شغلشان کاملاً بی انگیزه هستند(که ممکن است دلیل آن بی علاقگی و یا گزینش شغلی نادرست باشد). بدیهی است که افراد نوع اول هرگز نباید توسط سازمان و یا تیم توسعه استخدام شوند. نوع دوم افرادی هستند برای پروژه ای که در آن کار می کنند انگیزه ای ندارند. مشکل اینگونه افراد با انتقالشان به سایر پروژه ها و یا ارائه اطلاعات توجیه پذیر به آن ها در رابطه با پروژه حل نمود. نوع سوم نیز افرادی هستند که نسبت به وظیفه که در پروژه بر عهده دارند بی انگیزه هستند که مشکل انگیزشی این افراد اغلب با جابجایی بین وظایف بر طرف خواهد شد.

ایجاد انگیزش در افراد موجب ارتقاع سطح کیفی فرآیند توسعه و متعاقب آن موجب بالا رفتن کیفیت یک محصول خواهد شد. به افراد محیط کار بدهید؛ این جمله به ظاهر ساده از اصل فوق، مفهومی عمیق را در بر دارد. این مفهوم عمیق در مفهوم محیط کار نهفته است. داشتن محیط کار امری است بدیهی اما باید دانست که محیطی که افراد به اجبار در آن کار می کنند را نمی توان محیط کار نام نهاد. محیط کار محیطی است که داری جوی باشد که افراد تیم بدون دغدغه های جنبی (مثل شلوغی، رفت و آمد، ارباب رجوع و ...)بر بروی توسعه تمرکز کند. مهیا سازی چنین محیطی کار آسانی نیست؛ محیط کار تیم توسعه باید مستقل باشد. مستقل بودن به معنای مجزا بودن نیست. محل کار تیم می تواند از نظر جغرافیایی درون خود سازمان قرار گیرد با این تفاوت که واحدی مستقل است. این واحد مستقل بهتر است از لحاظ مدیرتی تا حد زیادی خود مختار بوده تا کمتر درگیر امور دست و پا گیر سازمانی شود. محیط کار باید قابلیت سفارشی سازی داشته باشد. تیم های مختلف محیط های کاری مختلفی را طلب می کنند. به طور کلی یک محیط کار خوب محیطی است که افراد درون آن احساس راحتی می کنند. از طرفی بهتر است ارتباط با بیرون برای افراد آسان باشد به خصوص اگر تیمی درون یک سازمان اسکان داده شود.

برخی افراد مایلند که صاحب دفاتر شخصی و جداگانه باشند. از آن جایی که در فرآیند توسعه افراد تیم ارتباط زیاد و البته بسیار نزدیکی با یکدیگر دارند بهتر است که افراد چندان دور از هم کار نکنند و در یک محل اسکان داده شوند.

تا آنجایی که ممکن است باید سعی شود تا واحد توسعه از واحد های دیگر مجزا شود تا از به وجود آمدن تداخلات احتمالی پیشگیری به عمل آید. این واحد دیگر سازمان از دو نظر قابل بررسی هستند: این واحدها می توانند به معنی واحدهای دیگر سازمان باشند با این فرض که تیم درون سازمان اسکان داده شده است و یا به عنوان دیگر واحدهای توسعه در یک محصول تجاری (که اغلب بزرگ می باشد و دارای واحدها و تیم های زیادی است) تلقی شوند.

6. کارآمدترین و موثرترین روش انتقال اطلاعات به تیم توسعه، محاوره رودرو است.

تبادل اطلاعات بین تیم توسعه و مشتری دو طرفه است. مشتری انتظاراتی را که از نرم افزار دلخواهش دارد در قالب اطلاعات به تیم منتقل می کند و در آن طرف تیم توسعه نیز اطلاعاتی از روند اجرایی پروژه در قالب گزارش به مشتری انتقال می دهد. در فرآیندهای سنتی این تبادل اطلاعات غالباً به صورت یک بروکراسی خشک و غیر قابل انعطاف صورت می پذیرد که باعث می شود سرعت تبادل اطلاعات به شدت تنزل کند. از آن جایی که این انتقال اطلاعات جزوی از فرآیند توسعه می باشد(جمع آوری نیازها)، طبیعتاً باعث کاهش سرعت کل توسعه خواهد شد. بدیهی است که این کاهش سرعت موجب افزایش زمان توسعه و متعاقب آن افزایش هزینه نرم افزار نهایی می گردد.

اغلب اطلاعاتی که سازمان ها در اختیار تیم های توسعه قرار می دهد شامل: فرم ها، چارت ها، گزارشات سازمانی است. برخی از این اسناد از نظر امنیت سازمانی حساس می باشند که در این صورت به سختی در اختیار تیم قرار می گیرد و بیشتر اوقات ناقص و ناکافی است. این نقص اطلاعات هم موجب اشتباه تیم در استخراج نیازمندی های نرم افزار می شود و هم کار تحلیل را مشکل تر می سازد. یکی دیگر از مشکلاتی که در این نوع تبادل اطلاعات سنتی وجود دارد، اطلاعات اشتباه است. به علت عدم ارتباط مستقیم در این ارتباط ممکن است اطلاعاتی به تیم داده شود که اشتباه باشد(مثلاً به تیم اطلاعاتی راجع به فرم های منسوخ شده تحویل گردد). طرف دیگر قضیه اطلاعاتی است که تیم راجع به توسعه در اختیار سازمان قرار می دهد. هر چقدر هم که گزارشات کامل باشند باز هم نمی توان درک کامل و مناسبی را از نرم افزار در حال توسعه منتقل کرد. به مشکلات فوق، مشکل تفسیر نادرست درخواست های تیم از سازمان را نیز باید اضافه کرد.

در این اصل صراحتاً گفته شده که مکالمه باید چهره به چهره صورت بگیرد. از طریق برگزاری جلساتی بین تیم و مشتری(نمایندگان سازمان) می توان به سرعت هر نوع اطلاعات را به صورت دو طرفه دنبال کرد. در واقع مکالمه حضوری طرفین سریع ترین راه انتقال اطلاعات می باشد. در یک جلسه مشتری انتظارات خود را از نرم افزار بیان می کند و همچنین اطلاعات لازم جهت توسعه را در اختیار تیم قرار می دهد. در آن طرف تیم هم در صورت کمبود اطلاعات در آن واحد می تواند درخواست اطلاعات نماید. گزارشات تیم نیز در جلسه به صورت ارائه شفاهی منتقل می شود. مزیت مهم این شیوه این است که در صورت به وجود آمدن هر ابهامی، آن ابهام در لحظه بر طرف خواهد شد.

تا اینجا در رابطه با سرعت انتقال اطلاعات و صحت اطلاعات منتقل شده مطالبی کفته شد. سوالی که این جا مطرح است این است که آیا می توان در جلسات حضوری، اطلاعات کاملی را منتقل کرد؟ هر شیوه ای چالش های خاص خود را دارد. در روش پیشین هر چقدر هم اطلاعات زیاد و طولانی باشد مشکل خاصی برای مخاطب ایجاد نمی کند ولی روش محاوره ای رو در رو روشی است زمان محور؛ به این دلیل که حجم اطلاعات زیاد است و جلسات را نمی توان برای مدت زیادی ادامه داد. جلسات بلند مدت باعث کاهش کارایی اعضای می شود. یک جلسه طولانی را می توان به چند جلسه کوتاه تر تقسیم کرد. از طرفی ازدیاد جلسات کوتاه مدت نیز از سوی طرفین پذیرفته نشود. بهترین راه حل ایجاد توازنی بین روش سنتی(تبادل اسناد) و روش جدید(محاوره رو در رو) می باشد. بهتر است مهمترین موضوعات در جلسات مطرح شوند و موضوعاتی که دقت و بررسی بیشتری نیاز دارند را به صورت سند و فرم منتقل کرد. به طور کلی جلسات محاوره ای باید از مدیریت زمان خوبی برخوردار باشند و بهتر است بررسی ها و تفحص ها پیش از جلسه صورت بپذیرد و جلسه مکانی باشد برای توافقات و ارائه نتایج.(تیم توسعه و مشتری می توانند برای صرفه جویی در وقت از تکنولوژی هایی چون ویدیو کنفرانس نیز بهره ببرند).

7. نرم افزار کار کننده اصلی ترین معیار پیشرفت است.

همانطور که قبلاً اشاره شد نرم افزار کار کننده نرم افزاری است که در خود دارای پتانسیل کاربرد پذیری باشد. این موضوع در حقیقت روی یکی از بدیهیات توسعه نرم افزار تاکید می کند. البته لازم به ذکر است برای برخی از توسعه دهندگان به ویژه استفاده کنندگان از روش ها سنتی چندان بدیهی نمی باشد.

یکی از معیارهای نه چندان خوبی که در گذشته برای پیشرفت پروژه و حتی قیمت گذاری استفاده می شد، تعداد خطوط کد منبع بود. به این صورت که اگر برنامه ای دارای 10000 خط کد بود و پس از یک دوره تعداد خطوطش به 15000 خط کد می رسید این گونه تفسیر می شد که پروژه X% پیشرفت کرده است. البته این روش زمانی برنامه نویسی ساخت یافته(لوله ای) مرسوم بود استفاده می شد(که البته در همان موقع نیز روش بسیار ضعیفی بود). مشکل این روش این است که برای یک کاربرد بسته به مهارت توسعه دهندگان پیاده سازی های متفاوتی را می توان ارائه داد. بنابراین ممکن است با 5000 خط کد از تیمی یک کاربرد و از تیمی دیگر 5 کاربرد ارائه شود. از طرفی در دنیای واقعی این کاربر نهایی است که از کاربرد استفاده می کند و پیشرفت آن را تعیین می کند بنابراین اگر کاربر در کاربرد تغییری حس نکند، نرم افزار عملاً پیشرفتی نداشته است.

علاوه بر روش های فوق الذکر روش های فاجعه آمیز(!) دیگری نیز پیش از این آزمایش شده است مانند معیار زمان استفاده شده. فرض کنید پروژه ای برای 10 ماه برنامه ریزی شده باشد با منطق روش پیشرفت پروژه با استفاده از زمان با گذشت دو ماه از شروع، پروژه دارای پیشرفت 20% است! که برآوردی کاملاً اشتباه است. متاسفانه این روش از صنایع دیگر به عاریه گرفته شده است. فرآیند توسعه نرم افزار یک فرآیند قطعی نیست که بتوان با قاطعیت درباره زمان و کاربردهای آن گاهی در بهترین شرایط توسعه نیز دوباره کاری اجتناب ناپذیر است. دوباره کاری همواره با اتلاف زمان همراه است. امروزه در توسعه دهندگان در راستای حفظ زمان توسعه از روش هایی چون "توسعه نرم افزار مبتنی بر قطعه استفاده می شود. به عقیده بسیاری این کاربرد های نرم افزار است که پیشرفت توسعه را تعیین می کند. به عبارت دیگر در صورتی می توان پیشرفت یک نرم افزار را رو به رشد تعبیر کردکه کاربردهای مفید (قابل استفاده) آن در هر تکرار رو به افزایش باشد.

8. فرآیندهای آجیل توسعه پایدار را ترویج می دهند. حامیان، توسعه دهندگان و کاربران باید قادر به حمایت از سرعت ثابتی به صورت نامحدود باشند.

پایداری در یک نرم افزار به این معنی است که یک ورودی در شرایط مختلف خروجی های یکسانی را ارائه کند. در حقیقت ممانعت در برابر تغییراتِ پاسخ یک نرم افزار را در حالات مختلف، پایداری نرم افزار می گویند و نرم افزاری که این ویژگی را دارا باشد نرم افزار پایدار می نامند. اما موضوع مورد بحث در این فصل پایداری نرم افزار نیست بلکه توسعه پایدار می باشد. توسعه پایدار به این موضوع اشاره دارد که تلاش برای توسعه در یک بازه زمانی در شرایط مختلف نتیجه یکسانی را ارائه دهد. فرض کنید بازه زمانی توسعه 2 هفته ای باشد و این تکرارها در توسعه ای با شرایط مختلف انجام شوند. اگر این تکرارها از نظر کمیت و کیفیت دارای نتایج تقریباً یکسانی باشند توسعه مذکر توسعه ای پایدار است. بخش دوم اصل از همه اعضای درگیر در پروژه(حامیان، توسعه دهندگان و کاربران) سرعتی ثابت را برای توسعه طلب می کند. این سرعت ثابت نتیجه یک توسعه پابدار می تواند باشد. حال سوال اینجا است که این سرعت ثابت تا چه زمانی باید ادامه داشته باشد. اگر به جمله دوم از اصل جاری توجه شود واژه "نامحدود" به چشم می خورد که پاسخ سوال فوق الذکر است. این کلمه بسیار هوشمندانه در جمله تعبیه شده است. پروژه ها دارای زمان های محدودی می باشند؛ محدوده زمانی زیر مجموعه ای از زمان نامحدود است. اگر فرآیندی بتواند تضمین کند که توسعه را به صورت نامحدود با سرعتی ثابت پیش ببرد، این فرآیند را می توان برای انواع پروژه با گستره زمانی متفاوت به کار برد.

برخی از صاحب نظران مزیت متدهای سنتی را بر متدهای فرآیند مدرن پیش بینی پذیر بودن آن ها می دانند. پیش بینی پذیری صفت مهمی در یک پروژه است. پیش بینی پذیری، باعث می شود یک پروژه مدیریت پذیر شود به این معنا که اگر بتوان پیش بینی های مناسبی از زمان و هزینه یک پروژه بدست آورد مدیریت آن پروژه به مراتب آسان تر خواهد شد. سرعت توسعه ثابت با قابلیت پیش بینی پذیری یک پروژه رابطه مستقیم دارد. از این رو مهمترین مزیت ثابت بودن سرعت توسعه یک پروژه را می توان مدیریت پذیری بهتر دانست.

9. توجه مداوم به برتری فنی و طراحی خوب موجب ارتقاع چالاکی می شود.

هدف اصلی بیانیه آجیل(*چالاک*) تسریع حرکت لاک پشتی توسعه سنتی نرم افزار و ایجاد چالاکی در آن است. به جرأت می توان گفت *زمان*، تعیین کننده ترین عنصر در توسعه نرم افزار است. در بسیاری از پروژه ها زمان معیار قیمت گذاری ارزش گذاری و موفقیت پروژه می باشد. حتی اگر در اثر اشتباه مجبور به دوباره کاری کل

پروژه شویم (نرم افزار را از نو بسازیم) باز هم این زمان است که تعیین کننده می باشد. در اوایل دهه 90میلادی بسیاری از توسعه دهندگان کلید افزایش کیفیت محصولات نرم افزاری را در تسریع فرآیند توسعه یافتند. بنابراین سعی در کشف روش هایی برای چالاک تر کردن توسعه نرم افزار شد. به بیان ساده تر چالاک سازی فرآیند توسعه باعث حفظ و مدیریت بهتر زمان می شود . ذخیره زمان بیشتر در توسعه، محصول با کیفیت تری را نتیجه می دهد.

برتری فنی و طراحی خوب صراحتاً در این اصل به عنوان دو عامل افزایش چالاکی در توسعه معرفی شدند. بهتر است توسعه دهندگان برای حفظ چالاکی در توسعه همواره در پی تکنیک های فنی برتر باشند. برخی تکنیک ها نسبت به دیگری مزیت های اضافه تری دارند. همه تکنیک های مشابه در نهایت به یک نتیجه ختم می شوند. اما در تکنیک های برتر عموماً طراحی، پیاده سازی و البته خطا یابی بهینه تر صورت می پذیرد به این معنی که کمتر دارای پیچیدگی هستند که این خود موجب ذخیره زمان و متعاقباً افزایش چالاکی در توسعه خواهد شد. در یک توسعه نوعی اگر طراحی اولیه با طراحی نهایی مقایسه شود ممکن است تفاوت فاحشی بین آن ها مشاهده شود؛ این حالت در توسعه نرم افزار کاملاً طبیعی است چرا که توسعه در طول عمر خودش تغییرات بسیاری را متحمل می شود، علاوه بر این نمی توان تضمین کرد که یک طراحی بدون تغییر نیازها نیز یک طراحی خوب است.

جایگزینی یک طرح با طراحی برتر امر عادی در توسعه نرم افزار است. اما این جایگزینی اغلب یک مشکل عمده به همراه دارد و آن هم دوباره کاری است. با اینکه دوباره کاری در توسعه های نرم افزاری اتفاقی معمول است اما توسعه دهندگان همیشه در پی کاهش آن هستند. اصطلاحی به نام طراحی خوب (Well Design که البته در بیانیه آجیل از آن به عنوان Good Design یاد شده است) بین توسعه دهندگان وجود دارد که اشاره به طراحی با حداقل دوباره کاری ممکن را دارد. به این معنی که طراحی از نظر فنی در سطح بالایی انجام می شود و به سبب این کیفیت بالا نیازی به دوباره کاری نخواهد بود. دوباره کاری همواره یک پیامد منفی را به همراه دارد و آن چیزی نیست جز اتلاف وقت. برای اجتناب از اتلاف وقت همواره باید با استفاده از یک طراحی خوب از دوباره کاری پیشگیری به عمل آورد. بدیهی است که این پیشگیری به طور مستقیم در بالا بردن چالاکی توسعه نقش دارد. البته ناگفته نماند یک طراحی خوب عملی زمان بر است.

برخی بر این عقیده هستند که استفاده ار شیوه آزمون و خطا و دوباره کاری طراحی به مراتب زمان کمتری از یک طراحی خوب خواهد برد. اما به طور قطع نمی توان در این مورد نظر داد. درست است که دوباره در برخی اوقات زمان کمتری از یک طراحی خوب اخذ می کند اما پیامدهای منفی دیگری چون تلف کردن انرژی توسعه دهندگان، افزایش مشقت و از همه مهمتر تاثیر منفی در روحیه تیم را با خود به همراه دارد. از طرفی طراحی خوب به دانش فنی بالایی نیز دارد و تلاش مضاعفی را طلب می کند. هنگامی که مزایا و معایب دو چیز برابر می شود بهتر است نسبت به شرایط، توازنی بین آن ها ایجاد کرد. در موضوع مورد بحث ما می توان یک طراحی خوب را جایگزین یک طراحی معمولی که این خود زمان کمتری نسبت به یک طراحی خوب(Well Design) می گیرد و از طرفی سایر کاستی های طراحی نسبتاً خوب را با دوباره کاری هایی به مراتب کمتر از حالتی که طراحی عادی است جبران کرد.

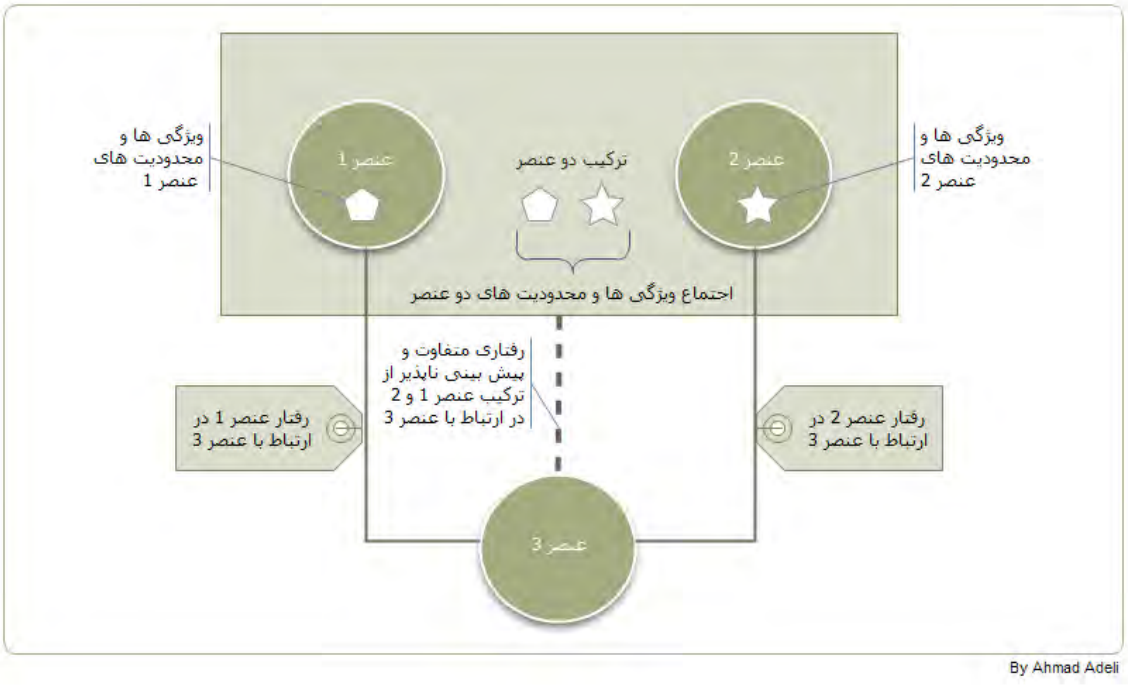
10. سادگی (هنر حداکثر سازی کارهایی که نباید انجام گیرد) ضروری می باشد.

این اصل به طور مستقیم به ساده سازی فرآیند توسعه اشاره دارد و از سادگی به عنوان یک ضرورت یاد می کند. در دل این جمله سادگی تعریف شده است. سادگی عبارت است از "هنر حداکثر سازی کارهایی که نباید انجام داد". در ابتدای این تعریف از سادگی به عنوان یک هنر یاد می شود. هنر بیشترین کار انجام نشده! این تعریف گنگ می تواند کمی ابهام ایجاد کند پس بهتر است به این صورت بیان شود: هنر کمینه کردن کارهای اضافی. به زبان ساده تر این اصل به ما می گوید که اگر در فرآیند توسعه کارهای اضافی و نالازم را به حداقل برسانیم سرعت و چالاکی توسعه افزایش خواهد یافت. به بیانی دیگر سادگی پاسخ این سوال است: چرا توسعه نرم افزار با استفاده از فرآیندهای قوی و پیچیده ای همچون RUP همچنان کند است؟ این سوال ذهن بسیاری از توسعه دهنگان اواخر دهه 80 میلادی و اوایل دهه 90 میلادی را به خود مشغول کرده بود.

بسیاری از این توسعه دهنگان بر این بودند که ریشه این معضل در پیچدگی فرآیند توسعه نهفته است. توسعه نرم افزار پیچیده و بسیار زمان بر و پر هزینه است. برای شناخت مفهوم پیچیدگی بهتر است آن را تعریف کنیم؛ پیچیدگی عبارت است از ارتباط مبهم دو یا چند عنصر به طوری که الگوی رفتاری آن ها به عنوان یک کل قابل پیش بینی نباشد. فرض کنید چند عنصر با ویژگی های کاملاً شناخته شده داشته باشیم رفتارهای این عناصر واحد تا حد زیادی قابل پیش بینی است. حال دو عنصر از این عناصر را با یکدیگر ارتباط می دهیم. از ارتباط این

دو، یک عنصر بزرگ تر حاصل خواهد شد که ویژگی ها و محدودیت های این عنصر، اجتماع ویژگی های و محدودیت های عناصر تشکیل دهنده آن است. اما آیا می توان گفت رفتار عنصر جدید نیز اجتماع رفتارهای عناصر تشکیل دهنده آن است؟ مطلقاً خیر. رفتارهای عناصر کوچک ممکن است با هم در تضاد باشد و یا ترکیب آن ها موجب رفتاری ناشناخته شود که این خود باعث عدم پیش بینی پذیری رفتار عنصر بزرگ تر خواهد شد(شکل 15-1).

با اضافه کردن عناصری دیگر به این ترکیب عنصر ترکیبی به عنصری بزرگ تر با ویژگی های بیشتر بدل خواهد شد و رفتار آن رفتاری به مراتب ناشناخته تر، پیش بینی ناپذیرتر و البته پیچیده تر می شود. در نتیجه به طور کلی می توان گفت موضوعاتی که شناخت کمتری از آن ها داریم موضوعاتی با قابلیت پیش بینی پذیری پایین هستند و به عنوان یک اصل باید به یاد داشته باشیم " هرچه قابلیت پیش بینی پذیری پایین باشد، پیچیدگی افزایش می یابد".



شکل 15-1: مدلی از مفهوم پیچیدگی

توسعه نرم افزار از اجزای زیادی تشکیل شده که ارتباط تنگاتنگی با یکدیگر دارند اما صرفاً این موضوع باعث پیچیدگی بی حد و حصر آن نشده است؛ بلکه مهمترین دلیل این پیچیدگی ازدیاد قوانین و هنجارها در فرآیند توسعه است. حال باید بررسی شود که این قوانین چگونه وارد فرآیند توسعه نرم افزار شده اند. در اویل شکل گیری صنعت نرم افزار (مانند هر صنعت دیگری) توسعه دهندگان اشتباهات زیادی را در توسعه نرم افزار مرتکب می شدند. این اشتباهات به مرور با پیشرفت صنعت نرم افزار روندی تنزلی را پیش گرفتند. این کاهش اشتباهات با افزایش کشف روش ها و افزودن آن ها به صورت قوانین در فرآیند توسعه همراه بود. از این رو فرآیند توسعه نرم افزار از یک حالت ساده به یک فرآیند قوی و غنی که سرشار از قوانی و باید و نباید ها بود تبدیل شد. با کم شدن خطا و کشف روش های جدید و بهبود قوانین قبلی کیفیت فرآیند توسعه بالا رفت و پیرو آن کیفیت نرم افزارهای نهایی نیز بهبود یافت. اما این افزایش کیفیت چندان بدون مشکل انجان نشد. غنی شدن فرآیند توسعه به قیمت افزایش پیچیدگی آن تمام شد. وجود قوانین و هنجارها در توسعه از طرفی باعث کاهش احتمال توسعه نادرست خواند شد اما از آن طرف پیچیدگی فرآیند توسعه را به همراه خواهد داشت. واضح است که انجام کار پیچیده مستلزم زمانی به مراتب بیشتر دارد.

به عقیده بسیاری پیچیدگی بزرگ ترین ضربه خود را به برنامه ریزی در توسه وارد می کند؛ چرا که با وجود پیچیدگی، پیش بینی امور سخت و گاهی غیر ممکن می شود. آموزش فرآیندی پیچیده کاری است بس دشوار و وقت گیر. فرآیند توسعه فرآیندی ذاتاً پیچیده است. اما چیزی که آن را پیچیده تر می کند ازدیاد قوانین و شیوه ها است.

راه حل مشکل پیچیدگی فرآیند توسعه در سادگی نهفته است. طبق توصیه اصل 10(اصل جاری) باید سعی شود فرآیند توسعه تا حد ممکن ساده نگه داشته شود. به این معنی که کارهای اضافی باید به حداقل رسد و همیشه به جای انجام شیوه های تجربه شده اما پیچیده باید به دنبال کشف شیوه های جدید بود که کارها را ساده تر کند. هر فرآیندی از قوانین و اصولی پیروی می کند که بر اساس آن طراحی شده است. اغلب فرآیندها به شرط پیروی از اصولی که برایشان تعریف شده، کیفیت نرم افزار را تضمین می کنند. گفتنی است که اصول و قوانین هر فرآیند ضروری، اما از طرفی خشک و غیرقابل انعطاف هستند. بدیهی است که توسعه ای غیر قابل انعطاف توسعه ای چالاک نخواهد بود. در راستای چالاک سازی توسه باید از برخی قوانین و اصول دوری جست و توسعه ای باز را پیش گرفت. اصول دقیقی برای ساده سازی وجود ندارد اما راجع به این موضوع چند اصل در اینجا ارائه می شود:

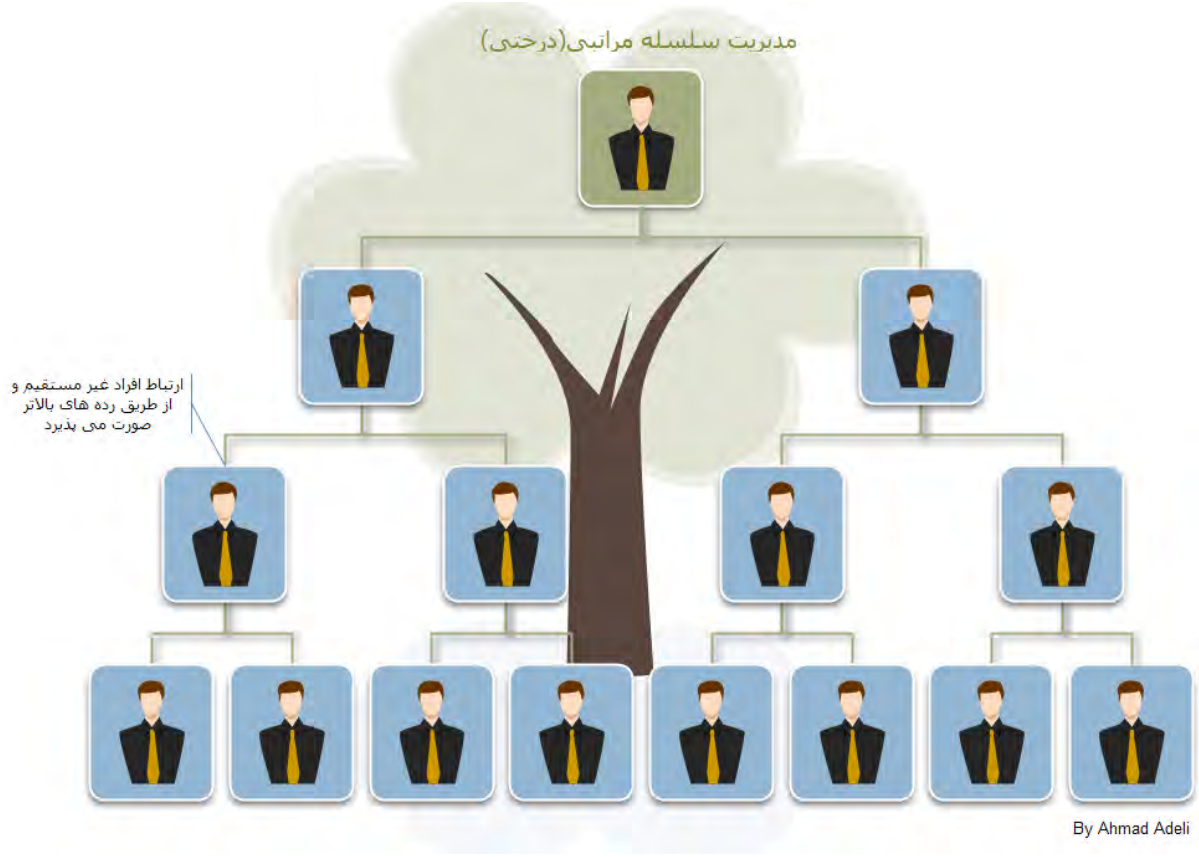
- از قوانینی که دارای توجیهی نمی باشند باید پرهیز شود. سعی شود اصول ابتدا درک شوند تا بینش لازم نسبت به آنها کامل شود سپس با دیدی باز و تمایلی کامل به آن مبادرت ورزید.
- توسعه نرم افزار یک عامل متغیر دارد و آن هم زمان است. این عامل طبیعتاً روی قوانین و اصول یک فرآیند تاثیر می گذارد. ممکن است یک هنجار به گذشته مرتبط باشد، نباید این دلیل که این هنجار پیش از این اجرا شده است و روی آن تأمل شده به آن اعتماد کرد.
- برخی قوانین فقط در حالات نادری روی خواهند داد. مواقعی وجود دارد که در توسعه روی مشکلی که هیچ گاه پیش نخواهد آمد زمان و انرژی زیادی می گذاریم که این خود باعث عقب افتادن زمان بندی پروژه می شود. برای این حالات نادر بهتر است اجازه دهیم تا اتفاق بیافتند(تنها در صورتی که جزو حالات حیاتی و اضطراری نباشد) و سپس به رفع عیب آن بپردازیم.

مجدداً بهترین روش بین چند روش، قوانین یا حالات ایجاد *توازن* بین آن ها است. برای ساده نگه داشتن فرآیند توسعه بهتر است اصول و چارچوب خاصی تعیین نشود تا تیم ها بتوانند خود به صورت موضوعی این ساده سازی را اعمال کنند. سادگی نه تنها در فرآیند توسعه بلکه در طراحی نرم افزار نیز باید اعمال شود؛ چرا که جهان امروز سرشار از پیچیدگی ها است و دور از انتظار نیست در آینده ای نزدیک بر اساس حس درونی راحت طلبی انسان، پیچیدگی های بیشتری (چون نرم افزاری پیچیده) را پس بزند. به بیانی ساده تر "راحتی در سادگی نهفته است".

11. بهترین معماری ها، نیازمندی ها و طراحی ها از تیم های خود سازمانده پدید می آید.

تیم های خودسازمانده تیم هایی هستند که در آن هر عضو هم رئیس است و هم مرئوس. در این تیم ها تصمیمات اغلب گروهی اخذ می شوند. در این نوع تیم ها معمولاً نوع مدیریتی خاصی وجود ندارد و مسائل بر اساس توافق بین اعضا حل می شود. هنگامی که قرار است تصمیمی گرفته شود هر عضو نظر خود را به همراه دلایل آن بیان می کند. تصمیمات بسته به گروه بر اساس اکثریت آرا، بهترین ایده، کم هزینه ترین راه حل، سریع ترین پیاده سازی و ... صورت می پذیرد. هنگام اخذ یک تصمیم همه اعضای یک تیم موظف هستند که به آن عمل کنند مگر این که دلیلی خلاف اجرای تصمیم را به اثبات رسانند. این تیم ها دارای مدیر نیستند بلکه خود اعضا هستند که تیم را مدیریت می کنند. اما در برخی از تیم ها شخصی به نام رهبر تعیین می شود تا تیم را رهبری کند و از انحراف تیم از هدف ممانعت به عمل آورد. یک رهبر یک مدیر نیست و نمی تواند به اتخاذ تصمیمات شخصی بپردازد یا فرمانی صادر کند. بلکه سعی می کند در مسیری درست قدم بر دارد و تیم را در همراه خود در آم مسیر هدایت کند.

اگر افراد یک تیم در تصمیمی گیری اعمالی که خودشان انجام می دهند شرکت داشته باشند عمکرد بهتری برای آن عمل خواهد داشت. فرمان برداری برای هیچ کس خوشایند نیست و افراد مایلند در همه امور (از تصمیمات تا اجرا) مشارکت داشته باشند. طبیعتاً کسانی که خود عامل هستند به دلیل آشنایی بیشتر با عمل مربوطه بهتر می توانند به نسبت به کسانی که صرفاً از دور اجرای آن را نظاره می کنند، برای آن عمل بهتر تصمیم گیری نمایند. تجربه این گونه مدیریت ثابت کرده این نوع تصمیمات منطقی تر و اجرایی تر خواهد بود(شکل16-1).



شکل 16-1: مدیریت سلسله مراتبی

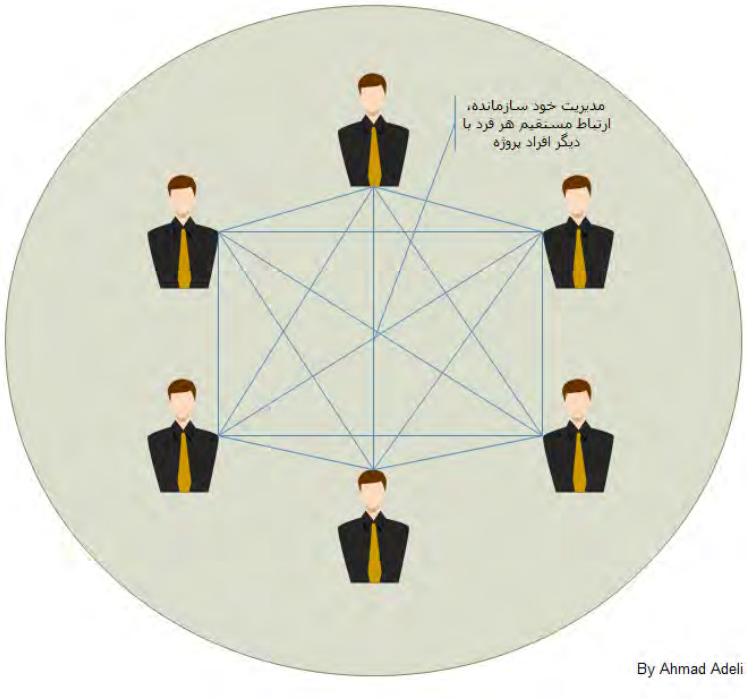
وجود تناقض برای افرادی که با یکدیگر همکاری دارند امری است طبیعی و معمول. در مدیریت سنتی معمولاً این گونه تناقضات توسط مدیر حل و فصل می شود. اما در گروه های خود سازمانده این گونه مشکلات با استفاده از قوانین از پیش تعیین شده و نهایتاً با دخالت رهبر گروه حل خواهد شد.

نقطه مقابل تیم های خود سازمانده، مدیریت بالا به پایین می باشد. همانطور که از نام آن پیدا است این نوع مدیریت (بالا به پایین) به صورت سلسله مراتبی اداره می شود. مدیریت بالا به پایین را مدیریت سنتی نیز می نامند چرا که سال های زیادی است که در صنایع مختلف از آن استفاده می شود. بر اساس این نوع مدیریت سازمان (یا تیم) به سطوح مختلفی تقسیم می شود. برای هر سطح یک مدیر تعیین می گردد تا امور سطح خود و سطوح پایین تر از خود را مدیریت نماید. در این نوع مدیریت تصمیمات اغلب در سطوح بالاتر اخذ شده و برای اجرا به بخش های مربوطه ابلاغ می گردد.

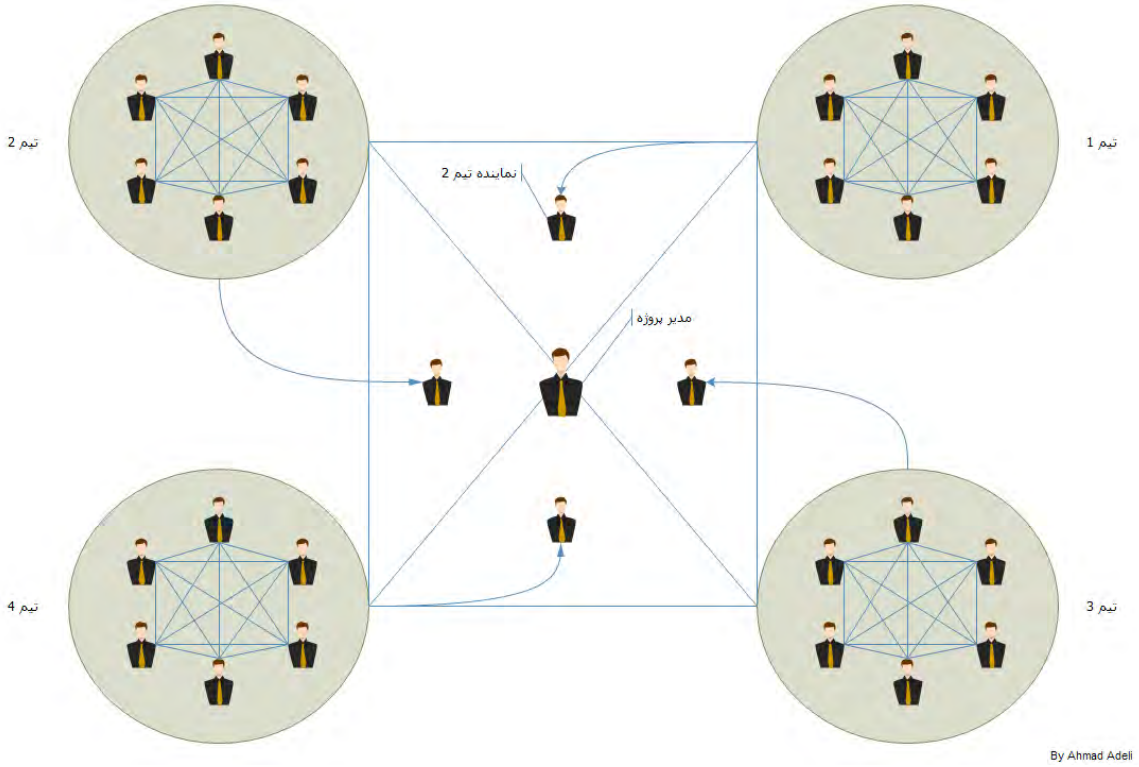
بازخوردهای مدیریتی از بخش های اجرایی با فاصله زمانی بسیار زیاد صورت می پذیرد. به داشتن حالت رئیس و مرئوسی افراد یا فرمان دهنده هستند یا فرمان پذیر به عبارت دیگر اغلب شناختی که مدیران از محیط های اجرایی دارند با شناخت حقیقی فاصله زیادی دارد. به دلیل یک طرفه بودن مدیریت، تصمیمات با خطا همراه است و اعمالی که بر اساس تصمیمات گرفته می شود و اغلب کارایی پایینی دارند. به دلیل فاصله زیاد مدیریت با پایین ترین سطح اجرایی و عبور تصمیمات از سلسله مراتب طولانی، سرعت اجرایی بسیار پایین و ناکارآمد است. این گونه مدیریت ها اغلب برای تیم ها و سازمان های بزرگ انتخاب می شوند. اما برای گروه های کوچک گزینه مناسبی نیستند. در آن طرف، گروه های خودسازمانده اغلب گروهی هایی با تعداد اعضای کم هستند اگر در این گونه تیم ها از افراد زیادی استفاده شود به دلیل تداخلات بیش از حد، هرج و مرج به وجود می آید و تیم از هم گسسته می شود(شکل 17-1). اما این موضوع به این معنی نیست که نوع مدیریت خود سازمانده برای تیم های بزرگ غیر قابل اجرا است.

برای اعمال این گونه مدیریت، افراد بر حسب فیلد کاری شان به تیم های کوچک تقسیم می شوند. هر تیم به صورت یک گروه خود سازمانده اداره می شود. تیم های مختلف هر کدام دارای نماینده ای می باشند و ارتباط این تیم های از طریق این نماینده بر قرار خواهد شد. در این صورت خود نمایندگان گروه ها نیز در سطحی بالاتر

یک گروه را تشکیل می دهند (چیزی شبیه به هیئت مدیره با اختیار و قدرت کمتر برای هر کدام از اعضا) که به صورت خود سازمانده مدیریت می شود. معمولاً در این نوع مدیریت مدیر پروژه مسئول هماهنگ ساختن کل تیم ها و اخذ تصمیمات کلان و استراتژیک می باشد(شکل 18-1).



شکل 17-1: تیم خود سازمانده



شکل 18-1: مدیریت کلان در تیم های خود سازمانده

متاسفانه سال ها است که در توسعه نرم افزار از مدیریت سنتی(بالا به پایین) استفاده می شود. مدیریت بالا به پایین ذاتاً بر اساس مدیریت اسیتا بنا شده است اما توسعه نرم افزار برخلاف آن پویا است. طبیعتاً اعمال

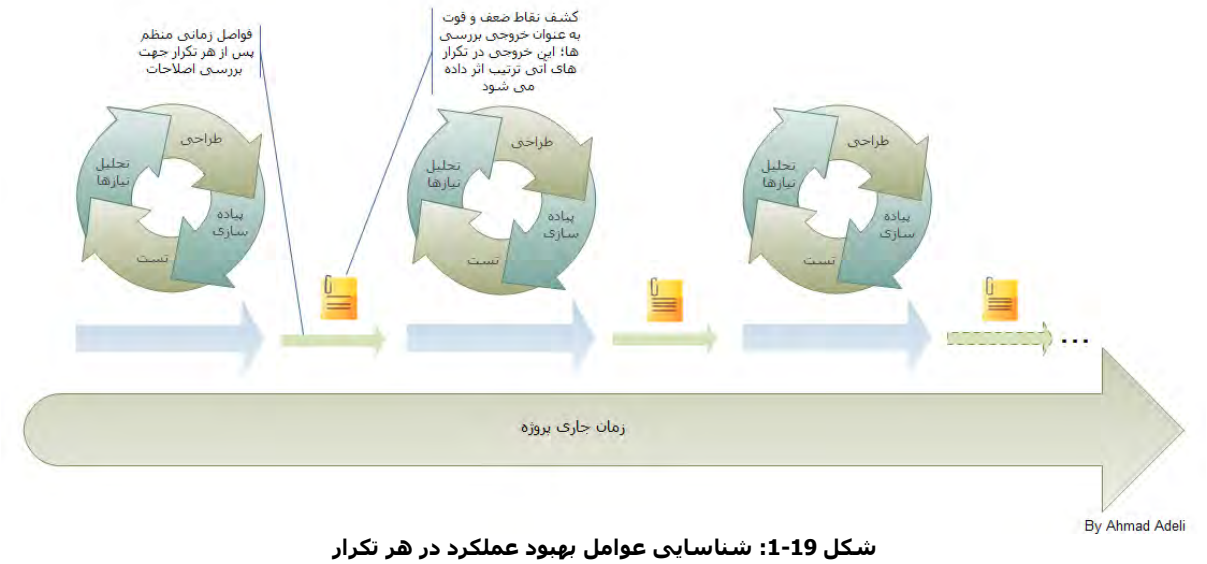
مدیریتی ایستا به یک فرآیند پویا موجب کاهش چالاکی و سرعت در آن فرآیند خواهد شد. از طرفی سراسری بودن موضوعاتی مثل نیازمندی های و طراحی موجب پراکندگی و عدم یکپارچگی مناسب در توسعه خواهد شد.

در طرف مقابل مدیریت خود سازمانده مدیریتی پویا است و با فرآیندی چون توسعه نرم افزار همخوانی مناسبی دارد. یک نرم افزار از دیگاه فنی مجموعه از کامپوننت ها است. درنتیجه می توان اعضای توسعه را به تیم هایی خودسازمانده تقسیم کرد به طوری که هر تیم مسئول توسعه یک کامپوننت باشد. بنابراین هر تیم به طور مستقل عمل کرده و نیازمندی ها و طراحی ها به خوبی محلی سازی می شوند تا نه تنها از تداخل با دیگر کامپوننت ها مصون بماند بلکه بتواند ارتباط خوبی با سایر کامپوننت ها (از دیدگاه منابع انسانی سایرتیم ها) بر قرار کند. از این رو به جرأت می توان ادعا کرد "بهترین معماری، نیازمندی ها و طراحی در تیم های خودسازمانده پدید می آید".

12. در فواصل زمانی منظم، تیم راجع به چگونه موثرتر شدن تفکر می کند، سپس رفتار خود را بر اساس آن تطبیق و تنظیم می کند.

همانطور که قبلاً ذکر شد زمان توسعه به فواصل زمانی منظمی تقسیم می شود که نام هر کدام از آن ها *تکرار* است. هر تکرار از مجموعه عملکردهایی تشکیل می شود که اعضای تیم جهت تکمیل پروژه انجام می دهند. این مجموعه عملکردها تقریباً در تکرار ثابت هستند اما در برخی از تکرارها بهتر از دیگر تکرارها انجام می شوند. کیفیت عملکردها به عوامل زیادی وابسته است که در تیم های مختلف متفاوت است. تفاوت این عوامل ناشی از شرایط مختلف توسعه، نوع تیم ها و نوع پروژه می باشد. اصول دقیقی وجود ندارد که تعیین کند تیم ها باید چگونه عملکردی داشته باشند. البته نباید هم وجود داشته باشد چرا که انعطاف پذیری و آزادی عمل از تیم در توسعه خواهد گرفت. از این رو بهتر است هرتیم خود این عوامل کیفی را در عملکرد خود بیابد و در پی اصلاح آن برآید.

همانطور که سابق بر این اشاره شد یکی از مزایای توسعه بر اساس تکرار(چند تکه) نسبت به توسعه یک تکه این است که تیم مجالی برای بررسی خود دارد. اعضای تیم ها بر اساس غریزه انسانی کمال طلبی خود همواره در پی بهبود عمکردهای گذشته شان هستند. بنابراین در بررسی های خود در انتهای هر تکرار به دنبال نقاط قوت و ضعف شان هستند. اما یافتن این بهبودها در عمل کاری است بس دشوار و زمان بر. در دنیای عملی یافتن ریشه مشکلات در برخی مواقع غیر ممکن است. اما بیشتر مشکلات دارای راه حل می باشند و تیم های اغلب سعی دارند تا رفتار و عملکرد خود را با آن راه حل ها تطبیق دهند. بدیهی است این بهبودها به یکباره صورت نخواهد پذیرفت و به زمان نیاز دارند. با انجام این فرآیند(بررسی عملکرد و تنظیم رفتار) تیم های شاهد بالا رفتن کیفیت عملکرد خود در توسعه خواهد بود. هر تیم با انجام این عملیات خود نظارتی می تواند به طور مستقیم شاهد افزایش کیفیت فرآیند توسعه خود باشد و همانطوریکه می دانید افزایش کیفیت فرآیند توسعه برابر است با افزایش کیفیت محصول نهایی(شکل 19-1).



شکل 19-1: شناسایی عوامل بهبود عملکرد در هر تکرار

هدف از طرح این مطرح شدن بیانیه آجیل این است که فرآیند توسعه سریع تر، مقرون به صرفه تر و البته با کیفیت تر انجام گیرد. در اصل این انتظاراتی است که توسعه دهندگان و مشتریان در عصر حاضر از یک فرآیند توسعه دارند و آنچه مسلم است فرآیند های سنتی دیگر قدرت ارضای این نیازها را ندارند. مولفان بیانیه آجیل بر این ادعا هستند که با پیاده سازی چهار ارزش بیانیه و دوازده اصل که حامی این ارزش ها هستند، فرآیند توسعه به سطح چالاکی بالایی خواهد رسید.

چالاک از منظر لغوی به معنای صحیح و سریع گام برداشتن می باشد. صحیح انجام دادن کارها بسیار مطلوب است و باعث می شود که نتیجه کار با کیفیت از آب در آید. اما درست انجام دادن امور در توسعه شرطی صرفاً لازم است ولی کافی نیست. این مهم است که کار با کیفیت درآید اما نه به قیمت از دست دادن زمان! تصور کنید پروژه ای به جای 3 سال پس از 10 سال به اتمام رسد و با فرض اینکه پروژه بسیار با کیفیت انجام شده است، اما پس از 10 سال این پروژه دیگر قابل استفاده نمی باشد و طول این مدت ممکن است جای خود را نرم افزارهای رقیب داده باشد. در این صورت با کیفیت بودن نرم افزار به تنهایی دردی را دوا نمی کند.

از طرف دیگر فدا کردن کیفیت برای تضمین زمان توسعه نیز منجر به موفقیت نخواهد شد. همانطور که قبلاً ذکر شد نرم افزار باید در وهله اول کار کند(ویژگی Working را تضمین کند) و نرم افزاری را می توان کار کننده نامید که دارای سطح کیفیت مناسبی باشد. سرعت با دقت نسبت عکس دارد و هرچه سرعت بالاتر رود کیفیت ممکن است تنزل پیدا کند پس در نتیجه عدم توازن بین سرعت با دقت به قیمت از دست دادن کیفیت تمام خواهد شد. از آن جایی که در فرآیند های پیشین یا بر روی دقت و درستی تاکید شده یا بر روی سرعت، چندان نتوانستند اعتماد توسعه دهندگان را جلب کند. بر این اساس توسعه دهندگان در راستای از بین بردن خلاءهایی که در فرآیند توسعه نرم فزار وجود داشت اصول و ارزش هایی ابداع کردند که اشتراک آنها را در بیانیه آجیل نظاره گردید. اگر با استفاده از روشی همه ارزش ها و اصول آجیل ارضا شود تضمین می شود که فرآیند توسعه به سطح بالایی از چالاکی خواهد رسید. اما ابداع یک روش(متد) کار چندان آسانی نیست و تجربه و آزمون نیاز دارد. از دهه 90 میلادی به بعد روش هایی جهت پیاده سازی آجیل معرفی شدند از جمله اسکرام، XP، Kanaban، Lean، کریستال و ... . هر کدام از این روش ها مزایا و معایب خاص خود دارند اما در یک چیز اشتراک دارند، آنها همگی اصول و ارزش های آجیل را پیاده سازی می کنند.

در این بین برخی متدها به دلیل طراحی خوب، گسترش بهتر و شهرت زیاد بیشتر مورد توجه توسعه دهندگان قرار گرفتند. در برخی اوقات نیز از ترکیب این متدها برای توسعه نرم افزار نیز استفاده می شود که به آن روش ترکیبی یا هیبرید گفته می شود. مشهورترین و به عقیده بسیاری کامل ترین متدی که در خانواده آجیل می توان یافت متد *اسکرام* است. این متد سال 1991 معرفی شد و از آن به بعد (به خصوص پس از تعریف بیانیه آجیل در سال 2000) بسیار توسعه یافت. این متد بیشتر در گروه های کوچک و متوسط استفاده می شود اما اخیراً نیز مورد توجه بسیاری از گروه های بزرگ به عنوان متدی درخور و مناسب توسعه های سریع قرار گرفته است. متد اسکرام با اصولی کاملاً متناسب با توسعه نرم افزار تعریف شده است از این رو آموزش آن بسیار سریع و راحت صورت می پذیرد. (از متد هایی که به صورت ترکیبی به کار برد می توان به اسکرام – XP و اسکرام – Lean اشاره کرد)

آموزه های این کتاب در راستای یادگیری پرطرفدارترین متد خانواده آجیل یعنی اسکرام ارائه می شود. در این کتاب سعی شده همه مباحث موجود در اسکرام پوشش داده شود. در این کتاب خواهید آموخت که اسکرام چیست؟ و چگونه باید آن را به کار ببندید تا در نهایت بتوانید گروهی درخور برای توسعه خودتان تشکیل دهید.

فصل دوم: اسکرام و مفاهیم آن

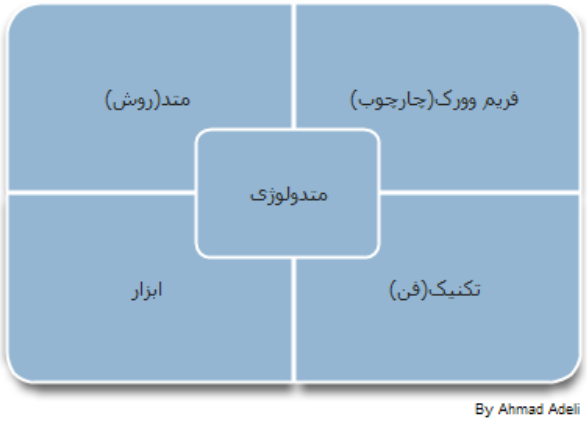
هر کسی با شنیدن اسم اسکرام اولین سوالی که ممکن است از خود بپرسد این است که اسکرام چیست؟ اسکرام در لغت نام نوعی بازی راگبی(فوتبال آمریکایی) است. در این بازی افراد تیم با همکاری نزدیک و البته زیاد سعی در کسب پیروزی دارند. متد توسعه نرم افزار اسکرام نیز چون به اصل همکاری بین اعضا پایه ریزی شده است قابل تشبیه به بازی اسکرام است.

برخی بر این عقیده اند که اسکرام یک فریم وورک یا چارچوب می باشد. این چارچوب در واقع قالبی برای توسعه نرم افزار می باشد و دارای قوانین و اصلی است که نرم افزار تحت آن ها توسعه می یابد. این دسته افراد بر این عقیده هستند که اسکرام چارچوب و قالبی را برای توسعه دهندگان آماده می سازد تا آن ها بتوانند نرم افزارشان را بر اساس آن توسعه دهند. این چارچوب نقش هایی را که در توسعه استفاده می شود تعریف می کند و بر اساس آن روابطی که این نقش ها با یکدیگر دارند را تدوین می کند. همچنین تعیین می شود که چه کارهایی و چگونه برای توسعه باید صورت گیرد. اما یک چارچوب به توسعه دهندگان دقیقاً نمی گوید از چه روشی برای توسعه استفاده کنند چرا که توسعه یک نرم افزار نوعی به روش های مختلفی می تواندصورت پذیرد. توسعه نرم افزار یک عمل انسانی است و نمی توان روش دقیقی برای آن تعریف نمود. بنابر آرا این دسته از از صاحب نظران باید اصطلاح فریم وورک و یا چارچوب را به اسکرام اطلاق کرد.

اما برخی دیگر از صاحب نظران، دیدگاه دیگری راجع به این موضوع دارند. آن ها بر این عقیده هستند که اسکرام به تنهایی خود یک متد(روش) توسعه نرم افزار می باشد این دسته عقیده دارند که اسکرام در حال حاضر تنها یک فریم وورک نیست بلکه روشی است قدم به قدم که توسعه دهندگان را تا انتهای توسعه همراهی می کنند. اسکرام روشی برای توسعه یک نرم افزار را تعریف می کند؛ تعیین می کند که چگونه کارها را اولویت بندی و سپس انتخاب کنیم و چگونه ارتباطات را سامان دهیم. بنابراین به دلیل این که اسکرام روش انجام دادن توسعه را ارائه می کند به جای یک فریم وورک بودن صرف می تواند یک متد توسعه نرم افزار باشد.

اگر به تعریف های فوق دقت شود مشاهده خواهد شد که هر دو گروه تفکر صحیحی راجع به اسکرام دارند. اختلاف این دو گروه ناشی از تعریف های متفاوت شان نسبت به روش یا متد است. گروه اول روش پیاده سازی نرم افزار را به عنوان متد تصور می کنند اما گروه دوم از روش به عنوان مراحل فرآیند توسعه یاد می کنند. در واقع بر اساس تعاریف هر دو نظر درست هستند و در آن ها اختلاف چندانی مشاهده نمی شود. تعاریف دیگری نیز از اسکرام وجود دارد که در نوع خود صحیح هستند(مثلاً گاهی اسکرام را مدل فرآیند نرم افزار می نامند که در نوع خود می تواند صحیح باشد). به عقیده نگارنده اگر بخواهیم تعریف کامل و جامعی از اسکرام داشته باشیم بهتر است بگوییم که اسکرام یک *متدولوژی* است.

معنای لغوی متدولوژی، روش شناسی است. متدولوژی از چهار آیتم فریم وورک(چارچوب)، متد(روش)، تکنیک(فن) و ابزار می باشد(شکل 1-2). بر این اساس که اسکرام نمونه ای از پیاده سازی آجیل می باشد می توان گفت که آجیل چارچوبی برای اجرای اسکرام می باشد. هر چارچوب یک سری اصول و قوانین را تعریف می کند و هر چه در این قالب جای می گیرد باید به این قوانین و اصول پایبند باشد. اسکرام در اصل پیاده سازی ارزش ها و اصول آجیل می باشد و رفتاری را تعریف می کند که این ارزش ها و اصول را ارضا کند. پیاده سازی اسکرام در حقیقت روشی است برای این که ارزش ها و اصول آجیل در قالب عمل در آیند. اسکرام یک روش کلی را تعریف می کند تا این ارزش های و اصول به شیوه درستی پیاده سازی شوند. فرآیند توسعه نرم افزار مانند هر فرآیند دیگری دارای مراحل ترتیبی است. اسکرام اجرای مرحله به مرحله را تا تکمیل فرآیند توسعه تعریف می کند. به زبانی دیگر اسکرام روشی را معرفی می کند تا بتوان توسعه را بر اساس اصول و قوانینی از پیش تعریف شده به سطحی از چالاکی رسانید. علاوه بر این اسکرام دارای تکنیک های بی نظیری در دل خود می باشد که از جمله آن ها می توان به تعیین اولویت کارها، توزیع نیروی انسانی، تخمین زمان و هزینه و... اشاره کرد.



شکل 1-2: مفهوم متدولوژی

اجرای این تکنیک ها از ملزومات مدیریت پروژه می باشد. تکنیک های فوق الذکر در حقیقت روش هایی برای انجام عملیات فرعی در توسعه نرم افزار می باشند. آیتم آخر یک متدولوژی که متاسفانه با وجود ارزش زیادی که برای توسعه دارد کمترین بها به آن داده شده ابزارها است. ابزارها بخش مهمی از یک متدولوژی محسوب می شوند. ابزارها به توسعه دهندگان کمک می کنند تا یک پروژه را بهتر اداره کنند. ابزارها را اغلب بر اساس روش ها و اصول یک متدولوژی می سازند از این رو هر متدولوژی ابزار خاص خود را دارا است. اما برخی ابزارها نیز وجود دارد که حالت عمومی دارند و برای بازه وسیعی از متولوژی ها به کار می روند. به هر حال تجربه ثابت کرده است که اعمال درون یک پروژه توسط ابزارهای مدیریتی بهتر اداره می شود. خوشبختانه اسکرام دارای ابزار های با کیفیت و جامعی است که می تواند هر نوع نیاز توسعه دهندگان را پاسخ دهد. بر اساس آیتم های ذکر شده اسکرام را می توان یک متدولوژی مدیریت پروژه با پکیجی کامل نام نهاد.

اسکرام متدولوژی است که با بسیاری از انواع پروژه های نرم افزاری چون برنامه های تجاری، برنامه های حیاتی-امنیتی، برنامه های داده محور و بازی های رایانه ای سازگار است. همچنین از بازه وسیعی از پروژه ها با مدت زمان مختلف پشتیبانی می کند. در عصر امروز توسعه سریع بیش از پیش اهمیت یافته از این رو بسیاری از متدولوژی های نوین (مانند اسکرام) بر این اساس(سرعت در توسعه) ساخته شده اند. یکی از بزرگ ترین مزایای اسکرام این است که از همه پلت فرم های موجود پشتیبانی می کند و محدود به سیستم های خاصی نمی باشد. برخی متدولوژی های مدیریت پروژه توسط شرکت های تجاری ساخته می شوند. هر شرکتی معمولاً این متدولوژی ها را بر اساس سیستم های خود طراحی کند. اگر چه ممکن است این متدها قدرت زیادی داشته باشند اما از آن جایی که برای پلت فرم خاصی طراحی شدند ممکن است برای دیگر پلت فرم ها چندان قابل استفاده نباشند.

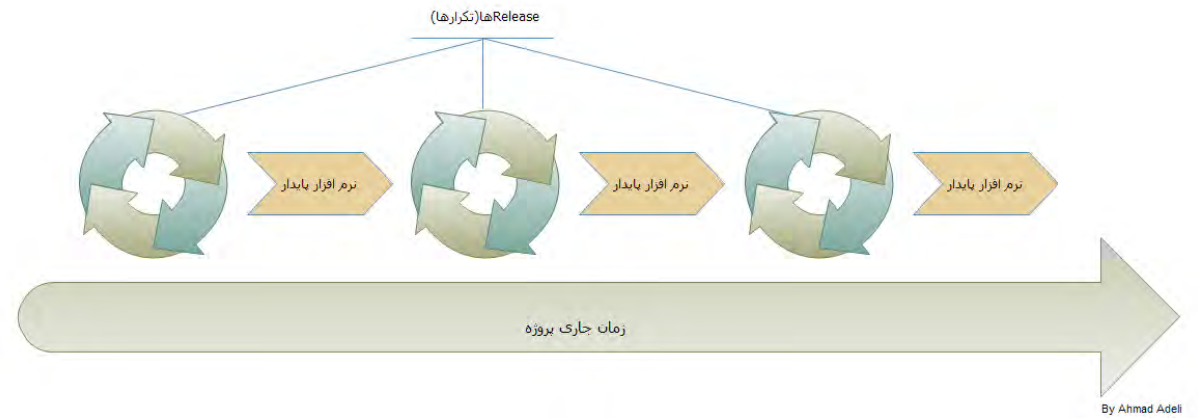
مدیریت منابع در اسکرام بسیار قدرتمند است. در اسکرام کل تیم به گروه های کوچکتری شکسته می شود. این گروه ها هر کدام به صورت مستقل کار می کنند و به صورت خود سازمانده مدیریت می شوند. اسکرام متدولوژی است که می توان آن را در شرکت های کوچک تا سازمان های بزرگ اجرا کرد. خود سازمانده بودن مدیریت گروه در اسکرام یکی از چالش های پیاده سازی آن در سازمان های سنتی است. سازمان ها اغلب از مدیریت بالا به پایین استفاده می کنند و همین باعث می شود برخی اوقات از این متدولوژی چندان استقبال نشود. از این رو توصیه شده که برای پیاده سازی اسکرام بهتر است کل سازمان به حالت چابک درآید به خصوص اگر تیم های توسعه درون سازمان مشغول به کار هستند؛ باید به گونه ای این تیم های و همچنین سازمان از این نظر هماهنگ باشد. بدیهی است حضور اسکرام به عنوان یک متدولوژی مدرن توسعه در یک مدیریت سازمان سنتی، ناهمخواهی ایجاد کرده و ممکن است مانند یک عضو پیوندی پس زده شود. بنابراین لازم است که در سازمانی که اسکرام در آن اجرا می شود قبلاً آموزه های لازم به خصوص برای افرادی که بیش از دیگران با تیم توسعه ارتباط دارند ارائه شود. خوشبختانه اسکرام از مفاهیم طبیعی و مطابق با دنیای واقعی استفاده می کند که خود باعث آموزش سریع تر و ساده تر افراد درگیر پروژه در سازمان می شود.

پیش از ورود به مباحث اصلی اسکرام ابتدا مروری کلی از چگونگی انجام اسکرام ارائه می شود تا یک آشنایی کلی از چند و چون اجرای اسکرام حاصل شود. اسکرام دارای مفاهیم و اصطلاحات زیادی است اما در این بخش

این مفاهیم در حد مختصری ارائه می شوند تا شرح مرور کلی ساده تر باشد. مفاهیم ارائه شده در این بخش در فصل های آتی به طور مفصل توضیح داده خواهند شد.

مرور کلی اسکرام

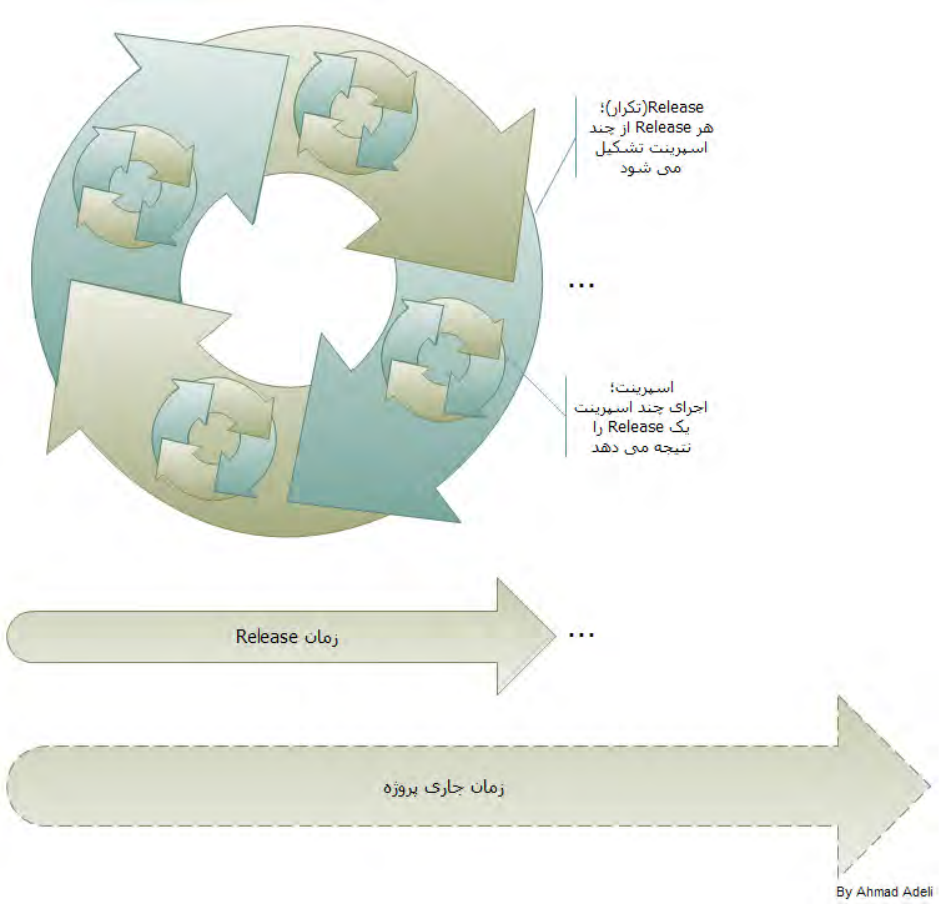
اسکرام مانند بسیاری از فرآیند های نوین توسعه نرم افزار از مفهوم توسعه بر اساس تکرار استفاده می کند. به عبارتی کل زمان توسعه به زمان های کوچکتری تقسیم می شود. کارهایی که در این زمان های کوچکتر صورت می پذیرد مانند یکدیگر هستند در نتیجه می توان هر کدام از آن ها را تکرارها یکدیگر دانست. در فصل اول بر روی این زمان ها کوچکتر نام تکرار را نهادیم. گفته شده که هر چه در طول یک توسعه انجام می شود در قالب تکرار ها قرار می گیرد. هر تکرار نسبت به تکرار قبلی خود قسمتی از پروژه را تکمیل می کند. خروجی همه آن ها نرم افزار کار کننده ای است که نسبت به حالت قبل خود(تکرار ماقبل) کامل تر است. در اسکرام به هر یک از این تکرارها یک Release می گوییم. هر پروژه به نسبت اندازه آن از تعدادی Release تشکیل می شود. تعداد این Releaseها به توافق گروه و مشتری بستگی دارد. پس از اتمام هر Release نرم افزار در دسترس کاربر قرار می گیرد تا از آن استفاده کند. باید پس از هر Release نرم افزار حالتی Stable(پایدار) داشته باشد تا بتواند در طول Release های آینده قابل استفاده باشد. این Release تا زمانی ادامه خواهند داشت که پروژه به طور کامل تکمیل گردد(شکل 2-2).



شکل 2-2: Release(تکرار)ها در اسکرام

Releaseها معمولاً بازه های زمانی بزرگی هستند مثلاً برای یک پروژه 4 ساله ممکن است هر Release 8 ماهه باشد یعنی حدوداً Release 6 وجود داشته باشد. زمان هشت ماه زمان زیادی است و می تواند کل زمان یک پروژه کوچک باشد. بدیهی است این زمان برای تکرارهای یک توسعه زمان مناسبی نخواهد بود.

در اسکرم تقسیم زمانی دیگری نیز وجود دارد. هر Release نیز به نوبه خود به تکرار های کوچکتری تقسیم می شود. به این تکرارهای کوچک تر اسپرینت گفته می شود. مرسوم است که هر اسپرینت زمانی بین دو تا هشت هفته را داشته باشد. این زمان به انتخاب تیم توسعه وابسته است. بعضی از تیم های توسعه مایلند زمانی کمتر از دو هفته و یا بیشتر از هشت هفته را برای یک اسپرینت داشته باشند. به دلیل انعطاف زیاد اسکرام این عمل شدنی است. معمول بر این است اسپرینت را برای راحتی در محاسبات زمانی بر اساس هفته حساب می کنند(مثلاً دو هفته، سه هفته و ...). اما تیم توسعه قادر است این واحد را به روز نیز تبدیل نماید. به طور کلی هیچ کدام از قوانین اسکرام به صورت قطعی لازم الاجرا نیستند و بر اساس هر تیمی قابل تغییر و سفارشی سازی می باشند. تکرارهای اسپرینت دقیقاً همان زمانی هستند که در آن تیم به اعمال مرتبط با توسعه می پردازد. هر کدام از اعضا به صورت روزانه کارهایی را اخذ کرده و سعی در به اتمام رسانی آن کار دارند(شکل 2-3).

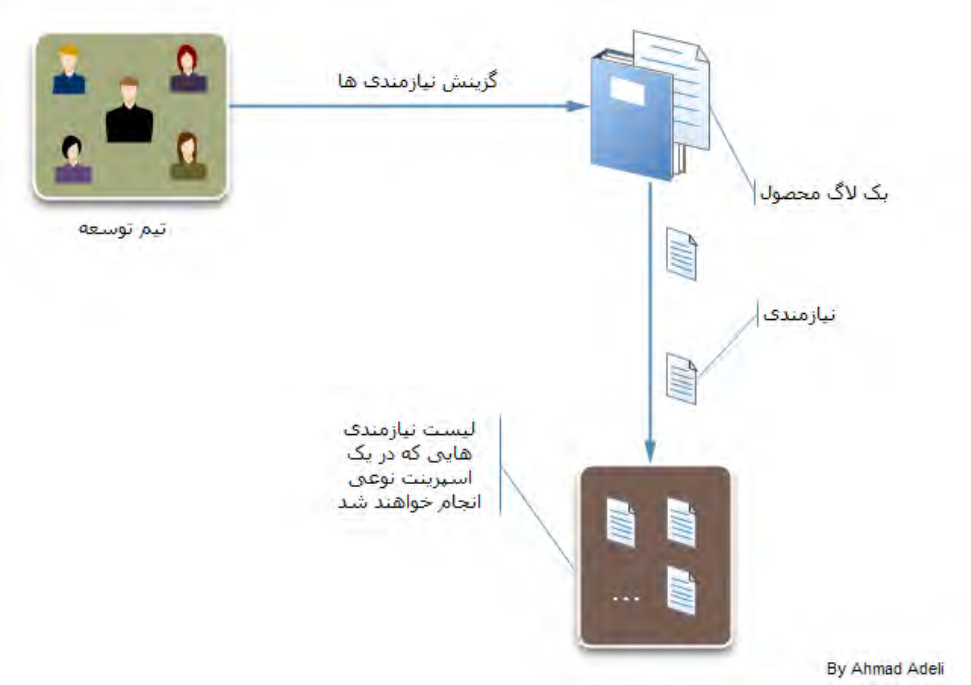


شکل 3-2: مفهوم Release و اسپرینت

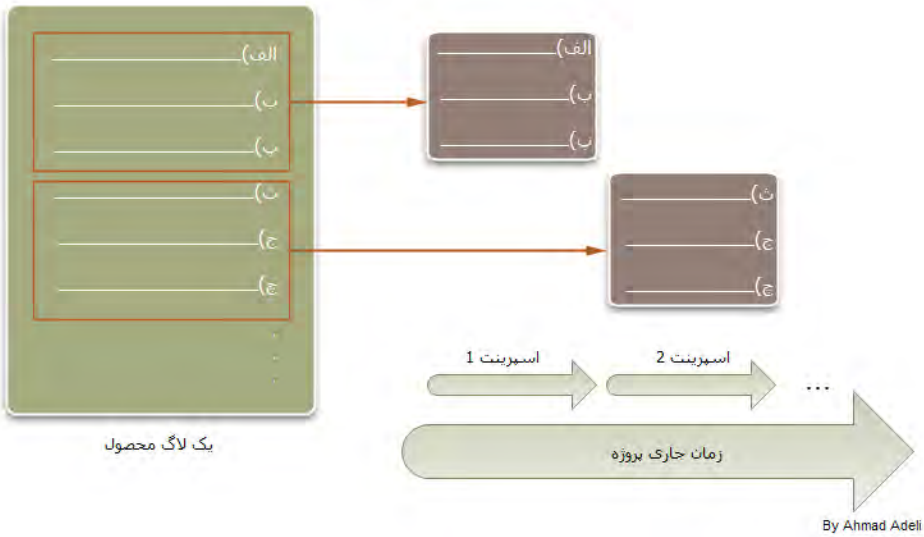
پیش از اجرای اسپرینت باید کارهایی که قرار است در آن انجام شود مشخص شوند. این کارها به صورت نیازمندی ها تعیین می شوند. لیستی کلی از این نیازمندی ها وجود دارد که همه نیازمندی هایی که باید ارضا شوند را در خود جای می دهد. این لیستِ نیازمندی ها را اصطلاحاً بک لاگ محصول می گویند. بک لاگ محصول لیستی از مشخصات و نیازمندی هایی است که باید پیاده سازی شوند. هر نیازمندی ای که تشخیص داده شود و بین تیم و مشتری روی آن توافق شود به این لیست افزوده خواهد شد. به این لیست به این دلیل بک لاگ *محصول* گفته می شود که نیازمندی های کلی یک محصول در آن جای می گیرد(شکل 4-2).

شکل 4-2: بستن بک لاک محصول

با فرض داشتن یک بک لاگ محصول کامل، پیش از اسپرینت طی جلسه ای اعضای تیم به صورت توافقی قسمتی از این نیازمندی ها را انتخاب نموده تا در طول اسپرینت تکمیل شوند(شکل 5-2). تیم موظف است در پایان اسپرینت نیازمندی های اخذ شده را تکمیل کند و پس از هر اسپرینت نرم افزاری غنی تر تحویل دهد. تیم ها اغلب سعی می کنند که بیشترین کار را گزینش کنند تا بتوانند در نهایت تحویل پربارتری داشته باشند. تیم باید خواه برای اسپرینت جاری و خواه برای اسپرینت های آتی برنامه ریزی مناسبی داشته باشد تا بتواند به موقع و در زمان معین پروژه را به اتمام برساند. نیازمندی هایی که در یک اسپرینت از بک لاگ محصول انتخاب می شوند لیستی را خواهند ساخت که در اسکرام به آن بک لاگ اسپرینت گفته می شود. آیتم های یک بک لاگ اسپرینت زیر مجموعه ای از آیتم های لیست اصلی یعنی بک لاگ محصول هستند. به عبارت دیگر بک لاگ اسپرینت نیازمندی هایی است که به صورت جاری در حال پیاده سازی هستند(شکل 6-2).



شکل 5-2: بستن بک لاگ اسپرینت



شکل 6-2: انتقال آیتم های بک لاگ محصول به بک لاگ هر اسپرینت

تیم توسعه در اسکرام افراد محدودی دارد چرا که از آن جایی این تیم خودسازمانده می باشد کوچک بودن گروه به مدیریت پذیر بودن آن کمک می کند. معمولاً در تیم از انواع تخصص ها چون برنامه نویس، آزمون گر(تستر)،

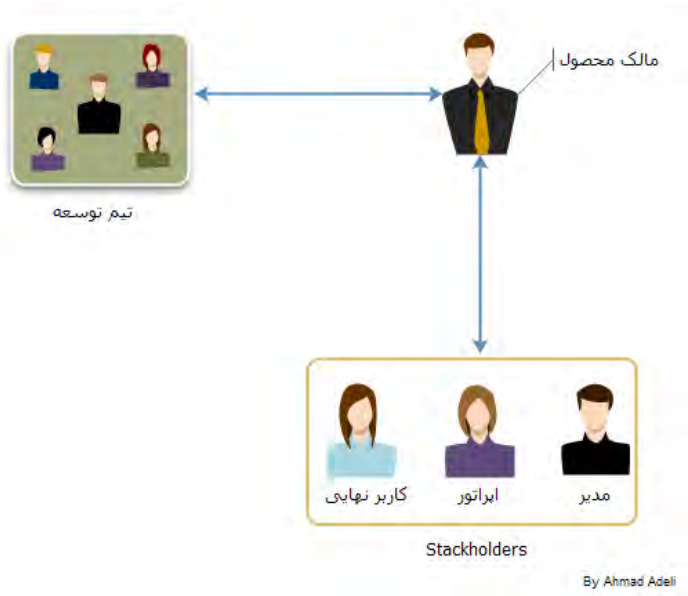
معمار نرم افزار، تحلیل گر، طراح UI(طراح رابط کاربری) و ... که در توسعه نرم افزار مرسوم است استفاده می شود. اما این نقش ها همه نقش های موجود در اسکرام نیست. تیم توسعه علاوه بر آن ها دارای دو نقش دیگری نیز می باشد که خاص اسکرام است: اسکرام مستر(رهبر اسکرام) و مالک محصول. متاسفانه در برخی منابع اسکرام مستر با مدیر اسکرام معادل سازی شده است که علاوه بر ایجاد شبهه به دلیل وجود واژه مدیر، واژه ای است نامناسب. همانطور که پیش از این ذکر شد گروه های اسکرام هایی خودسازمانده هستند و دارای شخص خاصی به عنوان مدیر نمی باشند. نقش اسکرام مستر در تیم اسکرام رهبری است(شکل 7-2). اسکرام مستر سعی می کند تیم در مسیر درستی گام بردارد؛ او عملاً مسئول اجرای مستقیم اسکرام در تیم توسعه می باشد. اگر کل تیم و حتی افرادی که از خارج با تیم در ارتباط اند به اسکرام اشراف داشته باشند باز هم بدون وجود یک رهبر نمی توان اجرای اسکرام را در مسیر صحیحی کنترل نمود.

شکل 7-2: تیم توسعه

اسکرام متدولوژی قدرتمندی است اما به این معنی نیست که خودکار است. فرآیند توسعه یک فرآیند انسانی است. از این رو شخصی به طور اخص باید مسئول اجرای توسعه نرم افزار باشد. در دیگر فرآیند ها این وظیفه را مدیر پروژه انجام می دهد. مدیر پروژه تنها مسئول اجرای توسعه نیست بلکه او مسئولیت هایی چون کنترل نیروی انسانی، کنترل برآورد، اخذ تصمیمات استراتژیک و ... را نیز بر عهده دارد. اما در اسکرام، اسکرام مستر مسئولیت های محدودی دارد و بیشتر مشغول نظارت بر اجرای صحیح اسکرام می باشد. معمول بر این است شخصی از تیم به عنوان اسکرام مستر انتخاب می شود که بیشترین دانش را در اجرای اسکرام دارد.

مالک محصول نقش دیگری است در اسکرام که جایگاهی ویژه دارد. بر خلاف نامی که برای آن انتخاب شده مالک محصول به هیچ وجه صاحب امتیاز پروژه نمی باشد. اگر بخواهیم به نحوی این نقش را با نقش های دیگر متدولوژی ها مقایسه کنیم مالک محصول را می توانیم با کمی تساهل مهندس نیازها بنامیم. اما مهندس نیازها تنها یک بخش از این نقش در توسعه است. مالک محصول در حقیقت پل ارتباط تیم و مشتری می باشد. در همین راستا دو وظیفه دارد: خواسته های مشتری را به نیازمندی هایی برای توسعه تبدیل کند، این نیازمندی های را به تیم توسعه انتقال دهد. مالک محصول مسئول اصلی تکمیل پروژه است و سخت می کوشد تا خواسته های مشتری را به قابلیت های نرم افزار تبدیل کند. او مسئول اصلی کاستی ها و نقص های احتمالی پروژه خواهد بود. مالک محصول مانند پیمان کاری است که از مشتری توسه یک نرم افزار را تحویل گرفته و از تمام منابع بهره می برد تا پروژه را به موقع و مطابق خواسته های مشتری ارائه دهد. نکته مهم اینجاست که با این همه او عضوی از تیم توسعه است و مالکیت معنوی و حقوقی او در پروژه همانند دیگر اعضا می باشد.

تیم در طول توسعه با افراد زیادی سر و کار دارد. بدیهی که کل اعضای تیم همگی نمی توانند با افراد خارج از پروژه ارتباط مستقیم داشته باشند. از این رو مالک محصول وظیفه دارد از افرادی چون مدیران، کاربران نهایی، اپراتورها و Stakeholder ها نیازمندی ها را استخراج کند و به تیم توسعه انتقال دهد.به طور کلی هر کدام از افراد فوق الذکر در بخشی ا توسعه حضوری کوتاه خواهند داشت اما آنچه مسلم است ارتباط دائمی مالک محصول با همه این افراد است(شکل 8-2).



شکل 8-2: ارتباط اعضای اسکرام

همانطور که برای ورود نیازمندی ها به فرآیند لیستی وجود دارد(بک لاگ محصول)، لیستی خروجی نیز برای کارهای انجام شده به عنوان لیست پیاده سازی ها وجود دارد. این لیست عیناً مانند لیست بک لاگ محصول است با این تفاوت که در بک لاگ محصول قابلیت های بالقوه وجود دارد(آیتم هایی که قرار است انجام شوند) ولی لیست پیاده سازی ها دارای قابلیت هایی بالفعل است(آیتم هایی پیاده سازی شده اند). به بیانی دیگر هر گاه آیتمی از بک لاگ محصول انتخاب شود در صورت انجام کامل آن و تایید توسط افراد ذی ربط، آن آیتم از لیست بک لاگ محصول به لیست پیاده سازی ها منتقل می شود. مشابه با لیست پیاده سازی های یک بک لاگ محصول لیستی برای بک لاگ اسپرینت وجود دارد که آیتم های انجام شده را در آن قرار می گیرد. تیم برای هر اسپرینت بخشی از بک لاگ محصول را که قرار است در طول اسپرینت روی آن کار شود انتخاب می کند و در بک لاگ اسپرینت قرار می دهد. پس از تکمیل هر کدام از آیتم های بک لاگ اسپرینت از آن لیست حذف شده به لیست پیاده سازی های یک اسپرینت اضافه می شوند. انتظار می رود که لیست پیاده سازی های یک اسپرینت عیناً با بک لاگ آن اسپرینت برابری کند. به این معنی که نیازمندی ها در طول اسپرینت مذکور پیاده سازی شوند. ناگفته پیدا است بک لاگ اسپرینت و لیست پیاده سازی های اسپرینت به ترتیب زیر مجموعه بک لاگ و لیست پیاده سازی های محصول می باشند.

اسکرام یک متدولوژی جلسه محور است. بیشتر ارتباطات تیم از طریق جلسه صورت می پذیرد. این ارتباطات یا درون تیم می باشد یا خارج از آن. بر اساس اصول بیانیه آجیل که ارتباط دائم را متذکر می شود؛ اسکرام نیز از طریق جلسات مکرر به این اصول جامه عمل می پوشاند. جلسات اسکرام اغلب کوتاه مدت می باشند تا با تکرار آن ها موجب آزردگی خاطر افراد حاضر در جلسات نشود و هم این که اتلاف وقت کمتری را سبب شود.

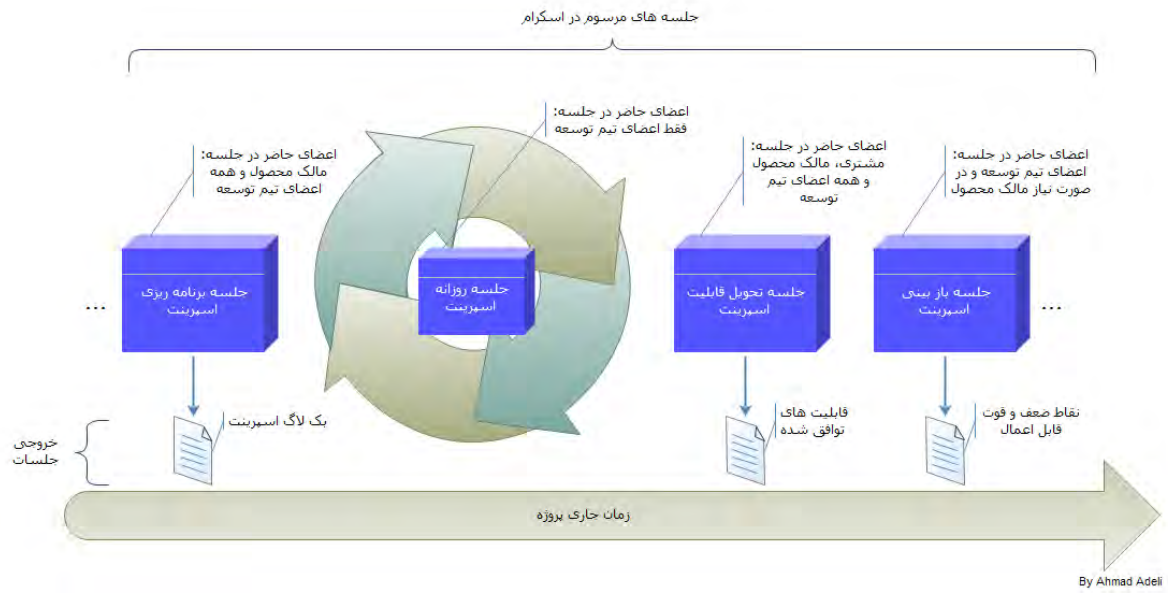
چهار نوع جلسه اصلی برای اسکرام وجود دارد که لازم الاجرا است. مهمترین جلسه پیش از شروع هر اسپرینت تشکیل می شود که به آن جلسه برنامه ریزی اسپرینت گفته می شود. انتخاب آیتم هایی که در بک لاگ اسپرینت آماده می شود در این جلسه صورت می پذیرد. این جلسه مهمترین جلسه هر اسپرینت می باشد و حضور همگی اعضا را می طلبد. جلساتی نیز وجود دارد که به آنها جلسات روزانه اسکرام گفته می شود. همانطور که از نام آن ها پیدا است این جلسات هر روز در طول یک اسپرینت انجام می شوند. طول این جلسات بسیار کوتاه است و هدف از انجام آن این است که افراد تیم ارتباطی دائمی و اجباری همه روزه باید یکدیگر داشته باشند تا مسائل را عنوان و بررسی نمایند. این جلسه با مروری بر کارهای روز قبل هر شخص و کارهای روز جاری او دنبال می شود. پس از آنکه همه توسعه دهندگان گزارش خود را شفاهاً ارائه نمودند تیم به طرح و رفع ابهام های احتمالی در توسعه می پردازد. گفتنی است این جلسه برای هر روز توسعه الزاماً باید تکرار شود.

پس از پایان هر اسپرینت دو جلسه دیگر نیز صورت می پذیرد: جلسه تحویل قابلیت و جلسه بازبینی. جلسه تحویل قابلیت با حضور همه اعضای تیم توسعه، اسکرام مستر، مالک محصول، مشتری یا نمایندگان مشتری انجام می شود. در این جلسه نیازمندی هایی که در اسپرینت سپری شده پیاده سازی شدند به صورت قابلیت

های یک نرم افزار ارائه می شوند. این جلسه در حقیقت توافقی است بر سر آنچه پیاده سازی شده به نحوی در صورت عدم تایید مشتری تیم مجبور است مجدداً در اسپرینت های آینده روی آن کار کند.در این جلسه تیم به ارائه آنچه انجام داده است می پردازد و مشکلات و مسائل آتی توسعه را همراه با مشتری بررسی می کند. مشتری نیز ممکن است نیازمندی های جدیدی را پیشنهاد کند و یا از نیازهایی که قبلاً درخواست داده صرف نظر نماید.

همچنین مشتری ممکن است پیاده سازی یک نیاز را به عللی نپذیرد و در خواست اصلاح و تجدید نظر در پیاده سازی آن نیاز را داشته باشد. همه این مسائل در جلسه تحویل قابلیت مطرح خواهند شد. اعضای جلسه بسیار امیدوار هستند که در یک جلسه همه مسائل مطرح شده و بتوانند درباره آن ها به تصمیم گیری بپردازند. چرا که بر خلاف جلسات قبلی، انجام این جلسه بسیار مشکل و زمان بر است. آخرین نوع جلسه ای که در این بخش بررسی خواهد شد جلسه بازبینی است. در فصل اول این جلسه به طور صریح به عنوان یکی از اصول بیانیه آجیل ارائه شد. در اسکرام این جلسه بسیار پر اهمیت است چرا که موجب ارتقاء کیفی توسعه خواهد شد. این جلسه که مختص اعضای توسعه تیم توسعه است پس از جلسه تحویل قابلیت (یا یک روز پس از آن) صورت می پذیرد.

در این جلسه در خصوص پروژه بحثی مطرح نمی شود بلکه تیم به بررسی و بازبینی عملکرد خود می پردازد. تیم بر سر این که در اسپرینت گذشته چگونه عمل شده و باید چگونه عمل می شد به گفتگو می پردازد. تیم توسعه سعی بر آن دارد تا با استفاده از این جلسه نقاط ضعف خود را در اسپرینت قبلی از لحاظ عمکرد بشناسد و آن را در اسپرینت های بعدی بر طرف کند. اعضای تیم بدین شکل بر چگونه بهتر شدن نظارت می کنند و در تلاش خواهند بود تا برای آن راه حلی مناسب بیابند. علاوه بر آن سعی می شود تا با اصلاح رفتار در فرآیند توسعه نقاط قوت تیم تقویت شود(شکل 9-2).



شکل 9-2: جلسات اسکرام

در اسکرام علاوه بر نیازمندی ها پیاده سازی شده که به صورت قابلیت ارائه می شود خروجی های دیگری نیز وجود دارد. این خروجی ها مستنداتی هستند که به طور مستقیم دارای ارزش تجاری نیستند. به این معنی که که نه تنها مشتری آن را طلب نکرده و همچنین درک شان نمی کند بلکه از نظر تجاری و مالی این خروجی را (مستندات) را دارای ارزش نمی داند. از این مستندات صرفاً برای کمک به توسعه و اجرای اسکرام استفاده می شود. به این گونه مستندات اصطلاحاً مصنوعات پروژه گفته می شود. مصنوعات تاثیر مستقیمی در نرم افزار کار کننده نخواهند داشت و فقط برای مدیریت بهتر اسکرام مورد استفاده قرار می گیرند. دو نوع از انواع مصنوعاتی که در اسکرام وجود دارد پیش از معرفی شدند: بک لاگ محصول و بک لاگ اسپرینت. همانطور که مشاهده شد در حقیقت مشتری آن ها عضوی از نرم افزار نمی داند و برای آنها وجهی نمی پردازد و استفاده از آن ها محدود به تیم توسعه می باشد. علاوه بر این دو نوع، نوع دیگری از این مصنوعات در ادامه معرفی می شود و توضیح سایر مصنوعات به فصل های آتی موکول می شود.

اسکرام یک متدولوژی خودناظر است در اسکرام تکنینک هایی وجود دارد که تیم می تواند عملکرد جاری خویش را نظاره، بررسی و بر اساس آن تصمیم گیری کند. این نظارت عموماً از طریق گزارشات صورت می گیرد. هر گزارش در اسکرام در زمان خاصی انجام می شود. مهمترین گزارش عملکردی که اسکرام از آن برای کنترل اجرای توسعه استفاده می کند، نمودار Sprint Burn Down می باشد. این نمودار روزانه به روزآوری می شود تا آخرین تغییرات را منعکس کند. نمودار Sprint Burn Down نسبت کار انجام شده به زمان است و تیم از طریق آن در می یابد که آیا پیاده سازی نیازمندی ها مطابق انتظار پیش بینی شده پیش می رود یا خیر. اگر کارهای کمی آماده و پیاده سازی شوند، نمودار در طول یک اسپرینت این شرایط را به وضوح نشان داده خواهد داد. از طرفی اگر کارها سریع تر از حد انتظار تمام شوند باز هم نمودار این حالت را نیز گوش زد خواهد کرد. مزیت این نمودار این است که گزارش لحظه به لحظه و به روزی را برای تیم مهیا می کند(لحظه به لحظه به این معنا نیست که نمودار باید هر لحظه به روز شود، این نمودار معمولاً به صورت روزانه به روز خواهد شد). نمودار Sprint Burn Down مختص یک اسپرینت می باشد اما برخی تیم ها از آن برای یک Release نیز استفاده می کنند. به این صورت که نمودار پس از هر اسپرینت به روز می شود. تا واقعیت را با آنچه مورد انتظار تیم است تطبیق دهد. به این ترتیب می توان یک کنترل نظارتی روی هر Release اعمال نمود.

به طور کلی باید گفت که مصنوعات جزئی از یک متدولوژی می باشند. گرچه مصنوعات پروژه از نظر تجاری دارای ارزش نمی باشند اما نقش مهمی در توسعه نرم افزار ایفا می کنند.

ممکن است پرسیده شود که چرا در اسکرام از تکرارهای تودرتو استفاده شده است؟ در فصل اول گفته شده ارائه نرم افزار کار کننده در بازه زمانی الزامی می باشد. ما این بازه های زمانی را تکرار نام نهادیم. گفته شد که پس از هر تکرار نرم افزار کار کننده باید وجود داشته باشد. همچنین گفته شد که هر تکرار نباید خیلی طولانی باشد و هر چه کوچک تر باشد مطلوب تر است. اما در این جا تناقضی وجود دراد. اساساً یک نرم افزار کار کننده نرم افزاری است پایدار به طوری که هر قابلیتی که به مرور به آن اضافه می شود نه تنها باید خود پایدار باشد بلکه باید پایداری کلی نرم افزار را نیز تضمین کند.

برای ساخت نرم افزاری پایا بدیهی است که تیم توسعه به زمان احتیاج دارد و در مدت های کوتاهی چون تکرارها نمی توان کار کنندگی و پایدار بودن نرم افزار را تضمین نمود. در اسکرام برای تحصیل همه مزیت ها دو نوع تکرار معرفی شده است. تکرار اول که تکرار بزرگ تر می باشد همان Release است. در پایان مدت زمان این تکرار تیم می تواند یک نرم افزار کار کننده و پایدار ارائه کند چرا که به اندازه کافی بزرگ می باشد که ساخت و ارائه نرم افزار کار کننده در آن تضمین شده باشد(به عنوان مثال هشت ماه). ناگفته پیدا است که این زمان بزرگ تر آن است که از مزایای توسعه بر اساس تکرار بهره ببرد. بنابراین در اسکرام تکرارهای کوچک تری در دل تکرار Release قرار داده شده که همان اسپرینت می باشد. اسپرینت ها زمان بسیار مناسبی برای توسعه بخش بندی شده دارند اما نمی توانند نرم افزاری پایا در انتهای زمان کوچک شان ارائه کنند.

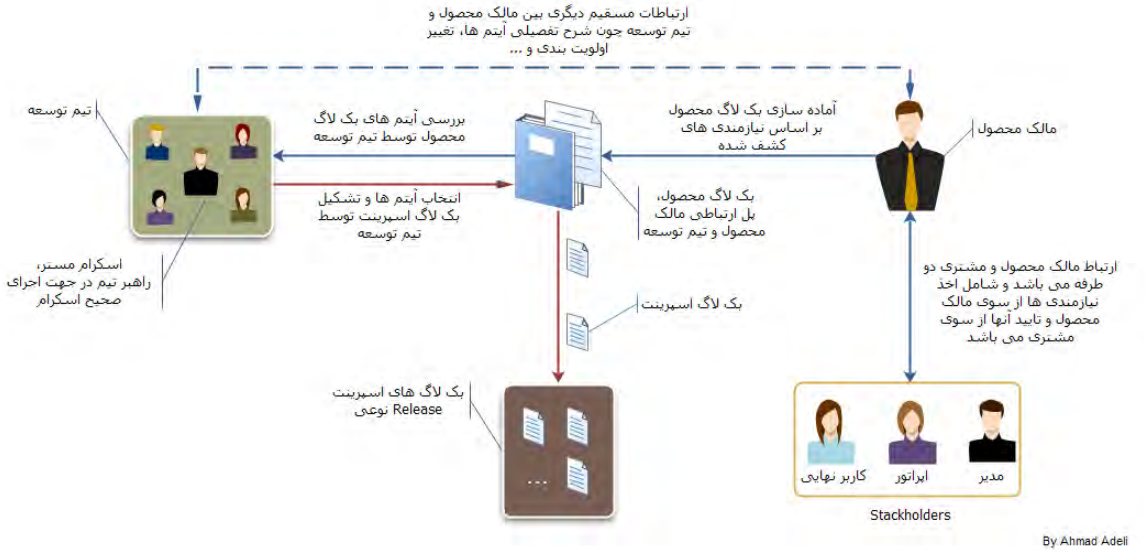
بخش مهمی که در فصل یک در کنار توسعه بر اساس تکرار شرح داده شد، توسعه افزایشی می باشد. توسعه افزایشی به این معنا است که نرم افزار به مرور و در بازه های منظم کامل می شود و به قابلیت های آن اضافه می گردد. از این رو وظیفه یک اسپرینت با توجه به زمان کوتاه آن ارائه این قابلیت های افزایشی است. از آن جایی که مجموعه اسپرینت ها یک Release را می سازند پس مجموعه این قابلیت های افزایشی در طول یک Release به نرم افزار اضافه شده و خروجی Release را به یک نرم افزار کار کننده، پایدار بهبود یافته تبدیل می کند. ممکن است در قسمت هایی از توسعه برخی اسپرینت ها، قابلیت هایی افزایشی را ایجاد نکنند و در عوض خروجی آن ها توسعه نرم افزاری کمکی و یا زیر ساختی برای نرم افزار اصلی باشد. این خروجی به طور قطع برای ارائه چندان مناسب نیست اما از آنجایی که احتمال پشت سر هم بودن اسپرینت های این چنینی بسیار پایین است پس بر خلاف تکرار های نوع اسپرینت، تکرارهای نوع Release به طور حتم دارای خروجی مناسب و قابل ارائه ای می باشند.

چگونگی اجرای اسکرام

در بخش قبل به چیستی و مفاهیم اولیه آن تا حدودی اشاره شد. حال خواهیم دید که این متدولوژی چگونه اجرا می شود و چگونه می توان توسط آن یک پروژه را اداره نمود.

فرض شود که پروژه ای اخذ شده و مقدمات آن آماده شده است تیم توسعه آن، در مرحله پیش از اجرا به سر می برد. برای شروع کار تیم نیازمند مشخصات نرم افزار است. مالک محصول وظیفه جمع آوری این مشخصات را دارد. او سعی می کند طی جلساتی که با مشتری (اعم از مدیران، اپراتورها وکاربران) می گذارد تا حد ممکن

مشخصات نرم افزار را استخراج کند. مالک محصول نیازهای مشتری جستجو می کند و از طریق این جلسات و مستندات موجود در سازمان، مشخصات نرم افزاری که باید ساخته شود را کشف کرده و به تیم توسعه انتقال می دهد. این انتقال مشخصات از طریق پل ارتباطی رسمی مالک محصول و تیم توسعه یعنی بک لاگ محصول صورت می گیرد. مالک محصول کارها و قابلیت هایی را که نرم افزار باید از آن پشتیبانی کند به طور مستقیم وارد بک لاگ محصول می کند. بدیهی است که آیتم هایی که به بک لاگ محصول می شوند قبلاً توسط مشتری درخواست و تایید شده اند(شکل 10-2).



شکل 10-2: مدل کلی ارتباطات درون اسکرام

مالک محصول نه تنها در ابتدای پروژه بلکه در طول توسعه نیز بک لاگ محصول را تغذیه می کند. آیتم هایی که در بک لاگ محصول قرار دارند صرفاً تعیین می کنند که قابلیت های نرم افزار چگونه باید باشند. شیوه پیاده سازی قابلیت ها در آیتم ها بک لاگ قرار نمی گیرد چرا که این آیتم ها توسط مالک محصول تهیه می شوند و این موضوع از اختیارات او خارج است. در اسکرام همه وظایف به طور دقیق برای هر نقشی تعریف شده است تا کمترین تداخل بین آن ها به وجود آید.

با شروع توسعه تیم باید طول مدت هر Release را تعیین نماید. هر Release نرم افزاری کار کننده را ارائه میکند از این رو مدت زمان هر Release با برنامه ریزی کاری مشتری ارتباط مستقیم دارد. بنابراین تیم سعی می کند با مشتری برای تعیین این مدت به توافق برسد. پس از تعیین بازه هایی که در آن نرم افزار ارائه می شود(Release)، تیم باید مدت زمان هر اسپرینت را نیز مشخص کند. مدت اسپرینت را تیم می تواند بدون دخالت مشتری انتخاب کرده و حتی در طول توسعه تغییر دهد. معمولاً بنابر تجربه خود و بر یک تصمیم گروهی مدت زمان هر اسپرینت را معین می کند.

پس از تعیین این مدت تیم بر اساس آن، کارهایی را که باید انجام شود از بک لاگ محصول استخراج کرده و در بک لاگ اولین اسپرینت قرار می دهد. در اسپرینت ها ابتدایی احتمالاً نرم افزار قابل ارائه ای وجود نخواهد داشت و تیم بیشتر وقت خود را صرف تحلیل و طراحی زیر ساخت نرم افزار خواهد کرد اما در اسپرینت های آتی نرم افزاری با قابلیت های افزایشی ارائه می شود. برای اخذ کارها در یک اسپرینت تیم از تخمین سرعت استفاده می کند. به این معنی که تخمین زده می شود با تعداد افرادی که در اسپرینت حاضر هستند، تیم تحویل چه مقدار کار را تضمین می کند. این عمل در جلسه برنامه ریزی اسپرینت صورت می گیرد. این جلسه با حضور همه اعضای تیم و به خصوص مالک محصول صورت می پذیرد. برای تعیین کارهایی که باید در یک اسپرینت انجام شود مالک محصول و تیم توسعه باید به توافق برسند. پس از انجام پذیرفتن توافقات، بک لاگ اسپرینت بسته خواهد شد. تیم توسعه تمرکز خود را در طول اسپرینت روی آیتم هایی که در بک لاگ اسپرینت وجود دارند قرار می دهد و سعی می کند طبق زمان تعیین شده (طول اسپرینت) این آیتم ها را به پایان برساند. تیم در طول یک اسپرینت به عملیات معمول در یک توسعه می پردازد. از نمونه های این عملیات می توان به تحلیل، طراحی، معماری، طراحی داده، طراحی UI، طراحی قطعات، برنامه نویسی، تست، یکپارچه سازی، Build و ... اشاره کرد. در حقیقت تیم توسعه نرم افزار کوچکی را در اسپرینت خواهد داشت.

اسکرام مستر که اغلب خود عضوی از تیم توسعه است سعی می کند فرآیند اسکرام را در مسیری درست قرار دهد و از کج روی آن به ایجاد مشکل ختم می شود جلوگیری کند. تیم در پایان هر اسپرینت، جلسه ارائه قابلیت را با حضور همه اعضای خودش، مالک محصول و مشتری برگزار کند. در این جلسه اعضای تیم به ارائه کارهای انجام شده می پردازد و قابلیت های اضافه شده را معرفی می کند. مشتری این قابلیت ها را مشاهده کرده و در صورت موافقت تایید می کند. پس از این جلسه تیم جلسه بازبینی را انجام می دهد و در آن به بررسی عملکرد خود در طول اسپرینت قبل می پردازد و پس از آن با ارائه راه حل سعی در بهبود این عملکرد دارد. پس از جلسه بازبینی تیم خود را برای اسپرینت آینده آماده می کند. مالک محصول در طول هر اسپرینت همچنان به جمع آوری و اصلاح نیازها و مشخصات محصول می پردازد. تیم مجدداً عملیات فوق الذکر را برای اسپرینت های بعدی تا زمانی صورت می دهد که Release به پایان برسد. پس از پایان Release تیم یک نرم افزار پایدار و کارکننده را به مشتری تحویل می دهد. برای این تحویل نرم افزار نیز جلساتی با مشتری برگزار می شود. بدیهی است که پس از این ارائه در صورت ادامه پروژه تیم وارد Release بعدی خواهد شد.

با توجه به مطالب مذکور اسکرام را می توان به یک ماشین خودکار تولید نرم افزار تشبیه کرد به طوری که این ماشین با استفاده از منابعی از جمله منابع انسانی، مشخصات یک نرم افزار را از ورودی گرفته و با صرف زمانی معین یک نرم افزار کارکننده را به خروجی می فرستد(شکل 11-2). این تشبیه ملموس ترین تشریحی است که می توان از این متدولوژی توسعه نرم افزار ارائه کرد. فرآیندی که گفته شد اسکرام را به طور کلی توصیف می کند، فصل های آینده جزئیات کامل تر و جامع تری از اسکرام پوشش داده خواهد شد.

شکل 11-2: مدل ماشین خودکار اسکرام

چگونگی شروع یک پروژه

پیش از شروع اولین اسپرینت لازم است تا یک سری کارهای مقدماتی روی پروژه صورت بگیرد. این کارهای مقدماتی باعث آمادگی تیم در روند توسعه خواهد شد. برخی تیم ها مقدمات پروژه را در قالب اسپرینت ها انجام می دهند. این باعث می شود که اسپرینت های ابتدایی خروجی متفاوت تری از بقیه اسپرینت ها داشته باشند که خود سبب از بین رفتن یکپارچگی در توسعه خواهد شد. از طرف دیگر این گونه کارهای مقدماتی اصولاً به همه توسعه مربوط می شود و در کل اسپرینت های تاثیر گذار است و به نوعی زیر ساخت توسعه را تشکیل می دهند. پس بهتر است این گونه عملیات پیش از شروع اسپرینت ها صورت گیرد. کارهای مقدماتی هر پروژه با پروژه ای دیگر بر اساس نوع آن متفاوت است. همچنین هر گروه و سازمانی معمولاً قوانین و اصول خود را برای آن اجرا می کند. در این بخش برخی کارهای اساسی و معمول کع در اکثر پروژه ها مرسوم است شرح داده می شود:

تعریف پروژه

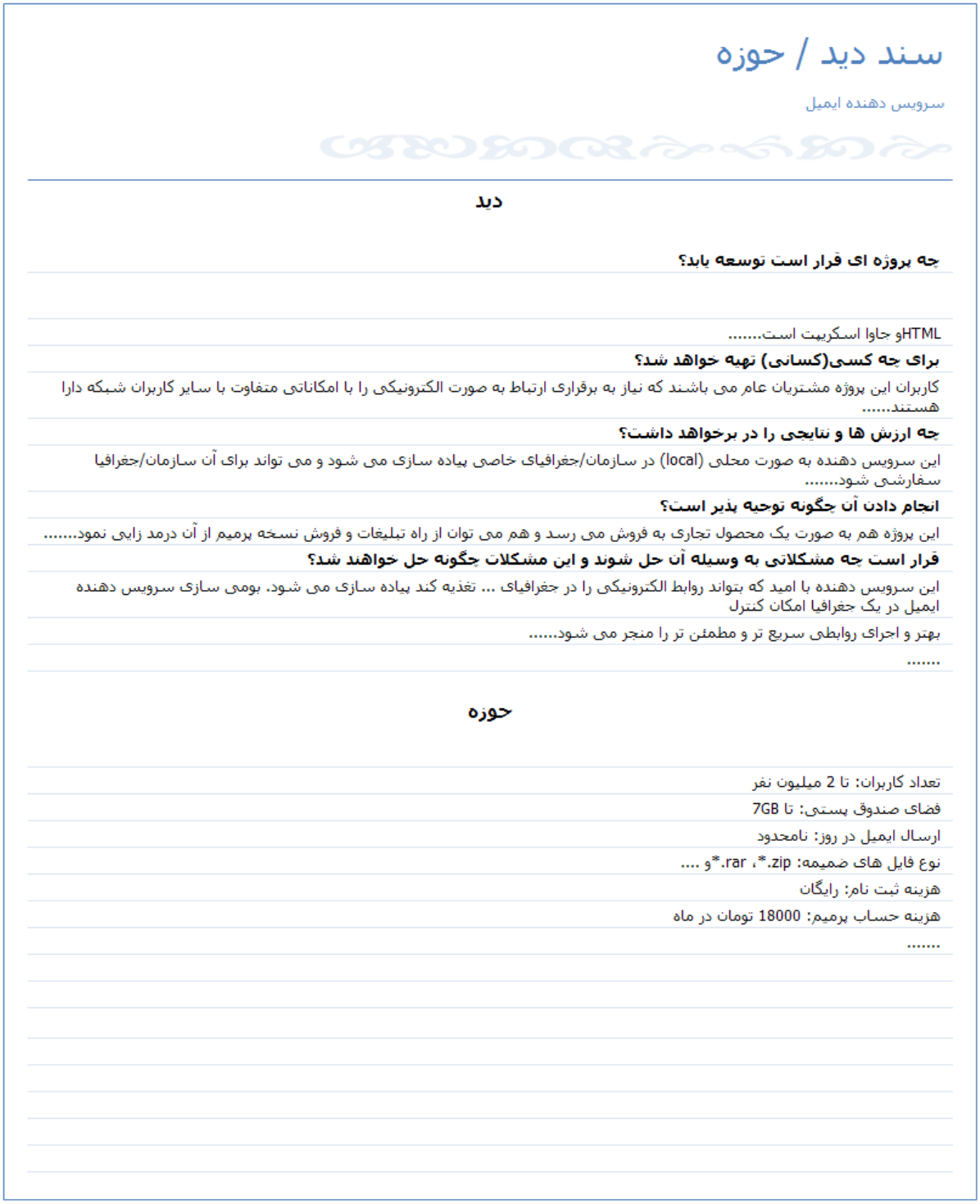
پروژه ای به خوبی توسعه می یابد که به خوبی نیز تعریف شده باشد. برای تعریف یک پروژه لازم است بتوانیم به سوالات زیر پاسخ دهیم:

- چه پروژه ای قرار است توسعه یابد؟
- برای چه کسی(کسانی) تهیه خواهد شد؟
- چه ارزش ها و نتایجی را در برخواهد داشت؟

- انجام دادن آن چگونه توجیه پذیر است؟
- قرار است چه مشکلاتی به وسیله آن حل شوند و این مشکلات چگونه حل خواهند شد؟

این سوالات باعث می شوند دید بهتری از کاری که قرار است انجام دهیم پیدا کنیم. این دید برای جلوگیری از شبهات و کج روی ها بسیار ضروری است و باید بین اعضای تیم توسعه و همچنین مشتری یکسان باشد.

در تعریف پروژه پیدا کرده دید به تنهایی کافی نخواهد بود. بلکه باید ابعاد پروژه نیز معرفی شوند. در تشریح ابعاد یک پروژه به این موضوع می پردازیم که محدوده راه حل هایی که قرار است پروژه، آن ها را پیاده سازی کند چیستند؟ مثلاً اگر قرار است یک سرویس دهنده ایمیل راه اندازی شود باید معین شود که چه تعداد کاربر را پوشش خواهد داد، صندوق ایمیل دارای چه فضایی است، ارسال چه تعداد ایمیل در روز مجاز است و... . به عبارت دیگر حوزه یک پروژه تشریح محدوده دیدی است که برای پروژه تعریف می گردد. در حوزه پروژه جزئیات بیشتری معرفی می گردد که باعث می شود شناخت بهتر و دقیق تری از پروژه داشته باشیم. دقیق تر از این جهت که همان طور که در مثال مشاهده شد در بحث حوزه به تعریف عددی از پروژه می پردازیم. ترکیب دید/حوزه تعریف مناسبی از پروژه به دست می دهد. اغلب برای آن سندی رسمی با عنوان "سند دید/حوزه" تهیه می شود(شکل 12-2).



شکل 12-2: سند دید/حوزه برای پروژه سرویس دهنده ایمیل

برنامه ریزی

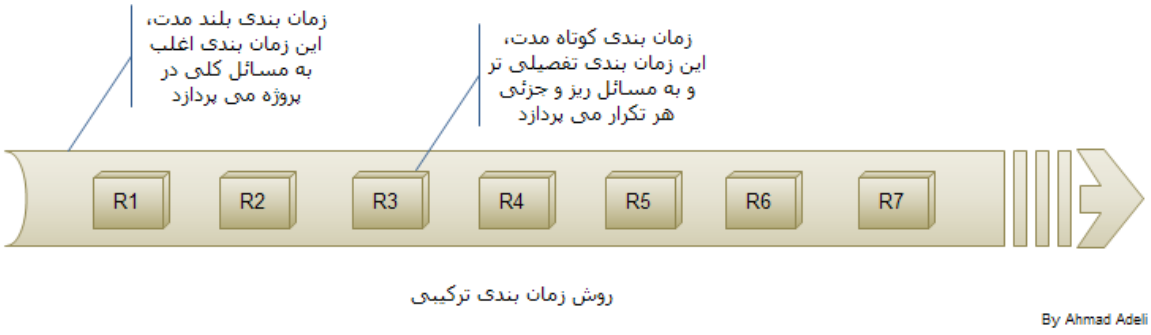
همانند پروژه های دیگر صنایع پروژه های صنعت نرم افزار نیز نیاز به برنامه ریزی دارد. در برنامه ریزی تعیین می شود چه چیزی، چگونه، در چه زمانی و توسط چه کسی باید انجام شود. برنامه ریزی اصولاً ابتدای هر پروژه صورت می پذیرند به شکلی که برای کل پروژه مورد استفاده قرار گیرند. تجربه فرآیندهای پیشین ثابت کرده است که توسعه نرم افزار را نمی توان به صورت بلند مدت برنامه ریزی نمود. توسعه نرم افزار فرآیندی متغیر و بی ثبات است. از این رو برنامه ریزی بلند مدت گزینه خوبی برای آن نخواهد بود. در اسکرام اغلب برنامه ریزی های به صورت کوتاه مدت صورت می پذیرد. این برنامه ریزی ها در ابتدای هر اسپرینت و در برخی مواقع ابتدای

هر Release انجام می شوند مزیت برنامه ریزی کوتاه مدت این است که تغییرات احتمالی بهتر قابل مدیریت می باشند. اما این بدین معنا نیست که نباید ابتدا پروژه برنامه ریزی انجام داد. ابتدای پروژه (قبل شروع سلسله اسپرینت ها) معمولاً برنامه ریزی های استراتژیک و کلی صورت می گیرد. به طوری که این برنامه ریزی به تمام اسپرینت ها قابل اعمال باشد. در برنامه ریزی ابتدای پروژه اغلب آیتم هایی چون زمان بندی پروژه، تعداد Release ها، برآورد هزینه ها، برآورد منابع خارجی، برآورد منابع انسانی، ریسک های احتمالی، استراتژی های رویارویی با ریسک ها و ... پرداخته می شود. این برنامه ریزی به صورت کلی انجام می شود و جزئیات آن در برنامه ریزی های کوتاه مدت و محلی صورت می پذیرد.

زمان بندی

زمان بندی جزئی از برنامه ریزی است. از رو مانند برنامه ریزی، به دو نوع تقسیم می شود: زمان بندی بلند مدت و زمان بندی کوتاه مدت. زمان بندی بلند مدت در ابتدای پروژه صورت می گیرد. هدف این نوع زمان بندی تعیین محدوده زمان کل پروژه می باشد. البته زمان بندی بلند مدت هیچ گاه با دقت خیلی بالا صورت نمی گیرد و تیم های اغلب از روش تخمین برای آن استفاده می کنند. در زمان بندی بلند مدت تیم زمان کل پروژه را تخمین میزند. این زمان ممکن است در طول پروژه تغییر کند در نتیجه تیم توسعه و پروژه باید انعطاف لازم برای آن را داشته باشند. مدتِ زمانیِ Release ها نیز در این زمان بندی تعیین می شود. مدتِ زمانیِ Releaseها پس از تعیین در ابتدای پروژه، اغلب تا انتها ثابت خواهند ماند. علاوه بر این ها در زمان بندی بلند مدت ممکن است زمان آیتم های دیگری چون تغییرات تیم، تحقیقات و ... نیز مشخص شوند.

نوع دیگر زمان بندی، زمان بندی کوتاه مدت است. در اسکرام پروژه در یک زمان واحد و یکپارچه انجام نمی شود بلکه زمان توسعه به بخش های کوچک تری تقسیم می شود. در نتیجه بهتر است زمان بندی هر کدام از این قسمت ها در وقت خود صورت گیرد چرا که باعث حداقل سازی تغییرات در زمان بندی خواهد شد. زمان بندی کوتاه مدت دقیق تر، راحت تر و از دیدگاهی کم هزینه تر از زمان بندی بلند مدت خواهد بود. اما از طرفی برای هر زمان بندی کوتاه مدت باید جلسه ای تشکیل شود که خود تا حدودی باعث هدر رفتن منابع و در نتیجه افزایش هزینه خواهد شد. زمان بندی کوتاه مدت معقول تر و قابل اجرا تر است اما به این معنی نیست که یک پروژه نیازی به زمان بندی کلی و بلند مدت ندارد. از این رو در اسکرام از هر دو زمان بندی به صورت ترکیبی استفاده می شود. توجه شود همان طور که ابتدای این بخش ذکر شد زمان بندی جزئی از برنامه ریزی است ترکیبی بودن اجرای زمان بندی از ترکیبی بودن برنامه ریزی ناشی می شود(شکل 13-2).



شکل 13-2: روش زمان بندی ترکیبی

بودجه بندی

بودجه بندی به زبان ساده عبارت است تخمین هزینه پروژه و برنامه ریزی برای تخصیص آن به قسمت های مختلف توسعه. بودجه بندی از مهم ترین برآوردهای مقدماتی پروژه می باشد بدون بودجه مناسب هیچ پروژه ای به درستی توسعه نخواهد یافت. بودجه کم باعث شکست پروژه خواهد شد و بودجه زیاد نیز اتلاف هزینه را در بر خواهد داشت. در بودجه بندی پروژه های نرم افزاری آیتم هایی چون هزینه نیروی انسانی، هزینه تهیه و تأمین منابع، هزینه های احتمالی تحقیقات جستجو، هزینه بازاریابی (در صورتی که محصول دارای مشتری عام باشد) و ... وجود دارند. یکی از مهمترین هزینه هایی که ریسک پروژه در صنعت نرم افزار را نسبت به دیگر صنایع کاهش می دهد، هزینه مواد اولیه می باشد. در صنایع دیگر این هزینه اصلی ترین بخش بودجه بندی را شامل می شود و اغلب تعیین بودجه آن با ریسک بالایی همراه است. اما در صنعت نرم افزار هزینه ای با این نام وجود

نداردکه باعث می شود نه تنها پروژه با ریسک های کمتری اجرا شود بلکه هزینه نهایی پروژه نیز نسبتاً پایین خواهد بود. بودجه بندی در هر پروژه ای صرف نظر نوع آن عملی دشوار است و باید بسیار با دقت صورت پذیرد.

برآورد منابع

منابع پروژه عبارت است از هر محلی که برای تغذیه پروژه از آن بهره می بریم. این منابع عبارت اند از نیروی انسانی، سخت افزار، نرم افزار، محل کار(سایت)، منابع آموزشی و تحقیقاتی و حتی زمان توسعه. هدف *تعیین منابع*، تقویت توانایی اداره آن ها می باشد. با تعیین دقیق منابع بهتر می توانیم عملیاتی چون برنامه ریزی و برآورد هزینه را انجام دهیم. اولین منبعی که معمولاً در پروژه تعیین می شود نیروی انسانی یا همان تیم توسعه می باشد. تیم توسعه مهمترین منبع برای یک پروژه می باشد و باید با دقت بسیار زیادی گزینش شود. نیروی انسانی برای برخی پروژه ها ممکن است پر هزینه ترین منبع باشد و به دلیل پیش بینی پذیری بسیار پایین آن برنامه ریزی برای آن بسیار دشوار است.

نیروی انسانی از منابعی است که احتمال تغییر آن در پروژه بسیار زیاد است. از این رو باید برنامه ریزی آن تا حد ممکن قابل انعطاف باشد. دیگر منابع اغلب بر اساس نیاز تیم توسعه مورد تعیین می شوند چرا که توسط تیم مورد استفاده قرار می گیرد. بسته به نوع پروژه و البته تیم توسعه انواع مختلفی از منابع سخت افزاری و نرم افزاری وجود دارد. برخی از آن ها رایگان هستند اما تهیه برخی از آن ها ممکن است هزینه بر باشد. منابع نرم افزاری اغلب مجدداً قابل استفاده هستند. همچنین در توسعه نرم افزار کمتر منابع سفارشی سخت افزاری مورد نیاز خواهد بود در حالت ایده آل هر شخص تنها نیاز به یک ماشین(کامپیوتر) خواهد داشت. از این رو مدیریت منابع در پروژه نرم افزاری نسبت به سایر صنایع به میزان قابل توجهی ساده تر است. به دلیل ارتباط نزدیک اعضای تیم، بهتر است همه در یک محل(سایت) در توسعه کار کنند. تجربه های نشان داده شیوه های غیر حضوری چون دورکاری برای پروژه های نرم افزاری چندان ضایت بخش نبوده است. بنابراین یکی از منابعی که حتماً برای آن باید بودجه تخصیص داده شود سایت(محل) توسعه می باشد. این منبع نیز از منابعی می باشد که تغییرات در آن اجتناب ناپذیر است(به خصوص در زمانی که پروژه به طول بیانجامد).

منابع تحقیقاتی و آموزشی همیشه و همه جا در دسترس هستند اما یک برنامه ریزی خوب حکم می کند که به آن ها به عنوان یک منبع قابل مدیریت نگاه شود. در پروژه هایی که با تکنولوژی جدید سر و کار دارند و یا دارای حساست موضوعی می باشند منابع آموزشی و مدیریتی چندان کم هزینه نخواهند بود در نتیجه باید در برنامه ریزی و بودجه بندی اولویت مناسبی به آن ها داده شود.

بیشترین منبعی که توسط پروژه مصرف می شود زمان است. این منبع برخلاف بسیاری از منابع غیر قابل بازگشت است. در جهان اطراف ما همه چیز در حصار زمان است. صرف زمان نیز هزینه طلب می کند. از طرفی زمان با منابع دیگری همچون نیروی انسانی و سایت توسعه ارتباط مستقیم دارد. بر اساس این آرا باید به زمان توسعه به عنوان یک منبع حساس و مصرف شدنی نگاه می شود و برنامه ریزی برای آن تحت عنوانی جداگانه یعنی زمان بندی صورت می گیرد. مدیریت منابع در اسکرام تفاوت چندانی با سایر متدولوژی ها ندارد و معمولاً منابع در سندی قابل تغییر نگهداری می شوند.

تحلیل محصول

خروجی یک پروژه نرم افزاری یا تجاری یا سفارشی است. به هر صورتی که باشد برای یک تحویل خوب باید استراتژی مناسبی را پیش گرفت. اگر نرم افزار تجاری باشد باید بر اساس سیاست جغرافیایی محلی که نرم افزار در آن توزیع می شود عمل شود. منظور از سیاست های جغرافیایی قوانین حاکم بر یک کشور یا ایالت نسبت به توزیع این گونه محصولات نرم افزاری مانند کپی رایت و مالکیت معنوی است.

شیوه ارائه محصول یکی دیگر از چالش های نرم افزارهای تجاری است در که هنگام تحلیل محصول باید مورد توجه قرار گیرد. ارائه محصول تجاری به دو شیوه صورت می پذیرد توزیع آنلاین و توزیع از طریق بازار. هر کدام از این دو مزایا و معایبی دارند که بسته به نوع محصول، هر تیم ممکن است یکی را بر دیگری ترجیح دهد یا حتی از ترکیب این دو استفاده کند. همچنین شیوه های بازاریابی محصول از اجزای این تحلیل است که باید با دقت و یا مشورت متخصصان بازاریابی صورت گیرد. فروشندگان محصولات تجاری به خوبی می دانند که ارائه محصول فقط فروش محصول نیست بلکه پشتیبانی و خدمات پس از فروش به همان میزان اهمیت دارد. در نتیجه خدمات پشتیبانی باید به خوبی بررسی و در طول توسعه طراحی شود تا همزمان با پایان پروژه آماده کار شود.

برای نرم افزارهای سفارشی وضعیت به کلی متفاوت است. نرم افزارهای سفارشی بر خلاف نرم افزارهای تجاری که مشتریان زیادی دارد، فقط دارای یک مشتری است و آن هم سازمانی است که نرم افزار را سفارش

داده است. بنابراین ارائه نرم افزار به معنای تحویل پروژه به سفارش دهنده می باشد. از این رو مسائلی چون بازاریابی و شیوه های ارائه در این نوع نرم افزار مطرح نیستند.

همچنین سیاست های جغرافیایی چندانی روی این گونه نرم افزارها اثر گذار نیستند و به جای آن پروژه های سفارشی به طور مستقیم از سیاست سازمانی تاثیر می پذیرد. سیاست های سازمانی همانطور که از نامشان پیدا است سیاست هایی هستند که یک سازمان(سفارش دهنده پروژه) در خصوص محصول نرم افزاری خود دارد مانند سیاست هایی که برخی سازمان ها برای استفاده از پلت فرم های خاص تعریف می کنند. پشتیبانی محصولات نرم افزاری نیز بسیار ساده تر از محصولات تجاری صورت می پذیرد. به دلیل بودن رابطه یک به یک بین مشتری و تهیه کننده در محصولات سفارشی، تیم های توسعه اغلب طرح خاصی را برای پشتیبانی دنبال نمی کنند و بیشتر راه حا ها را به زمان بروز مشکل موکول می کنند. به طور کلی ارائه توسعه محصولات سفارشی به مراتب آسان تر از محصولات تجاری است و سرمایه اولیه کمتری را نیز طلب می کند(اگر چه به همان نسبت سود حاصل از فروش آنها نیز پایین است).

اساساً در تحلیل بهتر است به نرم افزار به دید یک محصول نگاه شود. بنابراین می توان از تجارب دیگر صنایع در ارائه محصولات شان به عنوان آموزه های تجاری بهره برد.

تحلیل مقدمانی نرم افزار

تحلیل نرم افزار یکی از مهمترین اعضای توسعه می باشد که برای بخش های مختلف به صورت جداگانه انجام می شود. اما پیش از شروع توسعه نرم افزار تحلیلی کلی وجود دارد که به همه بخش ها مربوط می شود و عموماً در ابتدای توسعه صورت می پذیرد. در این تحلیل آیتم هایی چون نحوه پیاده سازی، چگونگی ارتباط کامپوننت ها محیط های توسعه شناخت کاربران و وظایفشان بررسی می شود. پیش از شروع تکرارهای توسعه تیم باید تصمیم بگیرد که نرم افزار را تحت چه تکنولوژی پیاده سازی کند. تکنولوژی توسعه انواع مختلفی هستند و نسبت به هدف پروژه و البته سیاست های سازمان انتخاب می شوند. برخی تکنولوژی ها تحت وب هستند و برخی تحت کلاینت. چنین انتخابی می تواند آسان باشد اما هنگامی که در یک زمینه تکنولوژی یکسانی وجود دارد فرآیند انتخاب ممکن است کمی پیچیده شوند. پیچیدگی هنگامی که اعضای تیم از پیش تعیین شده باشند و در تکنولوژی های مختلفی تسلط داشته باشند، دو چندان خواهد شد.

این اشتباه وجود دارد که در برخی توسعه ها به علت اعضای ثابت تیم توسعه تکنولوژی را بر اساس مهارت افراد انتخاب می کنند. در این صورت باعث از دست رفتن برخی مزایای تکنولوژی های برتر خواهد شد که حتی ممکن است تاثیری منفی در موفقیت پروژه داشته باشد. گرچه در توسعه های امروزی بحث زبان پیاده سازی چندان مطرح نیست اما می توان آن را جزئی از تحلیل پیاده سازی دانست. بعضی از تکنولوژی ها از چند زبان برنامه نویسی پشتبانی می کنند و برخی از یک زبان خاص. این خود می تواند یکی از اولویت های انتخاب تکنولوژی قلمداد شود. اهمیت انتخاب و بررسی زبان به ویژه برای نرم افزار هایی مهم است که نیاز به نگهداری بلند مدت خواهند داشت. چرا که ممکن است سیاست های گزینش نیروی انسانی یک سازمان بر اساس زبانی خاص اتخاذ شده باشند.

مبحث دیگری که در این تحلیل می گنجد بحث معماری نرم افزار است. اگر دامنه کاربرد نرم افزاری چندان نو و جدید نباشد می توان از معماری های معمول و تست شده الگو برداری کرد. مبحث معماری نرم افزار به چگونگی ارتباط بین کامپوننت های یک نرم افزار مرتبط می شود و تا پایان توسعه و با ارائه کامپونت های جدیدتر درست خوش تغییرات است. با این حال پیش از شروع توسعه باید زیر ساخت و پلت فرم مناسب آن تعیین و طراحی شود. معماری نرم افزار به طور مستقیم از تکنولوژی های پیاده سازی نرم افزار متأثر است.

معمول است که همراه هر تکنولوژی، حداقل یک معماری وجود داشته باشد. از این رو معماری نرم افزار چالش چندانی را برای تیم توسعه معمولاً ایجاد نمی کند. در توسعه نرم افزار عرف بر این است که در طول پروژه از یک محیط توسعه(IDE) استفاده می شود. بنابراین یک بار آن هم در ابتدای پروژه این محیط انتخاب می شود. ممکن است برای تکنولوژی و زبان پیاده سازی محیط های مختلفی موجود باشد. خوشبختانه در انتخاب محیط توسعه درست تیم تا حدود زیادی باز است و معمولاً بر اساس تجربه تیم یک محیط توسعه گزینش می شود. البته در این انتخاب فاکتورهای دیگری چون هزینه نیز دخیل هستند که اولویت های جدیدی غیر از تجربه تیم را عنوان می کند. در برخی سازمان ها محیط های توسعه سفارشی شده خاصی وجود دارد که انتخاب تیم را محدودتر می کند.

مشتری های یک نرم افزار ممکن است همیشه کابران نهایی نباشند و حتی ممکن است در یک نرم افزار بازه وسیعی از انواع کاربران تعریف شده باشد. شناسایی این کاربران و وظایف آن ها در ابتدای پروژه امری است ضروری چرا که تأثیر عمیقی در شناسایی، تحلیل و طراحی زیر سیستم ها و کامپوننت های یک نرم افزار خواهد گذاشت. اما این تحلیل، تحلیلی است مقدماتی و معمولاً در طول توسعه و با اشراف مضاعف تیم بر دامنه کاربرد تغییرات زیادی در آن حاصل می شود. تعیین کاربران و اعمال آن ها در سیستم به نحوی باعث بهبود شناسایی منابع می شود. به عنوان مثال اگر عملیاتی مالی برای کاربران تعریف شود تیم پی خواهد برد که نیاز به سامانه ای جهت اجرا تراکنش های مالی وجود دارد؛ بر این اساس اقدام به برنامه ریزی و بودجه بندی برای آن می کند. شناسایی کاربران می تواند مسیر شناسایی نیازمندی های نرم افزار را هموارتر سازد. چرا که باعث شناسایی بخش ها و افراد ذی ربطی می شود که نیازهای حقیقی قرار است از آن ها استخراج شود.

تحلیل مقدماتی نرم افزار از دشوار ترین اعمالی است که هر تیم مجبور است در ابتدای انجام دهد. دلیل این دشواری این است که نرم افزار خود، محصولی است انتزاعی و در ابتدای پروژه اساساً همان محصول انتزاعی نیز وجود ندارد. از این رو تحلیل های مقدماتی عموماً در بالاترین سطح انتزاع در توسعه یک نرم افزار صورت می گیرد. آیتم هایی که در تحلیل مقدماتی بررسی می شوند اغلب بیش این مواردی است که در این بخش ذکر شد، لیکن شرح همه آن ها از حصوله این کتاب خارج است.

اغلب تیم ها تمایل دارند پیش از شروع توسعه از آمادگی لازم برخودار شوند. بدیهی است که این آمادگی باعث بهبود کیفیت عمکرد آنها در توسعه خواهد شد. اما افراط در این امر نیز خود می تواند ضربه ای شدید به زمان بندی پروژه وارد سازد. بنابراین این هنر تیم توسعه است که با ایجاد توازن در انجام این گونه عملیات مقدماتی نه تنها زمان توسعه را حفظ کند بلکه بتواند آمادگی مناسب و کاملی برای ادامه توسعه کسب نماید.

فصل 3: نقش های اسکرام

در یک تیم اسکرام ممکن ات افراد مختلفی به کار بپردازند و هر کدام از آن ها وظایف متفاوتی داشته باشند. برخی از این افراد وظایف مشابهی دارند. به دسته بندی وظایف بر اساس نوع کار آن ها، *نقش* گفته می شود. در اسکرام سه نقش اصلی وجود دارد: مالک محصول، اسکرام مستر و توسعه دهنده. هر کدام از نقش ها ممکن است توسط بیش از یک نفر ایفا شوند و از طرفی هر فرد می تواند بیش از یک نقش را عهده دار شود. هر کدام از این نقش ها وظایف شفاف و تعریف شده دارند که باعث می شود افرادی که این نقش ها را بر عهده دارند با یکدیگر دچار تداخل وظیفه نشوند. هر سه نقش از درجه اهمیت یکسانی برخوردارند به طوری که فقدان هرکدام باعث از کار افتادن روند جاری توسعه خواهد شد. در فرآیند های توسعه سنتی نقش های به مراتب بیشتری وجود داشت که هر کدام به قسمتی از توسعه مربوط می شدند. در مقام مقایسه با این فرآیندها به نظر می رسد که تعدادی از نقش معمول در توسعه نرم افزار در اسکرام وجود ندارند. اما باید توجه داشت که هر کدام از این نقش ها شامل نقش های دیگری نیز می باشند به طوری که هر وظیفه ای که در توسعه معمول وجود دارد در دل یکی از این نقش ها تعبیه شده است. در ادامه این فصل سه نقش یاد شده با جزئیات بیشتری مورد بحث قرار خواهد گرفت.

مالک محصول

مالک محصول ارتباط دهنده تیم توسعه و مشتری می باشد. مالک محصول بر خلاف نام شبهه برانگیز آن پیمان کار یا تهیه کننده پروژه نیست؛ بلکه صرفاً پلی است ارتباطی میان همه اعضای درگیر در پروژه(شکل 10-2 مشاهده شود). اسکرام یک متدولوژی مبتنی بر روابط است و همخوان با ارزش اول از بیانیه آجیل به روابط بین فردی بیش از مستندات بها داده می شود. اما همان طور که گفته شد پیاده سازی چنین روابطی بین اعضای تیم توسعه و مشتریان کار چندان ساده ای نمی باشد.

فرض شود در پروژه ای، تیم توسعه از 8 نفر تشکیل شده باشد. در طرف دیگر نمایندگان مشتری نیز از دو مدیر در دو سطح مختلف و دو اپراتور تشکیل شده باشد. ارتباط بین این افراد نیاز به صرف زمان و انرژی بسیار زیاد و جلسات مکرر خواهد داشت. ارتباط هر کدام از اعضا با یکدیگر در نهایت جلسه تبدیل به یک شبکه ارتباطی پیچیده و کاملاً بی معنی خواهد کرد. در اسکرام برای حل این مشکل راه ساده تری تعریف شده است. همه ارتباطاتی که قرار است ایجاد شوند به طور مرکزی در نقش مالک محصول اداره می شوند. ارتباطی بدین شکل خود باعث می شود امور کاراتر صورت پذیرد چرا که توزیع ارتباط جامع تر خواهد بود و باعث خواهد شد سایر نقش ها به جای درگیر شدن در یک ارتباط پیچیده بر روی وظایف خود تمرکز بیشتری داشته باشند. مالک محصول دارای چهار وظیفه می باشد که شرح آن در زیر آمده است:

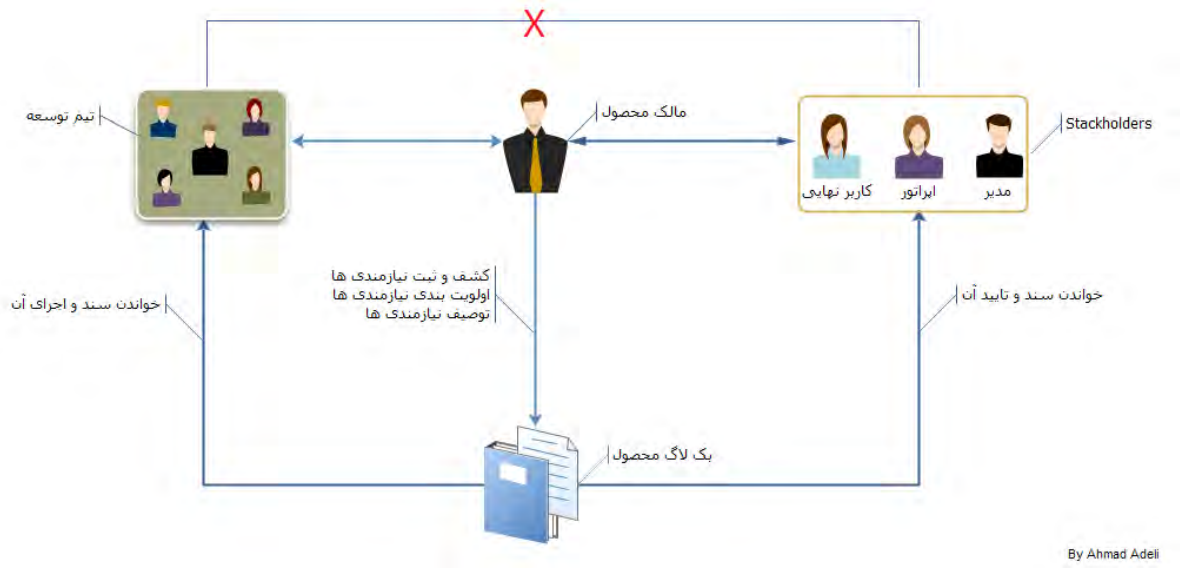
تعیین مشخصات نرم افزار

بارزترین وظیفه مالک محصول را می توان مهندسی نیازها دانست. مالک محصول در طول توسعه همواره به دنبال شناسایی نیازمندی های نرم افزار می باشد. او به طور پیوسته با مدیران، اپراتورها و افراد دیگر مرتبط با پروژه ارتباط برقرار می کند. او همچنین مستندات سازمانی را مطالعه می کند، چارت ها و نمودارها و گزارشات را بررسی می کند تا بتواند نیازمندی های حقیقی نرم افزار را کشف کند. مالک محصول پس از این پی می برد که انتظارات مشتری از نرم افزار چیست؟ سپس این انتظارات را به زبان توسعه که همان نیازمندی های نرم افزار است ترجمه کرده و به تیم توسعه تحویل می دهد. از این رو ناگفته پیدا است که مالک محصول یک متخصص توسعه یا همان مهندس نرم افزار است.

تنها سند تحت تملک مالک محصول بک لاگ محصول می باشد. بک لاگ محصول در حقیقت قلب اسکرام است. هر اقدامی برای توسعه از طرف تیم توسعه باید به تحریک آیتم های موجود در بک لاگ محصول صورت پذیرد. تنها فردی که می تواند در بک لاگ محصول تغییرات ایجاد کند شخص مالک محصول است. مالک محصول پس از شناسایی نیازها، آن ها را به صورت اولویت بندی شده، در بک لاگ محصول وارد می سازد. آیتم های بک لاگ اولویت بندی می شوند و تنها کسی که می تواند اولویت این آیتم ها را تعیین کند نیز خود مالک محصول است.

مالک محصول پروژه را از دید مشتری می بیند با این تفاوت که بر خلاف مشتری با تیم توسعه هم زبان است. پس در نتیجه آن چه او از تیم توسعه خواستار است همان است که مشتری انتظار دارد. او می داند که نرم افزار باید چه قابلیت هایی داشته باشد، چه قابلیت هایی دارای اهمیت بالاتری هستند و چه قابلیت هایی باید سریع تر پیاده سازی شوند. او این دانسته ها را در اولویت بندی شده بک لاگ محصول به تیم توسعه انتقال می دهد. تیم نمی تواند هیچ دخالتی در مشخصات محصولی که باید ساخته شود داشته باشد و صرفاً مسئول پیاده سازی آن چیزی است که مالک محصول وارد کرده است. مالک محصول به مشتری ضمانت می دهد که قابلیت

های خواسته شده به درستی و به موقع در نرم افزار پیاده سازی شوند و نرم افزار نهایی مشخصات تعیین شده را دارا باشد. از این رو مالک محصول همواره در طول توسعه دغدغه کاهش آیتم های بک لاگ محصول و یا به عبارتی افزایش قابلیت های نرم افزار را دارا است و برای اجرای صحیح آن تلاش می کند(شکل 1-3).



شکل 1-3: تعیین مشخصات نرم افزار توسط مالک محصول

کنترل برنامه ریزی Release و اسپرینت

برنامه ریزی برای اسپرینت و Release یکی از اعمالی است که در اسکرام حساسیت ویژه ای دارد. حساسیت آن از این جهت است که در پایان هر اسپرینت و در نهایت هر Release باید قابلیتی به نرم افزار اضافه شود. این کار به صورت گروهی توسط تیم توسعه اسکرام، اسکرام مستر و مالک محصول صورت می پذیرد. در این بین تنها شخصی که به حساسیت های محصول اشراف کامل دارد مالک محصول است. آنچه در پایان Release به مشتری تحویل داده می شود، اغلب با پیشنهاد مالک محصول تعیین می شود.

مشتری در بازه های زمانی انتظاراتی در خصوص نرم افزار در دست تکمیل دارد. این بازه های زمانی مورد انتظار اغلب با بازه های زمانی Release ها هماهنگ است. از آن جایی که مالک محصول ضمانت کننده یک نرم افزار کارکننده در بازه های مشخص می باشد؛ همواره سعی می کند برنامه ریزی Release را به گونه ای کنترل کند که موجبات ارضای انتظارات مشتری فرآهم آید. اما او تنها شخصی نیست که برای این برنامه ریزی تصمیم می گیرد. مالک محصول باید با تیم توسعه که وظیفه اصلی توسعه را برعهده دارد به توافق برسد. چرا که ممکن است انتظاراتی که، آن ها هم از انتظارات مشتری تغذیه می شوند از نظر عملی ناممکن باشند که این خود موجب می شود تیم توسعه هم که به نوبه خود وظیفه ارائه کارهای اخذ شده را در پایان Release بر عهده دارد، از ارضای برخی نیازها سر باز بزند.

تا این جا ممکن است این شبهه به وجود آمده باشد که مالک محصول باید از گروه مشتری باشد چرا که در پی ارضای احتیاجات مشتری می باشد. مالک محصول عضوی از تیم اسکرام است اگر مالک محصول وجود نداشته باشد اعضای تیم توسعه خود باید به جستجوی نیازهای مشتری بپردازند که در نهایت همان قابلیت ها را پیاده سازی خواهند کرد که مالک محصول در پی آن بود اما با این تفاوت که این کار زمان و انرژی زیادی را به هدر خواهد داد و کاملاً ناکارآمد می باشد. همان طور که قبلاً ذکر شد یکی از ویژگی های نرم افزار کارکننده *مفید بودن* آن می باشد. مفید بودن به این معنا است که نرم افزار احتیاجات واقعی مشتری را بر طرف سازد و نه صرفاً نیازهایی که او بیان می کند. کشف نیازهای واقعی مشتری یکی از اعمالی است که در صورت عدم وجود مالک محصول تقریباً غیر ممکن خواهد بود. به عبارت دیگر کار مالک محصول ترجمه ای از آن چه مشتری می اندشید و صرفاً نه آنچه او بیان می کند.

آماده سازی تست کیس ها

مالک محصول صرفاً قابلیت های باید توسعه پیدا کنند را تعیین نمی کند بلکه چگونگی این قابلیت ها را نیز تبیین می کند. به عنوان مثال اگر قابلیت جستجو داده های نرم افزار جزو قابلیت های مورد درخواست مالک محصول باشد، او تعیین می کند که جستجو چگونه، بر اساس چه آیتم هایی و با چه فرمتی باید توسط کاربر نهایی انجام شود. این باعث می شود که تیم توسعه دقیقاً آنچه را پیاده سازی کند که مورد نظر مشتری می باشد. گاهی پیش خواهد آمد که آیتمی را که تیم پس از صرف وقت بسیار پیاده سازی کرده است را مشتری نمی پذیرد. این حالت *ابهام در پیاده سازی* نام دارد. برای جلوگیری از این حالت باید پیش از پیاده سازی، مالک محصول شرایط تطبیق با انتظارات مشتری را به صورت مشروح مکتوب کند. تا هم سندی توافقی باشد بین تیم و مشتری و هم راهنمایی برای چگونه ساختن یک آیتم.

همچنین تیم بر اساس این سند می تواند آیتم های پیاده سازی شده را تست نماید. شرایط توافقی را می توان به صورت متنی ساده همراه با آیتم نگه داری کرد و یا اینکه می توان آن را به صورت مدل پیشرفته تری مانند تست کیس استفاده نمود. تست کیس ها اساساً از همان متن شرایط تطبیق تشکیل می شوند با این تفاوت که مراحل آن ها به صورت شفاف تری بیان می شود. پس از تکمیل یک آیتم تست کیسی که برای آن نوشته شده بر روی آن اجرا می شود تا قابلیت تکمیل شده آزموده شود و میزان تطبیق آن با انتظارات مشتری نمایان شود. مالک محصول به عنوان فردی که تسلط کامل بر نیازمندی های نرم افزار را دارد وظیفه نوشتن تست کیس ها را بر عهده دارد. تست کیس های یک آیتم عموماً پیش از توسعه آن آیتم نوشته و آماده می شوند.

برقراری ارتباط دو طرفه

وظیفه مالک محصول تنها انتقال درخواست و احتیاجات مشتری به تیم توسعه نیست. بلکه او وظیفه دارد بازخوردهایی از تیم توسعه داشته باشد که از نظر تیم توسعه به اطلاع مشتری رساند. به عنوان مثال مشتری ممکن است در خصوص یکی از قابلیت های نرم افزار داشته باشد که از نظر تیم توسعه به دلیل محدودیت های فنی غیر قابل پیاده سازی باشد. مالک محصول چنین شرایطی را به اطلاع مشتری خواهند رساند و از آن ها در خواست می کند که در خواسته خود تجدید نظر نمایند. حالت عکس آن هم ممکن است به این صورت که اگر مشتری در خواست یک قابلیت قدیمی را داشته باشد مالک محصول پیشنهاد استفاده از یک تکنولوژی برتر از تیم توسعه به مشتری را منتقل کند. پس در نتیجه شیوه ارتباط مالک محصول می توان شیوه ای دو طرفه باشد.

آنچه تا کنون در مورد مالک محصول گفته شد رویاروی او با مشتری سازمانی بوده است؛ حال باید دید که مالک محصول در خصوص مشتری عام چگونه عمل می کند. هنگامی که مشتری عام است ارتباط چندانی بین تیم توسعه و مشتری وجود ندارد. در این حالت نیز همچنان حضور مالک محصول پر رنگ خواهد بود عمدتاً این بار نقش سازمانده کارها را برعهده خواهد داشت. تشخیص نیازمندی های در توسعه محصولی که مشتری آن عام است در تیم های مختلف متفاوت است. به عنوان مثال گاهی شخص ایده پردازی وجود دارد که چنین نیازمندی هایی را کشف و به تیم توسعه انتقال می دهد که در این صورت می تواند خود نقش مالک محصول را ایفا کند و یا اینکه هیئت ایده پردازی وجود داشته باشد که مالک محصول می تواند نقش رابط این هیئت را با تیم توسعه بر عهده داشته باشد. در این گونه محصولات آیتم هایی چون برنامه ریزی Release به تصمیمات مدیریت سطح بالاتر بستگی خواهد داشت که در این خصوص مالک محصول این گونه تصمیمات را در برنامه ریزی اعمال خواهد کرد. کاهی در برخی شرکت ها که در آن ها پروژه های متعددی انجام می شود مالک محصول حکم مجری طرح را بر عهده دارد و مسئول اصلی اجرای طرح می شود. در چنین حالتی مالک محصول بر اساس سیاست های کلان شرکت و شاید ایده های اعضای تیم بک لاگ محصول را به عنوان سند نیازمندی های نرم افزار تغذیه می کند. به طور کلی می توان گفت پروژه به هر صورتی که باشد مالک محصول مسئول کشف، بررسی و ساماندهی نیازمندی های نرم افزار در حال توسعه خواهد بود.

ویژگی های نقش مالک محصول

برای شفاف سازی بیشتر یک نقش علاوه بر وظایف ما نیاز به بررسی ویژگی های آن نقش نیز داریم. پنج ویژگی بارز مالک محصول در ادامه شرح داده خواهد شد(شکل 2-3).

شکل 2-3: ویژگی های نقش مالک محصول

تخصص در توسعه و اسکرام

مالک محصول با وجود اینکه توسعه دهنده نمی باشد اما در علم مهندسی و توسعه نرم افزار تخصص دارد. در مواردی که مالک محصول خود تجربه توسعه را به عنوان یک توسعه دهنده دارد ارتباط بین او و توسعه دهندگان به دلیل درک فرآیند توسعه، ارتباطی به مراتب منطقی تر خواهد بود. از طرفی ترجمه خواسته های مشتریان به نیازمندی های قابل فهم توسط توسط تیم توسعه نیاز به آشنایی به علم مهندسی نیازها را خواهد داشت. بدیهی است که به مانند همه اعضای اسکرام باید تسلط کافی روی مفاهیم اسکرام و مهارت اجرای آن را نیز داشته باشد.

تسلط بر دامنه کاربر

مالک محصول همان طور که بر توسعه تخصص دارد باید بر دامنه کاربرد نرم افزار در حال توسعه نیز مسلط باشد. برای درک بهتر آنچه مشتری خواستار است و ترجمه صحیح آن به نیازمندی های نرم افزار، مالک محصول باید روی موضوع و دامنه کاربرد اشراف کامل داشته باشد. البته انتظار نمی رود که مالک محصول در همه زمینه ها سر رشته داشته باشد اما باید پیش از شروع توسعه به دانش لازم جهت کار با مشتری دست یابد. معمولاً این گونه است که افرادی که نقش مالک محصول را بر عهده می گیرند در زمینه های محدودی کار می کنند و پروژه هایی را بر اساس زمینه کاربرد آنها انتخاب می کنند. به عنوان مثال ممکن است فردی در زمینه های مالی و حسابداری دانش و تجربه کافی را داشته باشد طبیعتاً چنین فردی تمایل کمتری برای حضور در پروژه های سیستم های اطلاعاتی به عنوان مالک محصول را نشان می دهد. بدیهی است که در صورتی که بخواهد چنین نقشی را ایفا کند باید پیش از آن تسلط کافی در این زمینه را بدست آورد. توجه داشته باشید که آشنایی مختصر به هیچ وجه چاره ساز نخواهد بود. مالک محصول در اسکرام به عنوان شخصی متخصص در دامنه کاربرد پروژه شناخته می شود که قرار است مسائل تکنیکی آن دامنه کاربرد را حل نماید از این رو ضروری است که از شناختی عمیق در آن زمینه برخودار باشد.

قدرت مهارت اجتماعی بالا

کار کردن به صورت گروهی برای هر فردی مهارت های اجتماعی زیادی را طلب می کند. در این بین مالک محصول به سبب انواع ارتباط مختلف و حساسیت آن ها باید مهارت های خاص تری را دارا باشد. مهارت ارتباطی کلامی را در این بین شاید بتوان مهمترین آنها برشمرد. همچنین مالک محصول باید دارای مهارت نگارش گزارشات و مستندات باشد. تسلط داشتن به زبان های طبیعی دیگر به خصوص زبان های رایج نیز می تواند یک امتیاز مثبت برای مالک محصول تلقی شود. در دنیای امروز علاوه بر روابط دنیای واقعی، ارتباطات در دنیای مجازی(اینترنت) نیز به امری معمول بدل شده است. بدیهی است که مهارت های اجتماعی در دنیای مجازی مانند امیل،IM، شبکه های اجتماعی و ... برای نقشی چون مالک محصول امری است ضروری. علاوه بر این موارد مالک محصول به عنوان شخصی که با افرادی در زمینه های دیگر ارتباط برقرار می کند لازم است که از عرف حاکم در این صنایع شناخت داشته باشد تا بتواند به خوبی ازتباطات خویش را سامان دهد.

دارای روحیه همکاری بالا

به احتمال فراوان در فرآیند توسعه موانع زیادی ایجاد می شود که همکاری مالک محصول باعث تسهیل در رفع این گره ها خواهد شد. مالک محصول صرف این که شرکت در جلساتِ رفع موانع وظیفه او نیست ، نباید کمک خود را به تیم دریغ نماید. او به عنوان یک شخص متخصص هم در امور توسعه و هم در زمینه دامنه کاربرد، کلید حل بسیاری از مشکلات خواهد بود. همکاری پیوسته او با تیم توسعه باعث می شود که اعضای تیم به منبع اطلاعاتی خوبی همیشه دسترسی داشته باشند. از طرفی این همکاری برای خود مالک محصول باعث به وجود آمدن بازخوردی مناسب از توسعه خواهد شد.

بصیرت

بصیرت به معنای *درک عمیق* از یک مسئله می باشد و در مقابل دید سطحی مطرح می شود. بصیرت باعث می شود به مسائل به شیوه ای عمیق تر نگاه شود. وجود این ویژگی در فردی که نقش مالک محصول را در توسعه ایفا می کند می تواند شناختی عمیق نسبت به خواسته های مشتری ایجاد کند و از سطحی نگری در آن جلوگیری به عمل آورد. این شناخت عمیق باعث می شود که قابلیت های اضافه شده دقیقاً آنچه باشند که مشتری انتظارش را داشته (اگر چه ممکن است آن ها را صراحتاً طلب نکرده باشد) و برای او سودمند باشند. این خود باعث می شود که به علت ارضای نیازهای مشتری دوباره کاری کاهش یابد و متعاقب آن توسعه تسریع پیدا کند. البته این ویژگی به عنوان یک امتیاز مثبت نگاه می شود و الزامی نیست چرا که در همه افراد این ویژگی وجود ندارد.

مالک محصول اگرچه جزئی از تیم اسکرام می باشد اما عضو تیم توسعه نمی باشد. به این معنی که اعمالی مانند برنامه نویسی و تحلیل را انجام نمی دهد. حتی با وجود این که مالک محصول است که تست کیس ها را می نویسد اما خود، تست کیس ها را اجرا نمی کند. همان طور که گفته شد با وجود وظایف شفاف و تعریف شده مالک محصول او همکاری بسیار نزدیکی با تیم توسعه دارد. یکی از ویژگی های بارز محصول *دسترسی پذیری* او می باشد. او همواره باید برای حل مشکلات دامنه کاربرد حضور داشته باشد و یا به نحوی قابل دسترسی باشد.

در اغلب پروژه ها نقش مالک محصول را یک نفر ایفا می کند. اما در پروژه هایی که دامنه کاربرد سطوح مختلف و پیچیده ای دارند، نقش مالک محصول را چند نفر انجام می دهند. در حالت که چند مالک محصول وجود داشته باشد، پروژه بخش بندی می شود و هر بخش بر عهده یک مالک محصول گذاشته می شود. برای تمرکز این افراد و به خصوص وظایفشان، شخصی به عنوان مالک محصول اصلی ایفای نقش می کند. کل این افراد به عنوان تیم مالک محصول وظایف نقش مالک محصول را در اسکرام بر عهده می گیرند.

اسکرام مستر

این نقش که معمولاً توسط یک نفر ایفا می شود هدایت کننده اسکرام در یک پروژه می باشد. ممکن است به علت نام این نقش این شبهه به وجود می آیند که اسکرام مستر، مدیر اسکرام یا مدیر پروژه می باشد. اسکرام مستر رهبر اسکرام است و همواره سعی او بر این است که اسکرام توسط تیم به درستی اجرا شود. او همچنین تلاش می کند تا از حاد شدن چالش ها جلوگیری به عمل آورد و در صورت بروز مشکل تیم را در جهت حل آن حمایت کند. اسکرام مستر دو وظیفه کلی بر عهده دارد: حفظ بهره وری بالا در توسعه و هدایت توسعه. هر کدام از این وظایف کلی به وظایف جزئی تر شکسته می شوند. این وظایف جزئی در ادامه به تفصیل شرح داده خواهد شد(شکل 3-3).

شکل 3-3: وظایف اسکرام مستر

هدایت در توسعه

انجام جلسات

در اسکرام اکثر ارتباطات گروهی که اغلب منجر به تصمیم گیری می شود در جلسات صورت می گیرد. روند اجرای جلسات اسکرام کاملاً تعریف شده و قانونمند است. در گروه(سازمان)های سلسله مراتبی(سنتی) قوانین اداری و مدیران، انجام جلسات را کنترل و مدیریت می کنند. اما اسکرام دارای مدیریت سطح به سطح نمی باشد و کل اعضای تیم وظیفه کنترل جلسات را بر عهده خواهند داشت. علی رغم مزیت های بسیاری که جلسات اسکرام برای توسعه خواهد داشت گاهی پیش می آید که تیم از انجام جلسات سر باز می زند و یا اعضا برای عدم حضور در جلسات متوسل به بهانه هایی می شوند. این اهمال کاری می تواند ضربه های جبران ناپذیری به توسعه وارد سازد. به یاد داشته باشید ما همواره با شمارش معکوس زمان در توسعه نرم افزار روبرو هستیم و اشتباهات کوچک ممکن است برای توسعه و تیم توسعه بسیار گران تمام شود.

همان طور که گفته شد اعضای تیم اسکرام مسئول کنترل جلسات می باشند اما معمولاً این شخص اسکرام مستر است که با استفاده از اختیارات خود، تیم را مجبور به برگزاری جلسات می نماید. او باید تضمین کند که جلسات به طور منظم و صحیح انجام شوند. زمان بندی، برنامه ریزی و اجرای جلسات از دیگر وظایف اسکرام مستر می باشند.

تضمین نرم افزار کارکننده

در توسعه نرم افزار گاهی پیش می آید که تیم آیتم هایی که برای یک اسپرینت اخذ کرده را به طور کامل تکمیل نمی کند. دلیل آن هم می تواند برآورد اشتباه تیم توسعه از آیتم ها یا رخ دادن اتفاقی پیش بینی نشده در طول توسعه اسپرینت باشد (مثلاً بیماری برخی از اعضای تیم). به هر حال آنچه مسلم است این است که در توسعه روند پیش بینی شده ای وجود ندارد. بدیهی است در چنین شرایطی همه آیتم های بک لاگ اسپرینت تکمیل نمی شوند. تیم ممکن است برای آیتم ها در حالت عادی روندی موازی را پیش گیرد مثلاً اگر دو عمل برنامه نویسی و تست برای سه آیتم وجود داشته باشد ابتدا سه آیتم برنامه نویسی شده و هر سه با هم تست می شود. اگر در اسپرینتی حالتی مانند حالت فوق وجود روی ندهد هیچ قابلیت تعیین شده ای در اسپرینت ظاهر نمی شود. این بدین معنی است که اصل ارائه نرم افزار کارکننده در بیانیه آجیل نقض می شود. در چنین حالاتی اسکرام مستر وظیفه دارد با تغییر برنامه در خروجی، قابلیت های تکمیل شده را ظاهر نماید. مثلاً در مثال پیشین به جای ارائه سه قابلیت ناقص تنها یک قابلیت تحویل داده می شود اما با این تفاوت که کامل و تست

شده است. با بالانس کردن آیتم ها با شرایط موجود اسکرام مستر می تواند ارائه نرم افزار کارکننده را در زمان های تعیین شده تضمین نماید.

تسهیل ارتباطات

کارهای گروهی با وجود مزیت های بسیار زیاد خود حاشیه هایی را نیز در بر دارند. این حواشی طبیعتاً در گروه های خودسازمانده نیز وجود دارد. هرچقدر اعضای یک گروه با یکدیگر متحد و هماهنگ باشند هم ایجاد اصطکاک بین آن ها اجتناب ناپذیر است. در گروه(سازمان)های سلسه مراتبی معمولاً مدیریت حل این اختلافات را برعهده دارد. در گروه های اسکرام که خودسازمانده می باشند این وظیفه اسکرام مستر است که روابط اجتماعی بین افراد را بهبود بخشد. او نه تنها باید مانند یک رهبر مشکلات پیش آمده را حل و فصل کند بلکه باید با مساعد کردن جو اجتماعی محیط کار از به وجود آمدن هر گونه اختلاف و ناهماهنگی پیشگیری نماید. مهم است که کسی که این نقش را برعهده می گیرد از لحاظ مهارت های اجتماعی، فردی با وابط عمومی بالا و انعطاف پذیر باشد چرا که بنابر تعریف، رهبر شخصی است که پیشِ روی عده ای حرکت می کند و با قدم برداشتن خود، راه را به دیگران نشان دهد. دلایل زیادی را می توان برای اختلافات درون گروهی بر شمرد که از جمله آن می توان به حس رقابت، حس برتری جویی، فشار نامتوازن کار و ... اشاره کرد.

اگر جوی بسته بر تیم حاکم باشد ممکن است اعضا کمتر با یکدیگر ارتباط بر قرار کنند. این حالت اغلب در تیم هایی که اعضای آن تجربه کار با یکدیگر را ندارند پیش می آید. این حالت دقیقاً نقض ارزش اول از چهار ارزش بیانیه آجیل است و بدیهی است که باید تغییر کند. بدیهی است که اسکرام مستر رواشناس اجتماعی نیست اما باید بتواند به طریقی اعضا را بیش از پیش با یکدیگر درگیر کند(مثلاً می تواند شیوه برنامه نویسی جفتی استفاده کند) که موجب افزایش و بهبود روابط آن ها با یکدیگر خواهد شد. این مشکل اغلب به مرور زمان خود به خود حل خواهد شد و معمولاً اسکرام مستر مواقعی وارد عمل می شود که شرایط طبیعی نباشد و به دخالت صریح شخصی ثالث نیاز باشد.

اسکرام مستر باید کارایی وسایل ارتباط در تیم را بررسی کند و در صورت نامناسب بودن، آن ها با وسایلی کاراتر جایگزین کند. وسایل ارتباطی وسایلی هستند که در صورت ارتباط اعضا را تسهیل می بخشند مانند ایمیل، تلفن و ... . عدم حضور همه اعضای پروژه در برهه هایی از زمان توسعه امری است اجتناب ناپذیر. این حالت به خصوص در ارتباط اعضای تیم توسعه با مالک محصول عمومیت دارد. جستجو برای راه های ارتباطی سریع و قابل دسترس و آموزش اعضا در صورت نیاز از دیگر وظایف اسکرام مستر در این زمینه خواهد بود.

یکی از مزیت های تیم هایی که در آنها روابط اجتماعی اعضا بالا است اشتراک سریع اطلاعات می باشد. اشتراک اطلاعات یکی از بایدهایی است که تیم های توسعه باید توجه ویژه ای به آن داشته باشند. عدم توزیع مناسب دانش فنی در بین اعضای تیم و عدم توازن سطوح اطلاعاتی آنان می تواند به فرآیند توسه ضربه وارد سازد. همه افراد دارای همه نوع تخصصی نمی باشند بنابراین در یک تیم باید اطلاعات در طول توسعه بین اعضا همگام شود. اسکرام مستر علاوه بر فرآهم سازی زمینه مناسب برای این همگام سازی چه از لحاظ تکنولوژی و چه از لحاظ اجتماعی باید میزان این اطلاعات را نیز در بین اعضا کنترل کند.

همانطور که مشاهده شد کار اسکرام مستر در این بخش بسیار سنگین است چرا که ارتباطات انسانی به علت ریشه در کنش های درونی انسان می تواند بسیار پیچیده باشد که خود موجب می شود تسهیل ارتباطات به امری بسیار حساس بدل شود.

هدایت جلسه برنامه ریزی اسپرینت

به جرأت می توان گفت جلسه برنامه ریزی اسپرینت یکی از دشوارترین و حساس ترین عملیاتی است که گروه در طی توسعه انجام می دهد. خطا در این جلسه ممکن است به قیمت از دست دادن کل زمان پروژه تمام شود. هر کدام از عملیات تعیین سرعت(حجم) اسپرینت، برآورد آیتم ها و بستن بک لاگ اسپرینت نیاز به مهارت و تجربه خواهد داشت. برخی مواقع به عللی چون عدم تجربه کافی تیم در اسکرام و یا فشار کاری، تیم عملکرد مناسبی در عملیات فوق الذکر نخواهد داشت. اسکرام مستر به عنوان شخصی که تسلط کافی در اسکرام دارد وارد عمل می شود و اشتباهات احتمالی تیم را در تصمیم گیری ها متذکر می شود. علاوه بر تذکرات او باید راه حل هایی برای مشکلات احتمالی بیابد و به تیم ارائه کند. به عنوان مثال اگر تیم در تعیین سرعت اسپرینت دچار مشکل شود، او می تواند با استفاده از تکنیک هایی (که در آینده شرح داده خواهد شد) سرعت واقعی تیم را برآورد کند و از این طریق تیم را راهنمایی کند و یا این که ممکن است تیم نتواند برآورد خوبی از برخی آیتم ها انجام دهد، در این اسکرام مستر باید می تواند توجه تیم را به مسائلی معطوف سازد که باعث هدایت آنها در برآورد آیتم مذکور می می شود.

گاهی اوقات جلسات اسکرام به یک میدان جنگ بین مالک محصول و تیم توسعه بدل می شود. اساساً مالک محصول راغب بسیار است تا آیتم های بیشتری در یک اسپریت تکمیل شوند و در نهایت خروجی Release نرم افزار با قابلیت های بیشتری همراه باشد. از آن طرف اعضای تیم توسعه همواره در پی کم کردن تعداد آیتم ها در یک اسپرینت می باشد چرا که ازدیاد آیتم های توسعه در یک اسپرینت کار زیاد و مشقت زیادتری را برای آن ها به همراه خواهد داشت. برخی اوقات این کشمکش ها باعث می شوند که جلسه روند عادی خود را طی نکند و زمان تعیین شده جلسه به هدر رود.

وظیفه اسکرام مستر این است که ابتدا جلسه را کنترل نماید و سپس با استفاده از روش هایی، طرفین را به توافق نزدیک کند. یکی از این روش ها شفاف سازی نام دارد. اختلاف نظری که بین تیم و مالک محصول به وجود آمده احتمالاً به علت اشتباه یکی از طرفین در موضع خود می باشد. اسکرام مستر در این بین هر دو طرف را به شفاف سازی موضع خود تشویق می کند. به این معنی که از تیم می خواهد با توضیح چگونگی آیتم ها و شکستن آن ها به اجزای عملیاتی دلایل خود را مبنی بر عدم پذیرش آیتم ها شرح دهد. همچنین مالک محصول نیز باید تشریح برنامه ریزی Release ها باید ثابت کند که اسپرینت های باقی مانده ظرفیت کافی را برای انجام آیتم های جاری رد شده توسط ندارند. با وجود یک شفاف سازی مناسب انتظار می رود توافقی بین دو طرف حاصل شود.

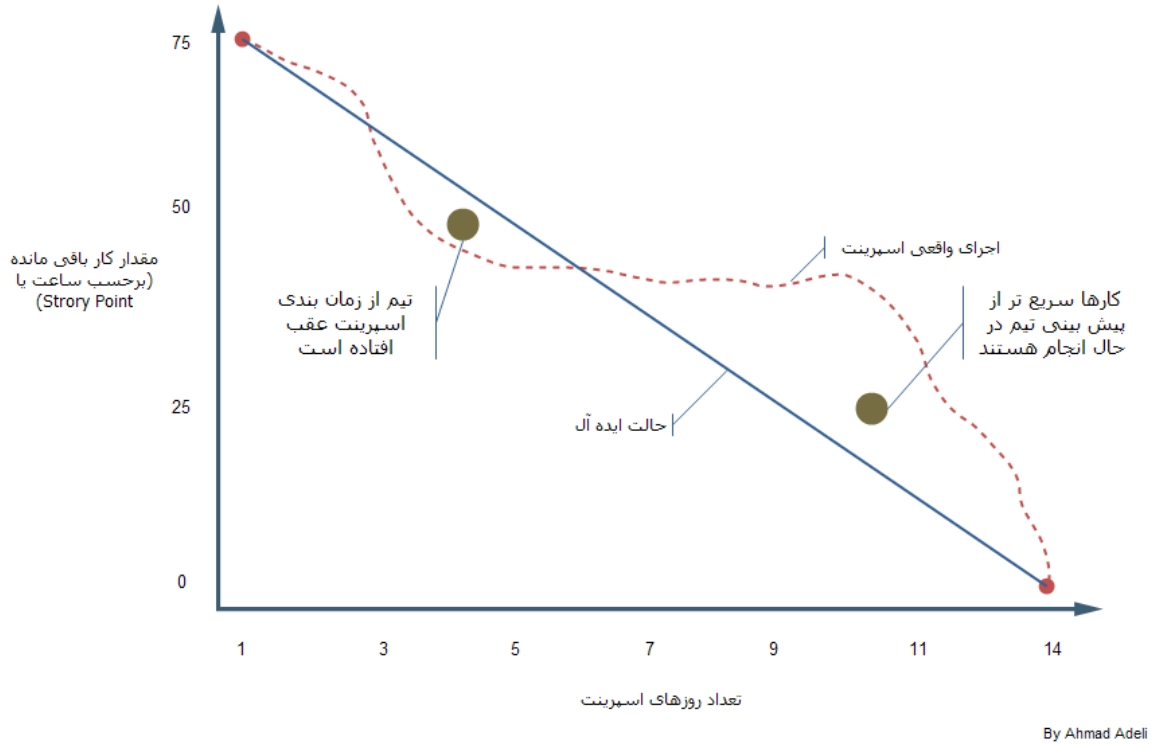
باید توجه داشت این تکنیک ممکن است زمان بر باشد و باید از آن فقط در موارد اختلاف استفاده نمود در غیر این صورت بیش از این که سودمند باشد، مضر و وقت گیر خواهد بود. به طور کلی اسکرام مستر باید جلسات را به گونه ای کارا اداره کند. در زمان های تعیین شده برای جسله، تیم با اولویت های مالک محصول باید حداکثر آیتمی را که می تواند در یک اسپرینت انجام دهد انتخاب کند و بک لاگ اسپرینت را ببند. انجام دادن چنین امری به دانش و مهارت کافی در متدولوژی اسکرام نیاز دارد.

ردگیری اسپرینت و Release

همان طور که در فصل اول گفته شد بخشی از مستندسازی جهت ردگیری و نظارت بر روند پروژه استفاده می شود. در اسکرام نموداری وجود دارد به نام Sprint Burn Down که تیم از طرق آن چگونگی اجرای اسکرام را دنبال می کند. Sprint Burn Down نموداری است ساده که از دو محور زمان و ساعت کار روی آیتم ها تشکیل می شود. زمان اسپرینت هر چه باشد به صورت تعداد روز روی محور Xها قرار می گیرد. محور Yها نیز از جمع ساعات همه کارهای درون اسپرینت تعیین می شود.

هر روز(معمولاً در پایان روز کاری) این نمودار به وسیله نقاطی که روی آن رسم می شود به روز خواهد شد. برای این نقاط قاعدتاً دو مختصات نیاز است روزی که سپری شد (محور)X و میزات ساعات باقی مانده(محور Y). برای محاسبه تعداد ساعات باقی مانده باید ساعات آیتم های انجام شده را با هم جمع نماییم و از تعداد کل ساعات کم کنیم. به این ترتیب میزان ساعات باقی مانده در اسپرینت بدست خواهد آمد. اگر برای همه روزهای اسپرینت این عمل تکرار شود مجموعه ای از نقاط بدست می آید. با اتصال این نقاط به وسیله خط روند اجرایی اسکرام بر اساس یک نموار خطی بدست می آید. خطی فرضی نیز وجود دارد که حالت ایده آل اجرای اسپرینت را نشان می دهد به طوری که میزان کم شدن مجموعه ساعات با روزهای سپری شده اسپرینت کامل هماهنگ می باشند. همانطور که گفته شد این خط فرضی است و حالت ایده آل را نشان می دهد و در عمل هیچ وقت اتفاق نمی افتد بلکه معیاری برای اجرای واقعی اسپرینت است.

اگر خط واقعی بالای خط فرضی باشد به این معنی است که کارها سریع تر از پیش بینی تیم در حال انجام هستند و یا اینکه پیش بینی تیم در خصوص تعداد آیتم های انتخابی نادرست است. همچنین اگر خط واقعی پایین تر از خط ایده آل باشد متوجه خواهیم شد که از زمان بندی اسپرینت عقب افتاده ایم و یا اینکه پیش بینی تیم نسبت به کارهای اخذ شده غیر واقعی بوده است(شکل 4-3).

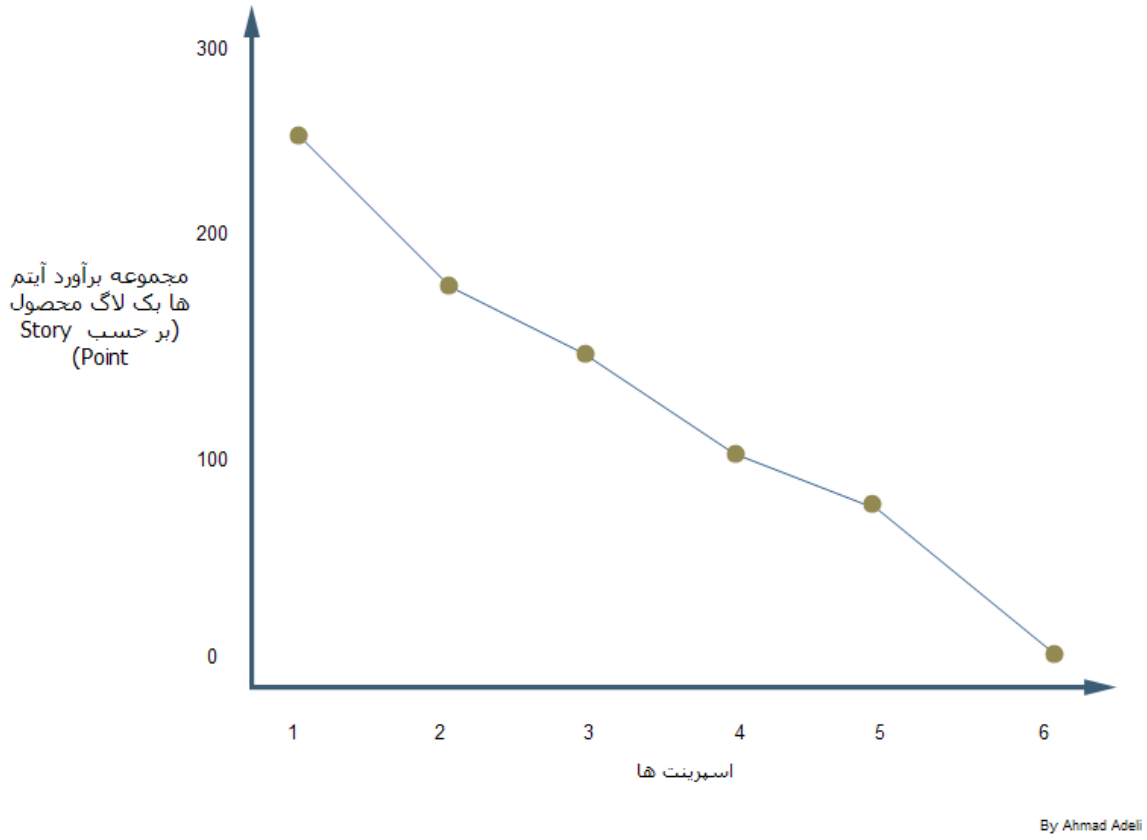


شکل 3-4: نمودار Sprint BurnDown

تیم وظیفه دارد روزانه این نمودار را به روز رسانی کند. چرا که در صورت به روز رسانی مداوم است که این نمودار اثر بخش خواهد بود. بیشترین استفاده از نمودار مربوط به اسکرام مستر است. او وظیفه دارد روند سابقه اجرایی اسپرینت را بررسی و ردگیری نماید و در صورت بروز مشکل آن را تشخیص داده و سریعاً برای رفع آن اقدام کند. نمودار Sprint Burn Down تنها برای اسپرینت جاری قابل استفاده نیست؛ در صورت نگه داشتن سابقه ای از Sprint Burn Down اسپرینت های ما قبل، اسکرام مستر می تواند الگویی از روند اجرایی تیم استخراج کند و در طول توسعه خصوصاً در جلسه برنامه ریزی اسپرینت از آن استفاده کند.

تأثیر این الگو از دو نظر قابل بررسی است: یکی از نظر ارائه راه حل و دیگری از نظر حالت پیشگیرانه. فرض شود تیم در اسپرینتی کار می کند که نمودار Sprint Burn Down در آن نشان می دهد که توسعه از زمان بندی عقب است (خط واقعی پایین تر از خط ایده آل باشد). قبل از اعمال راه حل اسکرام مستر به Sprint Burn Down قبل رجوع می کند و متوجه می شود که تیم در انتهای همه اسپرینت ها آیتم های بیشتری را نسبت به ابتدای آن ها تکمیل کرده است. درنتیجه پیش از هرگونه اخذ تصمیم، به تیم زمان بیشتری داده می شود تا آیتم های بیشتری را تکمیل کند. حالت پیشگیرانه هنگام برنامه ریزی صورت می گیرد. فرض شود که تیم در جلسه برنامه ریزی تصمیم گرفته چند آیتم بزرگ را برای یک اسپرینت انتخاب کند. اسکرام مستر طبق الگویی که از نمودار Sprint Burn Down اسپرینت های ماقبل تشخیص می دهد که تیم نمی تواند آن کارها را به موقع تکمیل نماید و از تیم درخواست تجدید نظر می کند.

نمودار Burn Down تنها برای اسپرینت ها قابل پیاده سازی نمی باشد؛ هر Release به طور جداگانه ای یک نمودار Burn Down دارد که به صورت ذکر شده کنترل و استفاده می شود. با این تفاوت که محور Xها اسپرینت ها را تشکیل می دهند و محور Yها مجموعه برآورد آیتم های بک لاگ محصول (بر حسب Story Point) می باشد. این نمودار در Release جهت ردگیری و نظارت بر روند اجرایی پروژه استفاده می شود. در این نمودار نیز از روش نقطه گذاری استفاده می شود. با هر بار اتمام یک اسپرینت نمودار به روز می شود(شکل 3-5).



شکل 5-3: روش نقطه گذاری برای رسم نمودار Sprint BurnDown

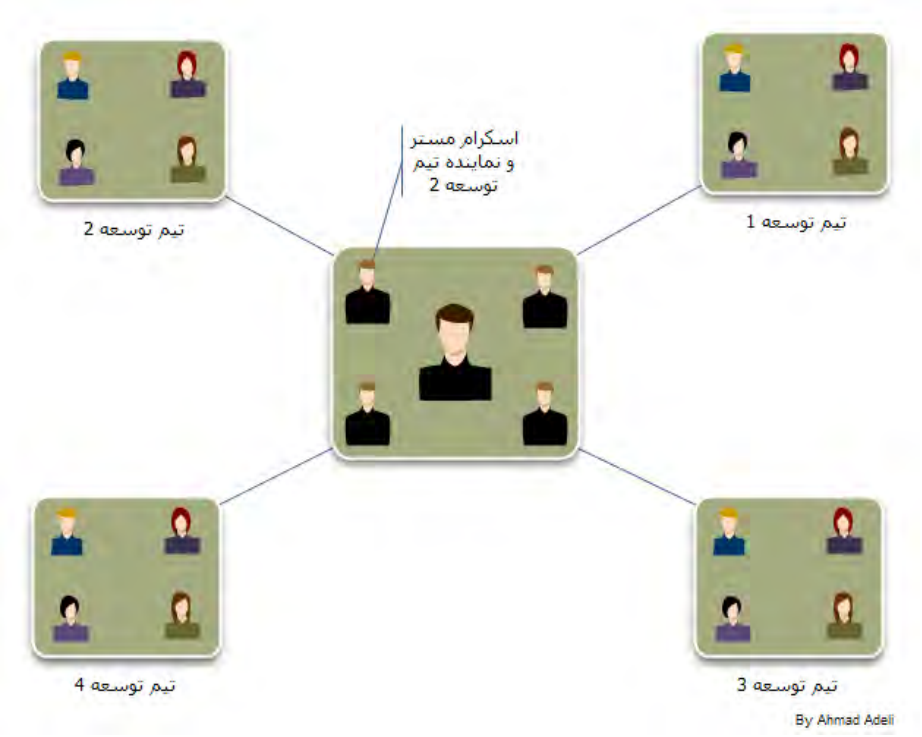
این نمودار به ما نشان می دهد که در اسپرینت های گذشته چه آیتم هایی تکمیل شدند و در Release جاری چه آیتم هایی باقی مانده اند. اسکرام مستر با استفاده از آن می تواند سرعت اسپرینت های بعدی را بهتر تخمین بزند. معمولاً پیش از اجرای هر Release جلسه ای برگزار می شود که در آن تعیین می شود که چه کارهایی و کی باید در Release انجام شوند. تعداد اسپرینت ها به عنوان زمان بندی Release در آن جلسه پیش بینی می شود.

نمودار Release Burn Down به ما نشان می دهد که برنامه ریزی انجام شده به درستی اجرا می شود یا خیر. فرض کنید در یک Release شش اسپرینت تعیین می شود با وجه به اینکه اسپرینت ها دارای زمان ثابتی هستند ، زمان بندی پایان اسپرینت مشخص و ابلاغ می شود. پس از اسپرینت شماره چهار اسکرام مستر در می یابد که با این در دو اسپرینت آینده تیم نمی تواند کل آیتم ها را تحویل دهد. بنابراین در جلسه اسپرینت بعدی افزایش سرعت اسپرینت و یا افزایش نیروی انسانی را به عنوان یک راه حل مطرح می کند. با نگه داری سابقه Release ها، اسکرام مستر می تواند الگویی را به عنوان روند اجرایی Release در توسعه از آن ها استخراج کند. بدیهی است که استفاده از این الگو در برنامه ریزی و زمان بندی Release های آتی اثر خواهد گذاشت.

نمایندگی گروه

در فصل دو اشاره به اجرای اسکرام به صورت سلسله مراتبی و چند گروهی شد. گفته شد که در گروه های بزرگ بهتر است افراد به چند گروه کوچک (مثلاً 10 نفره) تقسیم شوند و اسکرام در هر گروه به صورت جداگانه ای صورت می پذیرد. نماینده گروه ها به صورت یک کل گروهی را در سطح کلان تشکیل می دهند که به طور مستقیم وظیفه توسعه را بر عهده ندارد و وظیفه این گروه اغلب مرتبط با تصمیم گیری های استراتژیک می باشد. بدیهی است که همه اعضای تیم ها مانند آنچه در اسکرام دیده شد نم توانند در این گروه سطح کلان شرکت کنند؛ از این رو هر کدام از گروه های کوچک نماینده ای را به گروه سطح کلان می فرستند تا به جای آن در تصمیمات شرکت کند. نمایندگی گروه های اسکرام بر عهده اسکرام مستر است. هر کدام از گروه های سطح پایین توسعه قسمتی از پروژه را بر عهده دارند. کار گروه سطح کلان بررسی هر قسمت و برنامه ریزی

برای آن ها است. با این که در اسکرام مدیر پروژه تعریف نشده است اما در حالت کلان معمولاً شخصی به عنوان ارشد تعیین می شود که می تواند مدیر پروژه لقب بگیرد. این فرد یا از اسکرام مسترها گزینش می شود و یا به صورت خارجی به این سمت گماشته می شود(شکل 6-3).



شکل 6-3: اسکرام در سطح کلان

از دیگر وظایف اسکرام مستر در این خصوص ارائه گزارش های عملکرد تیم های خود در گروه سطح کلان می باشد. همچنین در صورت بر خوردن به مشکلی که حوزه آن فراتر از اختیارات یک تیم است و در تیم های دیگر نیز اثر گذار است؛ اسکرام مستر در گروه سطح کلان به ارائه مشکل پرداخته و آن را با دیگر اسکرام مسترها و به ویژه مدیر پروژه به اشتراک خواهد گذاشت. پس از اخذ تصمیمات اسکرام مسترها هر کدام باید این تصمیمات را به تیم های خود ابلاغ کنند و در صورت نیاز از آنها بازخورد گرفته و به مدیر پروژه مجدداً منتقل کنند.

بهره وری

بالا نگه داشتن بهره وری تیم از دیگر وظایف اسکرام مستر است. این وظیفه نیز مانند وظیفه پیشین یک وظیفه کلی است و از بخش های جزئی تر تشکیل می شود. هدف وظایفی که در زیر شرح داده خواهد شدT افزایش بهره وری تیم می باشد. اسکرام مستر با انجام آن ها بهره وری بالا را در توسعه تضمین می کند.

انجام جلسات کارا

جلسات عضو لاینفک اسکرام می باشند. برگزاری جلسات بسیار مهم و ضروری است. از این رو باید برای آن ها نیز برنامه ریزی شود. جلسه ای در اسکرام می تواند مفید باشد که صفت *کارایی* را یدک بکشد. یک جلسه کارا جلسه ای است که در زمان تعیین شده اهداف را ارضا کند. به عنوان مثال در جلسه روزانه اسپرینت اگر همه اعضا گزارش خود را جمعاً در 15 دقیقه ارائه دهند و جلسه بیش از این به طول نیانجامد، این جلسه می تواند یک جلسه کارا تلقی شود. جلسات به شرط کارا بودن *مفید* می باشند. جلساتی که کارا نباشند نه تنها مفید نخواهد بود بلکه موجب تلف شدن زمان توسعه نیز می شوند. حضور اسکرام مستر در همه جلسات الزامی است. چرا که او باید به اجرای جلسات نظارت داشته باشد و کارایی آن ها را در سطح بالایی نگه دارد.

اسکرام مستر ابتدا باید برای هر جلسه زمان مناسبی انتخاب نماید. زمان جلسه نه آنچنان باید زیاد باشد که موجب کسالت اعضا شود و نه آنچنان کم که اهداف جلسه ارضا نشوند. اسکرام مستر اگر تشخیص دهد جلسه بیش از حد معمول به خواهد انجامید، می تواند جلسه به چند جلسه با زمان های مناسب تر تقسیم کند.

تنظیم کردن زمان مناسب برای جلسه به تنهایی کافی نیست. در طول جلسات نیز اسکرام مستر باید روند اجرایی جلسه را به سمتی سوق دهد که جلسه بهره وری مناسبی داشته باشد. اول از همه حتماً باید برای جلسه برنامه ریزی شود تا از مطرح شدن مباحث بی ربط ممانعت به عمل آید. همچنین اسکرام مستر همواره در طول جلسه باید توجه اعضا را به اهداف تعیین شده جلب نماید. طولانی شدن یک بحث در جلسه باعث تلف شدن زمان جلسه یا افزایش زمان آن خواهد شد. اسکرام مستر باید در جلسات این گونه مباحث را کنترل نماید. به طور کلی نباید اجازه داد زمان جلسه از آن چه از پیش تعیین شده بیشتر شود چرا که باعث خستگی اعضا جلسه و متعاقباً موجب کاهش کارایی آن ها خواهد شد. بدیهی است که کاهش کارایی افراد روی کارایی کلی جلسه اثری سوء خواهد گذاشت.

کوچک نگه داشتن تیم ها

تعداد افرادی که در تیم اسکرام کار می کنند بسیار مهم است. تعداد افراد نباید آنچنان کم باشد که عملیات توسعه ناقص انجام شوند. باید برای هر نقش در توسعه (برنامه نویسی، تست و ...) حداقل یک نفر وجود داشته باشد. معمولاً مالک محصول عضو تیم توسعه به حساب نمی آید اما اسکرام مستر برعکس آن عضوی از تیم توسعه می باشد.

حداقل تعداد افراد برای یک تیم نوعی معمولاً 8 نفر است. این تعداد نباید از 12 نفر تجاوز کند(حداکثر 12 نفر). از آن جایی که تیم های اسکرام، تیم هایی خودسازمانده می باشند تعداد زیاد افراد باعث ایجاد هرج و مرج خواهد شد. در گروه شلوغ اغلب یک پارچگی از دست می رود و اعضای تیم نمی توانند خود را به خوبی متحد کنند. همچنین تعداد افراد زیاد، افزایش زمان جلسات و کاهش کارایی آن ها را در بر دارد.

در این گونه تیم ها معمولاً در بین اعضا چند دستگی به وجود می آید در صورتی که همبستگی یکی از ارکان مهم موفقیت هر تیم به حساب می آید. تداخل وظایف یکی دیگر از مضرات ازدیاد افراد تیم می باشد به این دلیل که به ازاری یک نقش در توسعه (یادآوری می شود که نقش های توسعه مانند برنامه نویسی و تست با نقش های اسکرام تفاوت دارند) افراد زیادی وجود دارد که باعث می شود برخی اوقات در وظایفشان تداخل ایجاد شود. به طور کلی اسکرام مستر باید تعداد افراد را در حد مناسبی نگه دارد و با احتیاط اقدام به تغییر آن نماید.

همخوان کردن افراد با مشاغل

در تیم توسعه هر فرد کاری را انجام می دهد. هر فردی به برخی شغل ها تمایل بیشتری دارند و برخی شغل ها را نمی پسندد. به دلیل اینکه هر کدام از افراد در توسعه چند مهارت مختلف را دارا هستند، ممکن است در قسمت های مختلفی نیز به کار گرفته شود. عدم رضایت فرد از شغلش باعث می شود او نتواند به خوبی روی کار خود متمرکز شود و در نتیجه بهره وری پایینی داشته باشد علاوه بر این احتمالاً کاری بی کیفیت را در نهایت ارائه خواهد داد. اسکرام مستر پیرو وظایفی که برای افزایش بهره وری در توسعه به او محول شده باید تضمین کند که افراد روی شغل هایی مناسب گماشته شده اند. گاهی در تیم های کوچک به علت کمبود نیروی انسانی این عمل کمتر اتفاق می افتد. در این حالت باید سعی شود تا افراد مذکور وظایفشان را مطابق میلشان انجام دهند؛ به نحوی که هم از آن رضایت کافی داشته باشند و هم به درستی و همنوا با توسعه انجام شود. نتیجه این همخوانی، کارایی بالاتر افراد و در نهایت حفظ کارایی توسعه می باشد.

تضمین می کند که هر توسعه دهنده ای مشغول کار روی آیتمی است

به دلیل خودسازمانده بودن تیم های اسکرام اغلب هر فرد خود کاری که انجام می دهد را بر می گزیند و از شخصی دیگری دستور مستقیم نمی گیرد. در تیم های هماهنگ معمولاً مشکل و تداخلی با این شیوه پیش نمی آید. اما در برخی تیم ها به خصوص آن هایی که افراد تازه کاری در آن ها کار می کنند، ممکن است برخی افراد در گزینش یک کار مستعصل شوند و کاری را نتوانند انجام دهند و یا اینکه فردی سهواً عملی را انجام می دهد که اصطلاحاً دارای ارزش تجاری نمی باشد و کاذب است.

وظیفه اسکرام مستر این است که این گونه حالات را سریعاً تشخیص دهد. تشخیص سریع به این دلیل است که به خاطر کوچک بودن تیم های اسکرام عدم حضور موثر یکی از اعضا به شدت محسوس خواهد بود که با نارضایتی دیگر اعضا و از دست رفتن زمان توسعه همراه است. پس از تشخیص چنین حالتی اسکرام مستر باید همخوان با مهارت شخص منفعل، کار موثری را به وی پیشنهاد کند و همچنین از با ارزش بودن اقدامات او اطمینان حاصل کند. در صورت عدم وجود وظیفه ای مناسب برای شخص مذکور، او می تواند با تکنیک هایی چون برنامه نویسی جفتی به دیگر اعضا در توسعه کمک کند.

پیش گیری از موانع و رفع آن ها هنگام وقوع
موانع به هر مشکلی گفته می شود که نمی توان آن را به راحتی حل نمود و روند توسعه را کند و یا در بدترین حالت متوقف می سازد. موانع کابوس همه توسعه دهندگان هستند. مواجهه با آن ها می تواند به قیمت از دست دادن کل زمان اسپرینت تمام شود. عدم دسترسی به یک تکنولوژی عدم وجود نرم افزار یا سخت افزار و حتی از دست دادن یکی از اعضای تیم توسعه می تواند نمونه هایی از یک مانع باشند.

یکی از ویژگی های مهم یک اسکرام مستر آینده نگری است. او باید بتواند وقوع چنین موانعی را پیش بینی کند و برای جلوگیری از آن تدبیری بیاندیشد. این آینده نگری و جلوگیری از مانع صرفاً وظیفه اسکرام مستر نمی باشد و همه اعضای تیم باید همواره با بررسی شرایط آینده برای رویارویی با چنین حالاتی آماده باشند. در حقیقت وظیفه اسکرام مستر سازماندهی این موانع و راه حل های آن ها می باشد. این گونه مشکلات عموماً در طول اسپرینت در جلسه روزانه اسکرام مطرح می شوند تا همه در جریان آن ها قرار گیرند. اما در این بین وظیفه اسکرام مستر است که مانع را تا بر طرف سازی کامل آن، مورد پیگرد قرار دهد. معمولاً موانع به صورت 100% قابل تشخیص و پیش بینی نیستند. گاهی اوقات پس از ظاهر شدن مانع تیم متوجه آن می شود. در این صورت تیم باید به سرعت و در کمترین زمان ممکن سعی در رفع مانع پیش آمده نماید.

گاهی نیز پیش می آید که راه حل در تیم وجود ندارد و تیم برای رفع مانع به منبعی خارجی متوسل می شود. یافتن این منبع خارجی به عهده شخص اسکرام مستر می باشد. او این منبع را که ممکن است نیروی انسانی(مثلاً یک مشاور) باشد را به توسعه منتقل می کند تا بدین وسیله راهی برای رفع مانع کشف شود. پس از بر طرف سازی مانع باید سعی شود که سابقه ای از مانع و راه حل رفع آن مستند شود تا در صورت بروز مشکلی مشابه آن در آینده، تیم قادر به حل سریع آن باشد.

تضمین امکانات مناسب
اعضای یک تیم هر چه که با مهارت و با تجربه باشند بدون امکانات مناسب کاری از پیش نخواهند برد. برای اینکه کارایی یک تیم بالا باشد و کاری با کیفیت تحویل دهد باید امکانات تهیه شده و در دسترس تیم قرار گیرد. این امکانات از هر نوعی می توانند باشند اعم از سخت افزار، نرم افزار و محیط اداری. با وجود اینکه تهیه این امکانات از وظایف اسکرام مستر نمی باشد، اما او باید با بررسی مرتب آن ها صحت کارایی آنها را تضمین کند.

همان طور که مشاهده شد نقش اسکرام مستر نقشی دشوار و پر وظیفه است. اما اسکرام مستر همواره در حال انجام چنین وظایفی نیست. اصولاً اسکرام مستر در وهله اول یک توسعه دهنده می باشد و در کنار توسعه به هدایت تیم اسکرام می پردازد. بنابراین دقیقاً مانند هر توسعه دهنده ای مهارت هایی دارد. توسعه دهنده بودن همراه با وظایف اسکرام مستر زمان و کار زیادی می طلبد. از این رو معمولاً اسکرام مسترها به نسبت دیگر توسعه دهندگان کار توسعه کمتری انجام می دهند.

توسعه دهنده
توسعه دهندگان یا اصطلاحاً مهندسان نرم افزار افرادی هستند که در اسکرام وظیفه توسعه و ساخت نرم افزار را برعهده دارند. به عبارتی دیگر در یک تیم اسکرام هر شخصی به غیر از مالک محصول و اسکرام مستر یک توسعه دهنده می باشند. توسعه دهندگان وظیفه انجام دادن کارهای اساسی توسعه را برعهده دارند. کارهایی چون تحلیل تجاری، برنامه نویسی، طراحی داده، تست، طراحی معماری، طراحی قطعات، طراحی رابط کاربری و ... از جمله کارهایی است که توسعه دهندگان در اسکرام انجام می دهند. هر کدام از توسعه دهندگان ممکن است مهارت های متفاوتی داشت باشند و در طول اسپرینت وظایف مختلفی بر عهده بگیرند.

با فرض آماده سازی بک لاگ محصول توسط مالک محصول، در جلسه برنامه ریزی اسپرینت توسعه دهندگان وظیفه دارند تا برای اسپرینت پیش رو آیتم هایی را که قرار است به انجام رسانند، گزینش کنند و در بک لاگ اسپرینت وارد سازند. این کار با حضور مالک محصول و اسکرام مستر که خود نیز یک توسعه دهنده می باشد به صورت گروهی انجام می شود؛ به این صورت که آیتم های انتخابی با توافق کل تیم انتخاب می شود. توافق کردن گروهی به دلیل کوچک بودن گروه کار چندان سختی نیست به ویژه برای گروه های با تجربه و هماهنگ. پس از آماده سازی بک لاگ اسپرینت تیم می تواند اسپرینت را آغاز کند. توسعه دهندگان با شروع اسپرینت، طراحی، برنامه نویسی و تست آیتم ها را آغاز می کنند. آنها وظیفه دارند کل آیتم های انتخابی را تا پایان اسپرینت به پایان برسانند. تکمیل نکردن آیتم های انتخابی به قیمت از دست دادن زمان، پول و اعتبار توسعه دهندگان تمام خواهد شد.

توسعه دهندگان باید حداکثر تلاش خود را برای آنچه در بک لاگ اسپرینت وجود دارد انجام دهند. تکمیل عجولانه آیتم ها باعث افت کیفیت قابلیت های خروجی خواهد شد که همین موضوع ممکن است باعث اعتراض مشتری در جلسه تحویل قابلیت شود و رد شدن این قابلیت ها را در پی داشته باشد. از آن جایی که انتخاب تعداد آیتم ها برای یک اسپرینت بر عهده خود توسعه دهندگان می باشد برخی توسعه دهندگان به فکر گزینش محدودتری می افتند. اما چنین کاری نیز غیر ممکن است چرا که درجلسه برنامه ریزی که گزینش آیتم های یک اسپرینت در آن انجام می شود، مالک محصول نیز حضور دارد. او بر خلاف تیم بسیار مایل است که در یک اسپرینت آیتم های بیشتری تکمیل شوند و کم بودن تعداد آیتم های انتخابی ممکن است این بار با اعتراض او همراه شود.

راه حل این مشکل توازن است. تیم باید به توازنی در انتخاب آیتم ها دست یابد که هم تعداد آنها موجب رضایت مالک محصول شود و هم این که تیم خود بتواند از پس تکمیل آیتم ها برآید. اما با این حال ممکن است به دلایلی برخی آیتم ها تکمیل نشوند. آیتم های ناقص مجدداً به بک لاگ محصول باز می گردند تا در اسپرینت های بعدی روی آنها بیشتر کار شود. ممکن است به دلیل عدم شناخت کافی از فرآیند توسعه، مشتری انتظاراتی غیر واقعی داشته باشد. در چنین صورتی مالک محصول وظیفه دارد که با مذاکره با مشتری انتظاراتشان را به سطحی منطقی بیاورد.

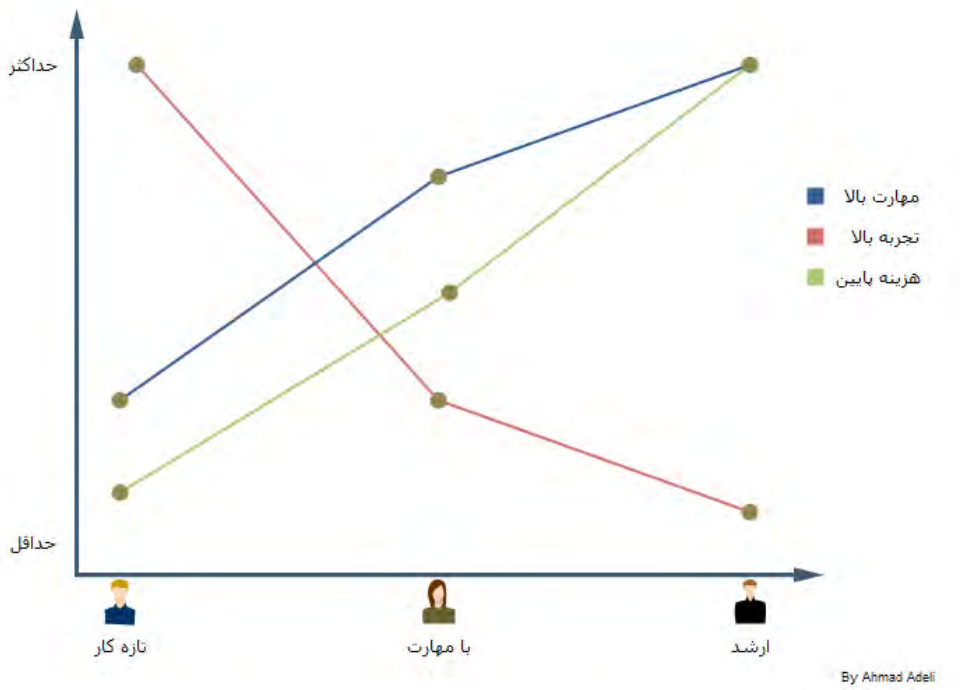
معمولاً هر توسعه دهنده ای با وجود مهارت های زیاد در یک مهارت تجربه بیشتری دارد. بهتر است در یک توسعه برای هر مهارت بیش از یک فرد وجود داشته باشد. دلیل آن حذف وابستگی مهارت در یک تیم است. فرض کنید در تیمی فقط طراح معماری وجود داشته باشد. با ساختن هر قطعه این فرد براساس قطعات جدید معماری را به روزرسانی می کند و سپس عمل یکپارچه سازی را روی آن انجام می دهد. حال اگر در میان توسعه این فرد از تیم کنار گذاشته شود(یا خود او کنار برود) تیم با مشکلی حاد مواجهه خواهد شد. از آنجایی که معماری بخش مهمی از توسعه نرم افزار است نمی توان برای مدت زیادی آن را به حالت تعلیق در آورد و یا از آن صرف نظر کرد. از این رو با فرض این که هیچ کدام از اعضای تیم چنین مهارتی را ندارند، آموزش یکی از اعضا احتمالاً می تواند بسیار زمان بر باشد. از طرفی افزودن یک عضو جدید به عنوان طراح معماری نیز ریسک بالایی دارد چرا که اولاً شناخت او از پروژه و به خصوص معماری موجب اتلاف زمان می شود؛ ثانیاً ممکن است تلاش های او برای شناخت معماری فعلی به شکست بیانجامد که در این صورت یا معماری مجدداً طراحی می شود و یا عضو جدیدی به جای او جایگزین خواهد شد که در هر دو حالت ممکن است زمان زیادی از پروژه به هدر رود و در بدترین حالت ممکن است پروژه با شکست مواجهه شود. از گفته های بالا پیدا است که نداشتن جایگزین برای یک مهارت عمل پر ریسکی است. چنین ریسکی برای پروژه های حساس ممکن است بسیار گران تمام شود پس بهتر است که با دقت در چینش تیم از چنین ریسکی پیشگیری به عمل آید.

در توسعه نرم افزار اعضای توسعه به سه دسته تقسیم می شوند: اعضای ارشد، اعضای با مهارت و اعضای تازه کار. یک عضو ارشد فردی است با مهارت، با تجربه و با توانایی های بسیار بالا. تعداد چنین افرادی در سازمان بسیار کم است و از نظر منابع نیروی انسانی جزو منابع بسیار با ارزش به شمار می روند. معمولاً این افراد محور اجرای فرآیند توسعه قرار داده می شوند. در هر تیم 8 تا 12 نفره اسکرام حداقل یک عضو ارشد باید حضور داشته باشد(توجه شود که دلیلی ندارد که این عضو الزاماً به عنوان اسکرام مستر شناخته شود).

اعضای با مهارت اعضایی هستند که مهارت های کافی را برای توسعه در اختیار دارند اما به اندازه اعضای ارشد در اجرای فرآیند توسعه با تجربه نیستند. از نظر درجه بندی، افراد با مهارت پایین تر از اعضای ارشد قرار می گیرند. اکثر اعضای تیم عضو این دسته از افراد هستند. آنها از تجربه و توانایی مناسبی جهت توسعه نرم افزار برخوردار هستند.

دسته سوم اعضای تازه کار هستند. این افراد همانطور که از نامشان پیدا است تازه کار و کم تجربه می باشند. اما از پتانسیل بالایی جهت یادگیری برخوردارند(بدیهی است که در صورت نداشتن این ویژگی در تیم توسعه جایی نخواهند داشت). آنها در حد قابل قبولی دارای مهارت های توسعه می باشند. بهتر است در پروژه هایی که حساسیت پایین تری دارند از ترکیب هر سه دسته استفاده شود. حضور اعضای ارشد در یک تیم می تواند موفقیت آن تیم را در وظایف محوله تضمین کند. اعضای با مهارت نیز می توانند خود از پس وظایف محوله برآیند و با اعضای ارشد ترکیب خوبی را تشکیل می دهند. اعضای تازه کار خود به خود با تجربه نخواهند شد. آنها باید در پروژه های واقعی حضور داشته باشند و مانند سایر اعضا وظایفی را انجا دهند تا به مهارت و تجربه آنها افزوده شود. برخی سازمان ها برای آموزش نیروی کاری خود چنین افرادی را در تیم ها قرار می دهند تا هم هزینه نیروی انسانی را کنترل کنند و هم برای سازمان به تربیت نیرو بپردازند. چنین آموزشی در یک تیم نیاز به اشتراک اطلاعات توسط اعضای تیم به خصوص اعضای ارشد خواهد داشت. این اشتراک اطلاعات علاوه بر

مزایایی که برای کل تیم و متعاقباً سازمان خواهد داشت باعث افزایش دانش و مهارت خود اعضا نیز می شود(شکل 7-3).



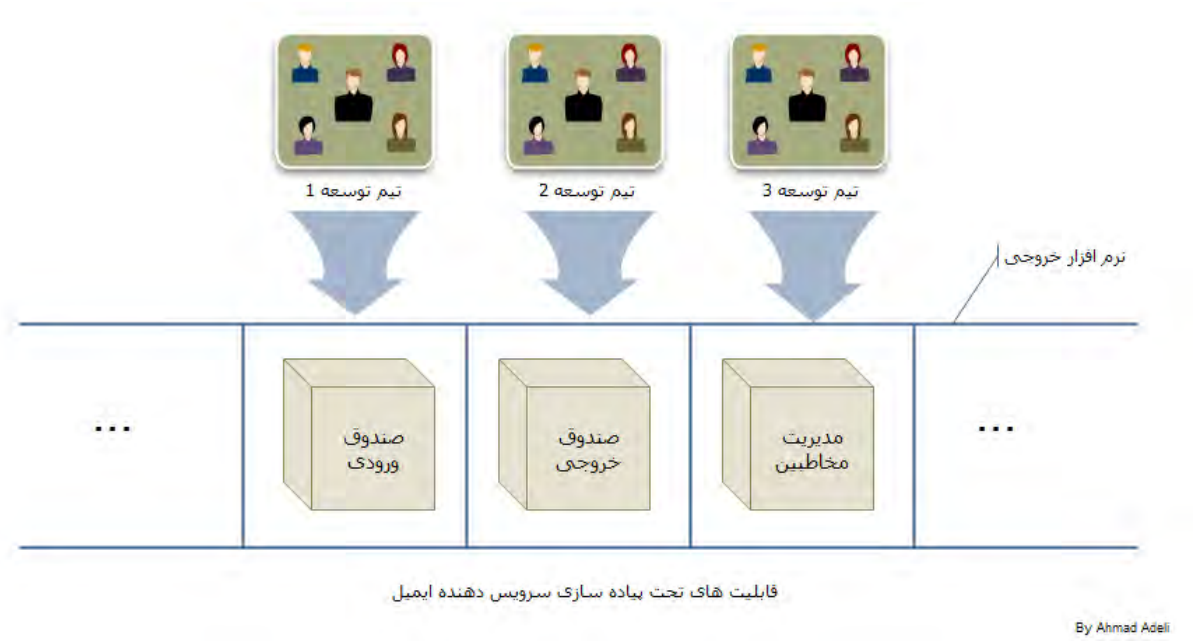
شکل 7-3: مقایسه انواع توسعه دهندگان

مشتریان، نرم افزارها را بر اساس قابلیت های آنهای می شناسند. به این دلیل که قابلیت های نرم افزار تنها چیزی است که کاربران می توانند مشاهده کنند. آن ها از درون نرم افزارها و چگونگی ساخت آنها بی اطلاع هستند. به همین دلیل تمایل دارند که آنچه هر بار تحویل می گیرند یک قابلیت از نرم افزار باشد. به تحویلی که بر اساس قابلیت های نرم افزار باشد تحویل *مشتری محور* گفته می شود. با این کار توسعه دهندگان در طول توسعه بیشتر بر روی نیازهای مشتری توجه خواهند کرد. از طرفی مشتری هم به دلیل شناخت مناسب و کافی بیشتر تمایل پیدا می کند که با فرآیند توسعه همکاری کند.

یک گروه اسکرام در مقیاس بزرگ ممکن است از چندین تیم کوچک تشکیل شود. تقسیم وظایف بین این تیم ها یکی از بزرگترین دغدغه های مدیر پروژه می باشد. سه شیوه تقسیم وظایفِ بین تیمی در اسکرام وجود دارد که در زیر به شرح آنها خواهیم پرداخت.

تقسیم وظایف بر اساس قابلیت

در این شیوه هر تیمی برخی قابلیت های نرم افزار را توسعه می دهد. ناگفته پیدا است که تحویل ها در پایان هر اسپرینت مشتری محور خواهند بود چرا که توسعه دقیقاً بر اساس قابلیت هایی انجام می شود که مشتری طلب کرده است. در این نوع توسعه برای تحویل یک قابلیت، یک تیم نوعی باید همه کارهای مرسوم در توسعه از قبیل طراحی، برنامه نویسی، تست را خود انجام دهد. سایر توسعه دهندگان در باقی تیم ها نیز به همین صورت قابلیت های دیگر نرم افزار را توسعه داده و تحویل می دهند. این باعث کاهش وابستگی تیم ها و توسعه دهندگان به یکدیگر خواهد شد. کاهش وابستگی خود موجب شدن تعاملات بین تیمی و تسریع تحویل قابلیت های جدید را سبب می شود. ایجاد یک قابلیت از ابتدا تا انتها توسط یک تیم باعث می شود توسعه دهندگان بتوانند کیفیت قابلیت های توسعه داده شده را تضمین نمایند. همچنین پیگیری مشکلات توسعه از این طریق آسان تر می شود چرا که دقیقاً مشخص است که یک قابلیت نوعی در کدام تیم و توسط کدام توسعه دهنده ایجاد شده است. از این شیوه همواره به عنوان با اولویت ترین و بهترین شیوه در بین توسعه دهندگان یاد می شود(شکل 8-3).



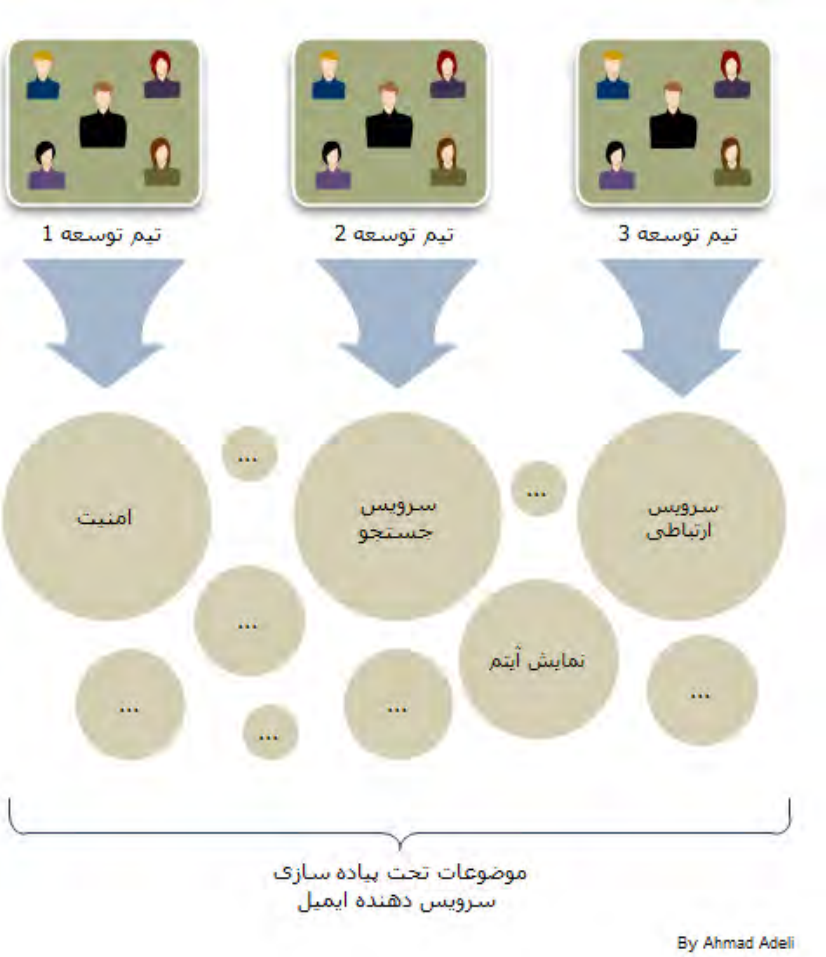
شکل 8-3: تقسیم وظایف بر اساس قابلیت

اما شیوه تقسیم وظایف بر اساس قابلیت، در کنار مزایای خوبی که دارد مشکلاتی را نیز می تواند به وجود آورد. در صورت عدم کنترل مناسب توسعه، بخش های نرم افزار به تکه های مستقلی تبدیل خواهد شد که یکپارچگی بسیار پایینی خواهند داشت. برای جلوگیری از این تکه تکه شدن، باید یک معماری جامع و یکپارچه برای کل نرم افزار طراحی شود. بهتر است معماری های نرم افزار در تیم ها در سطح کلان با یکدیگر تعامل داشته باشند تا نرم افزار معماری واحدتری داشته باشد. تعریف و اعمال استانداردهای طراحی نه تنها در طراحی معماری بلکه در همه زمینه ها باعث حفظ هارمونیک بین قطعات نرم افزار خواهد شد.

تقسیم بندی تیم ها بر اساس زیر سیستم ها می تواند گزینه مناسبی باشد چرا که در برخی نرم افزارها زیر سیستم ها دارای معماری مجزایی هستند. از این رو استقلال تیمی را می توان به صورت کامل اعمال نمود. اما با این حال نیز باز هم یک معماری کلی وجود دارد که باید در سطح فراتیمی طراحی و پیاده سازی شود. مزیت تیم های مستقل، حداقل سازی اصطکاک آنها می باشد. اما همین کاهش تعاملات خود می تواند باعث مشکلات دیگری شود. تعاملات پایین بین تیمی باعث می شودکه هر تیم از قطاعات ساخته شده توسط دیگر تیم ها بی اطلاع باشد و نتواند به آنها دسترسی داشته باشد. این حالت موجب دوباره سازی قطعاتی در نرم افزار می شود که یکسان هستند. دوباره سازی قطعات ساز پیش ساخته شده نه تنها باعث اتلاف وقت و هزینه خواهد شد بلکه به دلیل تفاوت در عمکرد قطعات یکسان، به پایداری کل سیستم لطمه وارد می کند.

تقسیم بر اساس موضوع

در این نوع تقسیم بندی تیم ها صرف نظر از زیر سیستم ها و ویژگی های آنها روی موضوعات واحد در پروژه کار خواهند کرد. به عنوان مثال در یک سرویس دهنده ایمیل ممکن است تیمی روی بخش سرویس جستجو کار کند. جستجوهای مختلفی می توانند در این سرویس دهنده وجود داشته باشند مانند جستجو ایمیل ها، جستجو مخاطبین و ... . تیمی که روی بخش جستجو کار می کند مسئول ساخت همه جستجوها می شود. تیم دیگری می تواند موضوع دیگری را انتخاب کند و مثلاً مسئول کار روی مسائل امنیتی سرویس دهنده ایمیل شود. این مباحث امنیتی می تواند موضوعاتی از قبیل تعریف زیر ساخت کلی امنیت، طراحی لایه های امنیتی در معماری و حتی تست امنیت سایر قطعات باشد(شکل 9-3).



شکل 9-3: تقسیم بر اساس موضوع

مزیت این روش این است که به وسیله آن می توان پیچیده ترین قابلیت های را با صرف زمانی کم و کیفیتی بالا پیاده سازی نمود. فرض کنید در مثال سرویس دهنده ایمیل قابلیت جستجو در زیر سیستم های مختلف توسط تیم های متفاوت ساخته شود. اگر مشتری قابلیت ایندکس گذاری را برای جستجوها طلب کند، ممکن است تیمی نتواند چنین خواسته ای را برآورده کند، همچنین در صورتی که همه تیم ها قادر به پیاده سازی این قابلیت باشند هر کدام باید به صورت جداگانه این عمل را انجام دهند جز افزایش هزینه چیزی در بر نخواهد داشت. این وضع در حالی حادتر می شود که بخواهیم این جستجوها را برای دیگری اعمال کنیم. بدیهی است که ساختن چنین قابلیت های پیچیده ای را به سختی می توان تضمین نمود. اما اگر تیمی به صورت تخصصی روی یک قابلیت در نرم افزار کار کند بهتر می تواند کیفیت قابلیت را (با وجود پیچیدگی های آن) کنترل کند. مزیت دیگر این روش سازگاری در معماری می باشد. به علت عدم طراحی های متفاوت در بخش های مختلف معماری سازگاری و یکپارچگی بهتری حاصل خواهد شد.

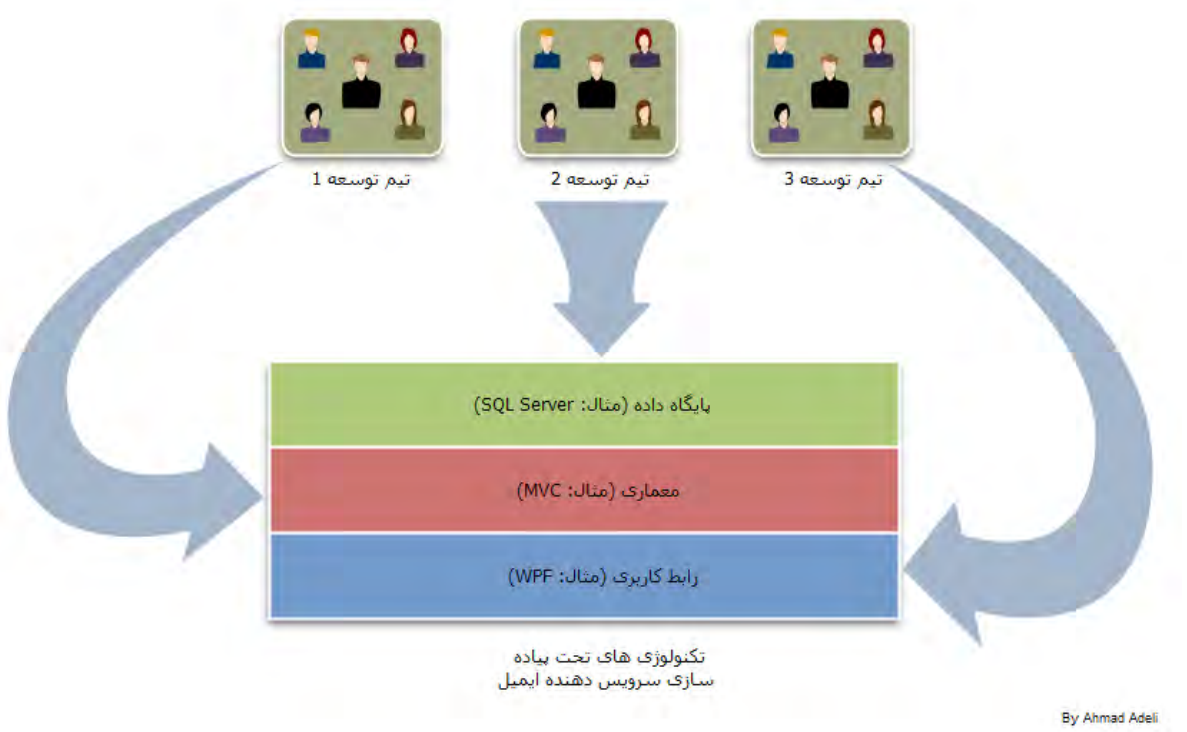
روش موضوع محور هم مانند روش پیشین مشکلاتی را ممکن است ایجاد کند. اولین و بارزترین اشکالی که این روش ایجاد می کند این است که تحویل قابلیت در آن مشتری محور نمی باشد. تحویل ها بشتر بر اساس اصطلاحات و سندات تخصصی شکل خواهد گرفت که همین موضوع باعث تعامل ضعیف با مشتری خواهد شد. یکی دیگر از مشکلات روش موضوع محور حوزه محدود دید توسعه دهندگان در توسعه می باشد. در روش قبلی توسعه بر اساس قابلیت بود و هر توسعه دهنده در هر تیمی حوزه دید کاملی از نرم افزار داشت و از ساخت آنچه در قسمت های دیگر نرم افزار (یا حداقل در قسمت های دیگر زیر سیستم که در آن کار می کند) آگاهی کامل داشت.

ولی در روش جاری چنین دیدی فقط به موضوع مربوطه محدود شده و موجب می شود که توسعه دهندگان نرم افزار را به دید یک کل مشاهده نکنند. این دید محدود، قدرت طراحی بخش های مرتبط را از توسعه دهندگان تا حدودی خواهد گرفت. بدیهی است که در پروژه بزرگ هر چند هم که از روش قابلیت محور استفاده شود داشتن

حوزه دید کامل روی نرم افزار غیر ممکن خواهد بود. برای استفاده از روش موضوع محور تیم ها مجبور هستند برای ارتباط موضوع تحت توسعه خود با دیگر بخش های نرم افزار، سریعاً از همان ابتدا رابط (InTeam Explorerrface) های خود را طراحی و ارائه نمایند. این گونه رابط ها باید بسیار با دقت طراحی شوند چرا که در مقابل تغییر انعطاف خوبی از خودشان نشان نمی دهند. این رابط ها معمولاً پیش از خود قطعات ساخته می شوند(رابط ها اساس ارتباط سیستم را تشکیل می دهند). تغییر این رابط ها در حین توسعه هزینه های سنگینی را می تواند به پروژه تحمیل کند. روش تقسیم موضوع محور نسبت به روش قابلیت محور برای بسیاری از توسعه ها و گروه ها اولیت پایین تری دارد.

تقسیم بر اساس تکنولوژی

در این روش تقسیم وظایف بر اساس تکنولوژی های پیاده سازی شده در نرم افزار صورت خواهد گرفت. به عنوان مثال تیمی که مسئول توسعه پایگاه داده می باشد، همه اعضای آن متخصصین تکنولوژی های پایگاه داده هستند. آنها همه وظایف طراحی، تست، نگهداری پایگاه داده های نرم افزار را بر عهده می گیرند. تیمی دیگر ممکن است روی رابط کاربری متمرکز شود و در آن زمینه به کار بپردازد. شکل تیم های که از این روش استفاده می کنند با آنچه تا به حال گفته شد متفاوت هستند. وظایفی که توسعه دهندگان در این تیم ها انجام می دهند محدود است و اعضای تیم از مهارت های مشابهی برخوردارند(شکل 10-3).



شکل 10-3: تقسیم بر اساس تکنولوژی

مزیت این روش این است که خروجی کار تیم ها بسیار با کیفیت است. این کیفیت از آنجایی ناشی می شود که اعضا فقط روی یکی از مهارت های خود در توسعه متمرکز می شوند . گروه آنها یک گروه کاملاً تخصصی می باشد. تیم ها و سازمان ها از روش بسیار کمتر سایر روش ها استفاده می کنند؛ به این دلیل که مشکلات آن بسیار بیشتر از مزایای آن می باشد. این روش تیم ها را عملاً به گروه های کاملاً مستقلی تبدیل می کند که کمتر با هم در ارتباط هستند که نتیجه آن ارتباط ضعیف در بخش های توسعه نرم افزار باشد. همچنین در این روش هیچ تیمی مسئول تحویل قابلیت های نرم افزار نمی باشد. مدیر پروژه نیز کمتر می تواند قابلیت های ساخته شده را ردگیری کند. در این بین احتمالاً تعداد زیادی میان افزار باید ساخت شود تا ارتباط این بخش ها را با یکدیگر برقرار کند. طراحی معماری بر اساس این واحدهای مستقل ممکن است کمی مشکل باشد و همچنین افراد بیشتری روی پیچیدگی های فنی تکنولوژی تحت مسئولیت خود کار خواهند کرد و احتمالاً کسی حل پیچیدگی های فنی خود نرم افزار را تضمین نخواهد کرد. معمولاً این روش به عنوان آخرین اولویت از بین این سه روش انتخاب می شود.

انتخاب روش تقسیم بندی بر اساس سیاست های سازمانی و نوع پروژه انتخاب می شود. هر کدام از روش های ذکر شده مزایا و معایب خاص خود را دارند. روشی که برای تقسیم بندی استفاده می شود می تواند تأثیر بسزایی بر روی موفقیت یا شکست پروژه بگذارد. مدیران با تجربه می دانند که روش تقسیم تیم ها را باید پیش از گزینش نیروی انسانی انتخاب کنند چرا که گزینش توسعه دهندگان تا حدودی بر اساس این تقسیم بندی انجام می شود.

دیگر نقش های موجود در اسکرام

علاوه بر سه نقش اصلی اسکرام یعنی مالک محصول، اسکرام مستر و توسعه دهنده نقش های جانبی دیگری نیز در اسکرام وجود دارند. سه نقش مذکور با یکدیگر یک تیم اسکرام کامل را تشکیل می دهند.دیگر نقش ها معمولاً خارج از تیم و در سطح کلان قرار دارند. که گاهی آنها را نقش های IT نیز می نامند. در ادامه سه نقش از این دسته(IT) معرفی خواهد شد.

مدیر پروژه

در پروژه های متوسط و بزرگ معمولاً بیش از یک تیم اسکرام مشغول کار هستند. حالت چند تیمی را نمی توان به صورت خودسازمانده اداره کرد و از طرفی بر خلاف پروژه های کوچک پیچیدگی وظایفی چون مدیریت نیروی انسانی، تخمین بودجه، زمان بندی، برنامه ریزی پروژه بیشتر است. از این رو معمولاً شخصی به عنوان مدیر پروژه انتخاب می شود و این وظایف به او محول خواهد شد. این فرد وظایفی دیگری چون اخذ تصمیمات استراتژیک و تعریف استاندارد ها را نیز بر عهده دارد. مدیر پروژه اغلب با نمایندگان تیم ها ارتباط دارد تا خود توسعه دهندگان اما گاهی جلساتی با مالکین محصول تیم ها نیز بر گذار می شود. عملیات آغازین پروژه (پیش از توسعه) که در فصل پیش معرفی شد در حالت چند تیمی جزو وظایف مدیر پروژه خواهد بود و هیچ تیمی مستقیماً خود را در گیر آن نمی کند.

انتساب یک مدیر در صدر پروژه به معنای نقض خودسازماندهی تیم ها نیست. مدیر صرفاً می تواند در امور کلی و کلان دخالت داشته باشد و مدیریت امور توسعه در هر بخش در حیطه تصمیمات تیم مربوطه خواهد بود و تیم ها با وجود مدیر پروژه باز هم استقلال خود را از دست نخواهد داد . مدیر پروژه بیشتر نقش هماهنگ کننده بین تیم ها را بر عهده دارد و به طور کلی هر تصمیمی بین تیم ها مشترک باشد از حوزه اختیارات آن ها خارج شده و تحت وظایف مدیر پروژه محسوب می گردد. مدیر پروژه نیز احتمالاً به وسیله شورایی متشکل از نمایندگان تیم ها(اسکرام مستر ها) این تصمیمات را اخذ خواهد کرد. وظایف مدیر پروژه تا حد زیادی به سازمان مشتری ارتباط خواهد داشت که ممکن است در پروژه های مختلف متفاوت نیز باشد. بنابراین به طور دقیق نمی توان این نقش را مورد بررسی و تحلیل قرار داد. ناگفته نماند این نقش در حالتی چند تیمی و کلان تعریف می شود و اساساً در پروژه های کوچک موضوعیت ندارد.

معمار نرم افزار

معمار نرم افزار شخصی است که مسئول حل پیچیدگی های فنی ارتباط و یکپارچه سازی قطعات یک نرم افزار یا سیستم می باشد. زیر ساخت هر نرم افزاری معماری آن می باشد و معمار نرم افزار وظیفه دارد این معماری را طراحی کند و تا پایان پروژه مدیریت نماید. معمولاً پروژه های کوچک و تک تیمی فردی از داخل تیم(مثلاً یکی از توسعه دهندگان ارشد) این وظیفه را بر عهده می گیرد. اما در سطح کلان و چند تیمی این وظیفه از حوزه تیم خارج شده و در سطح کلان مدیریت می شود. معمولاً برای پروژه های بزرگ نیز تعداد محدودی معمار تعریف می شود، این افرد به طراحی معماری نرم افزار به صورت یک کل خواهند پرداخت. در زیر سیستم هایی که به صورت مستقل توسعه می یابند طراحی معماری به صورت محلی و در سطح تیم ها صورت خواهد گرفت. بدیهی است هر چه که یک زیر سیستم مستقل باشد باز هم ارتباطی با معماری کلی نرم افزار خواهد داشت به همین دلیل معمولاً بین معماران کل و معماران محلی ارتباط نزدیکی برقرار می شود.

مدیر کیفیت

مدیر کیفیت نقشی است که چندان در توسعه نرم افزار مرسوم نیست. مدیر کیفیت وظیفه دارد کیفیت نرم افزار را به عنوان یک محصول نهایی تضمین کند. در حقیقت مدیریت کیفیت وظیفه ای از وظایف مدیر پروژه می باشد و برای افزایش بهره وری به دیگری تفویض می شود. مدیر کیفیت وظیفه دارد که استانداردهای کیفیتی را برای توسعه تعریف و اعمال نماید.

همچنین او عملکرد نظارتی در تست شدن قابلیت های نرم افزار توسعه داده شده را نیز بر عهده دارد. نظارت او از این جهت است که اطمینان حاصل شود که مالک محصول برای هر قابلیت طلب شده تست کیسی را تعریف

کرده است و از طرف دیگر نیز توسعه دهندگان این تست ها را پیش از تحویل پاس خواهند نمود. ممکن است پروژه به دلیل سطح حساسیت بالا نیاز به اجرای *مرور* (Review) در راستای اعتبار سنجی قطعات ساخته شده داشته باشد. از دیگر وظایف مدیر کیفیت تعریف انواع اعتبار سنجی و نظارت بر اجرای آن در پروژه می باشد.

نقش های درون یک پروژه اساساً نوعی دسته بندی از وظایف و حتی ویژگی های افراد توسعه دهنده در پروژه می باشند. در توسعه نرم افزار بسته به سایز و نوع پروژه اگر به انجام کاری نیاز شود دو راه وجود دارد: یا وظیفه ای جدید به دیگر وظایف یک نقش محول می شود و یا به صورت مجزا برای آن نقشی تعیین می شود. در این فصل سه نقش اصلی و پرکاربرد اسکرام معرفی شد. همچنین در این فصل به سه نقش مرسوم در حوزه کلان اشاره ای مختصر شد. باید به این نکته توجه شود که معرفی این نقش ها دلیلی بر پیاده سازی همه آن ها به صورت یکجا در یک پروژه وجود ندارد. به دلیل انعطاف بالای اسکرام تعریف و اضافه کردن نقش ها و وظایف جدید برای یک پروژه در صورت نیاز وجود دارد. تجربه نشان داده است که سفارشی سازی اسکرام بر اساس سازمان، پروژه و افراد نسبت به اجرای خشک قوانین، میزان بهره وری را تا حد قابل ملاحضه ای افزایش خواهد داد.

فصل چهارم: آیتم های کاری

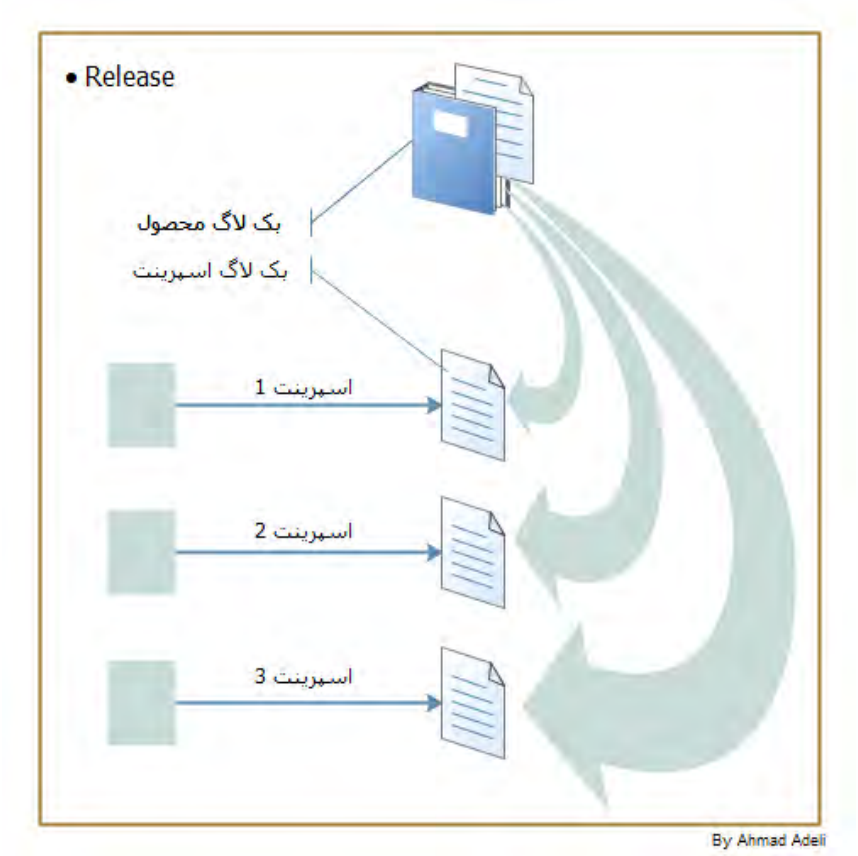
آیتم های کاری مستنداتی هستند که دارای ارزش تجاری نمی باشند اما برای اجرای فرآیند توسعه ضروری هستند. آنها معمولاً خروجی جلسات و یا خروجی وظایفی هستند که در توسعه انجام می شود. هیچ کدام از این آیتم ها به مشتری تحویل داده نمی شوند؛ آنچه که مشتری به طور مستقیم می تواند ببیند قابلیت های تکمیل شده نرم افزار است. آیتم های کاری صرفاً توسط اعضای تیم در توسعه مصرف می شوند. شناخت این گونه آیتم ها که اغلب به صورت سند بین اعضا منتقل می شود و همچنین چگونگی استفاده از آن در راستای اجرای موفقیت آمیز اسکرام امری ضروری است.

بک لاگ محصول

اولین و مهمترین آیتمی که معرفی می شود بک لاگ محصول است. همانطورکه پیشتر ذکر شد بک لاگ محصول در حقیقت لیستی از قابلیت های بالقوه نرم افزار می باشد که باید در طول توسعه پیاده سازی شوند. از طریق این لیست به توسعه دهندگان گفته می شود که برای توسعه نرم افزار چه آیتم هایی نیاز است که ساخته شود. آیتم های این لیست از دید مشتری ویژگی ها و قابلیت های نرم افزار مورد انتظارشان می باشد. انتظار می رود همه آیتم هایی که در این لیست وجود دارند(قابلیت های باقوه) پس از توسعه در نهایت به یک نرم افزار کارکننده تبدیل شوند.

بک لاگ محصول آیتمی است که تحت تملکِ صرفِ مالک محصول می باشد. در اسکرام تنها شخصی که می تواند در لیست تغییرات ایجاد کند، صرفاً مالک محصول است. مالک محصول معمولاً پیش از اجرای Release این لیست را کامل می کند(این حالت فقط به صورت تئوریک اتفاق می افتد چرا که در عمل هرگز نمی توان کل نیازمندی ها را به صورت یکجا کشف کرد. گفتنی است که این لیست در طول Release بارها تغییر خواهد کرد). او وظیفه دارد انتظارات مشتری را به آیتم های بک لاگ محصول تبدیل کند. بک لاگ محصول برای هر نقشی غیر از مالک محصول فقط خواندنی است.

در فصل ها پیشین دو نوع بک لاگ محصول معرفی شد. بک لاگ محصول و بک لاگ اسپرینت. بک لاگ محصول در حقیقت یک لیست مرکزی است که آیتم های بک لاگ اولین بار در آن وارد می شوند. برای هر Release تنها یک بک لاگ محصول نگه داری می شود. آیتم هایی که در یک بک لاگ محصول وارد می شوند باید در اسپرینت های یک Release پیاده سازی شوند. بدین وسیله هر کدام از اسپرینت ها دارای یک بک لاگ مخصوص خود می شوند که به بک لاگ اسپرینت موسوم است. بک لاگ اسپرینت لیست جداگانه ای از آیتم ها نمی باشد و در حقیقت شامل آیتم های بک لاگ محصول می باشد(شکل 3-11).



شکل 3-11: هر بک لاگ محصول وظیفه تغذیه همه اسپرینت های درون Release را بر عهده دارد.

در جلسه برنامه ریزی اسپرینت تیم قسمتی از آیتم های بک لاگ محصول را انتخاب کرده و وارد بک لاگ اسپرینت می کند(رجوع شود به شکل 2-6). به عبارت دیگر تنها در یک Ralease یک بک لاگ بیشتر وجود ندارد و بک لاگ اسپرینت از انتقال آیتم های بک لاگ محصول به دورن آن تشکیل می شود. اگر آیتمی به هر دلیل در یک اسپرینت کامل نشود، مجدداً به بک لاگ اسپرینت منتقل می شود تا در اسپرینت های دیگر دوباره برای تکمیل آن تلاش شود.

آیتم های بک لاگ باید به صورت یک ویژگی نوشته شوند. ویژگی یا قابلیت به جزییاتی گفته می شود که در نرم افزار باید وجود داشته باشد. به عنوان مثال در یک سرویس دهنده ایمیل ممکن است مشتری(کسی که سرویس دهنده را سفارش داده) امکانی را طلب کند که در آن لیستی از مخاطبین اخیر در صفحه اصلی نمایش داده شود. این امکان خود به تنهایی یک قابلیت می باشد و مالک محصول باید آن وارد بک لاگ محصول سازد(شکل 3-12).

شکل 12-3: مثال قابلیت های کشف شده برای پروژه سرویس دهنده ایمیل

قابلیت هایی که برای نرم افزار در یک لاگ اضافه می شوند باید صریح و به دور از ابهام باشند؛ به عنوان مثال مدیریت کردن اعضای وب سایت درخواست چندان شفافی نیست و مالک محصول نباید آن را به این صورت در یک لاگ محصول وارد کند. قابلیت های این چنینی بیش از حد بزرگ، کلی و گنگ هستند. بنابراین مالک محصول وظیفه دارد با شناخت دقیق بر آنچه مشتری طالب است، یک قابلیت بزرگ را به قابلیت های کوچک تری و جرئی تری خرد کند و سپس آن ها را وارد یک لاگ محصول کند.

نکته بسیار مهمی که وجود دارد این است که در آیتم های یک لاگ به هیچ وجه نباید از اصطلاحات فنی استفاده شود چرا که این لیست بین تیم و مشتری مشترک است پس در نتیجه هر دو طرف باید بتوانند به راحتی با آن ارتباط بر قرار کنند. برای مشتری مهم نیست که آیتم ها چگونه پیاده سازی خواهند شد؛ آنچه آنها در انتها تحویل خواهند گرفت قابلیت های واقعی یک نرم افزار کارکننده می باشد. نوشتن آیتم های یک لاگ امری است که با تجربه حاصل می شود. مالک محصول باید بتواند یک آیتم را به صورت کوچکترین ویژگی قابل فهم توسط مشتری بیان کند. ویژگی باید نه آنقدر کوچک باشد که مسائل فنی توسعه را نمایان کند و نه آنقدر بزرگ که برای تیم توسعه گنگ و نامفهوم باشد. توازن کردن اندازه این ویژگی ها با کمی سعی و خطا امکان پذیر است.

هر آیتم یک لاگ خود از بخش های دیگری تشکیل می شود. هر کدام از این بخش ها قسمتی از آیتم را شرح می دهند. برخی از بخش ها الزامی می باشند و حضور آنها در نوشتن یک آیتم اجباری اما برخی صرفاً برای اطلاعات بیشتر مورد استفاده قرار می گیرند و اختیاری هستند. در ادامه بخش های تشکیل دهنده آیتم یک لاگ معرفی می شوند(شکل 13-3).

عنوان	اولویت	برآورد	شرایط تطبیق
نمایش لیست نامه ها	100	8	لیست نامه ها با عنوان نامه و نام ارسال کننده نمایش می یابد. نامه خوانده نشده در ابتدای لیست قرار گرفته و با رنگی مجزا از نامه های خوانده شده تمیز داده می شوند.
جستجو مخاطبین	98	6	مخاطبین با استفاده از نام یا ایمیلشان مورد جستجو قرار می گیرند. با تایپ ابتدای نام یک مخاطب لیستی مطابق با آن کلمه کلیدی همزمان نمایش داده می شود.
...	...	...	...
برچسب گذاری نامه ها	87	4	برای دسته بندی نامه ها هر نامه می تواند دارای یک یا بیش از یک برچسب باشد. با جستجو هر برچسب نامه هایی که دارای آن برچسب می باشند نمایش می یابند.
ثبت خودکار مخاطبین	85	6	ایمیل هایی که در نامه های ارسالی یا دریافتی وجود دارند به طور خودکار به لیست مخاطبین افزوده خواهند شد. برای تکمیل اطلاعات مخاطب از اطلاعات موجود در نامه استفاده می شود.
...	...	...	...
ارسال زمان بندی شده نامه ها	52	10	نامه ها می توانند برای ارسال زمان بندی شوند. یک نامه پس آماده شدن ذخیره شده و زمانی متشکل روز ، ساعت و دقیقه تعیین می گردد تا به محض رسیدن به این زمان نامه به صورت خودکار ارسال شود.
...	...	...	...

شکل 13-3: مثال یک لاگ محصول برای پروژه سرویس دهنده ایمیل.
لازم به ذکر است که برآورد آیتم ها هنگام انتقال آنها به یک لاگ اسپرینت تعیین خواهند شد.

عنوان

عنوان، نامی است که برای سهولت استفاده از یک آیتم یک لاگ به آن آیتم اتلاق می گردد. عنوان نباید خیلی طولانی باشد تا کاربر برای خواندن آن دچار مشکل شود. همچنین عنوان کوتاه نیز معمولاً موجب ابهام می شود. توسعه دهندگان عموماً از 3 تا 5 کلمه برای نوشتن عنوان استفاده می کنند. متن عنوان بهتر است خبری باشد تا بتواند مفهوم آیتم را رساتر منتقل کند مانند "نمایش لیست نامه ها". در متن عنوان نباید از کلمات یا عبارات پیچیده و چند پهلو استفاده شود. به طور کلی عنوان باید ساده و صریح نوشته شود تا مفهوم خود را سریع و مفید منتقل کند.

اولویت

در توسعه بسیار مهم است که آیتم ها با چه ترتیبی پیاده سازی می شوند. یک لاگ محصول لیستی مرتب شده است. این مرتب سازی بر اساس اولویت صورت می پذیرد. اولویت ها به وسیله اعداد صحیح مشخص می شوند. هیچ قانونی برای اینکه از چه بازه اعدادی استفاده شود یا چگونه استفاده شود وجود ندارد. این بازه می تواند بین 0 تا 10، 0 تا 100 و یا هر محدوده دلخواهی باشد. آنچه که مهم است این است که به آیتم هایی که قرار است زودتر پیاده سازی شوند عدد کمتری نسبت داده شود تا با مرتب سازی نزولی این گونه آیتم ها در

صدر لیست ظاهر شوند. برای تغییر اولویت نیز کافی است عدد اولویت هر آیتم را کم یا زیاد کنیم. از آنجایی که ممکن است شناختی روی اولویت آیتم هایی که قرار است در آینده اضافه شوند وجود نداشته باشد پس بهتر است عدد اولویت آیتم ها به گونه ای انتخاب شود که در آینده بتوان بین آن ها آیتمی قرار داده شود. به عنوان مثال اولویت دو آیتم پشت سر هم بهتر است به صورت 23 و 24 نوشته نشود، بهتر است به جای عدد 24 از عدد 28 استفاده شود تا در بین آن ها بتوان 4 آیتم را قرار داد.

برآورد

در هر اسپرینت تیم باید تصمیم بگیرد که چه آیتم هایی را گزینش کند. برای این کار لازم است زمانی که هر آیتم به خود اختصاص می دهد به دقت پیش بینی شود؛ به این عمل، برآورد آیتم ها بک لاگ گفته می شود. تیم توسعه مسئول برآورد آیتم ها، در جلسه برنامه ریزی اسپرینت می باشد. این عمل با حضور و نظارت مالک محصول انجام می شود. اما دخالت در برآورد آیتم ها از حوزه اختیارات مالک محصول خارج است. مالک محصول فقط وظیفه دارد آیتم های بک لاگ محصول را به ترتیب اولویت آماده سازد. تیم هر بار بخشی از آیتم های با اولویت را برای برآورد انتخاب می کند.

واحد برآورد آیتم های Strory Point است. Strory Point معادل عبارت "نفر-ساعت" می باشد که در بسیاری از صنایع استفاده می شود. نفر-ساعت به این معنی است که یک کار توسط چند نفر و چند ساعته انجام خواهد شد. در اسکرام این واحد معمولاً به صورت نفر-روز استفاده می شود. به عنوان مثال اگر آیتمی در دو روز با چهار توسعه دهنده تکمیل می شود میزان Strory Point این آیتم برابر است با:

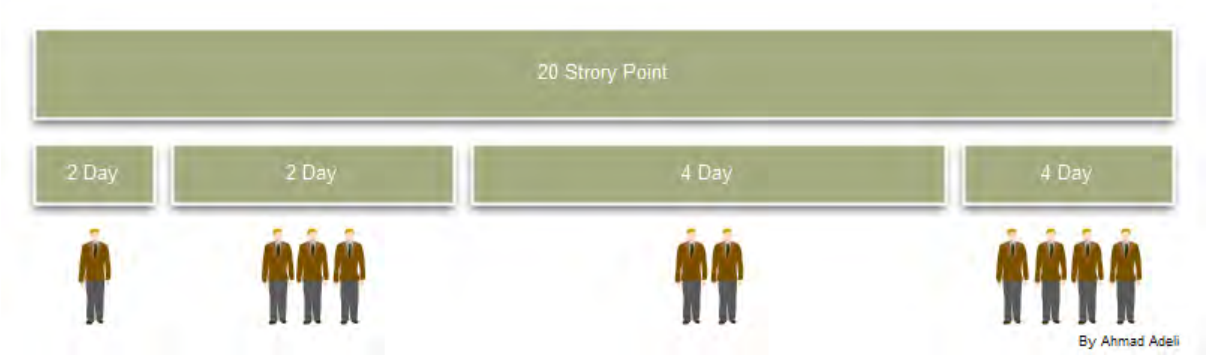
2 * 4 = 8

با توجه به رابطه مستقیمی که نیرو و روز با یکدیگر دارند(رابطه مستقیم به علت عمل ضرب می باشد)، با فرض بر این که طرف راست معادله ثابت باشد، با کاهش یکی از اعداد دیگر باید افزایش داشته باشد. پس اگر آیتم مثال فوق با دو نفر انجام شود قاعدتاً به علت نصف شدن تعداد توسعه دهندگان، تعداد روزهای توسعه باید دو برابر شوند:

4 * 2 = 8

Strory Point به این دلیل انعطاف پذیری بسیار زیاد در اسکرام استفاده می شود. در طول توسعه و اجرای اسپرینت ها نمی توان از همه چیز به صورت 100% یقین حاصل نمود. ممکن است در طول توسعه به طور موقت افرادی کناره گیری کنند(مثلاً به دلیل بیماری). در این صورت باید برای جبران آن سایر افراد توسعه باید وظایف شخص مذکور را پوشش دهند. یا حتی ممکن است یک آیتم در تعداد روز از پیش تعیین شده تکمیل نشود، در این صورت با افزایش تعداد نیرو می توان این نقصان را جبران نمود.

برآورد آیتم ها به تنهایی کاربردی ندارد. حجم یک اسپرینت نیز بر اساس واحد Strory Point باید تعیین شود. این تعیین حجم معمولاً توسط تیم و زیر نظر اسکرام مستر انجام می شود. اعضای تیم باید پیش از اجرای اسپرینت توافق کنند که چند Strory Point را می توانند در یک اسپرینت پوشش دهند(شکل 14-3). به عنوان مثال اگر اسپرینت 2 هفته ای دوازده روز کاری وجود داشته باشد و 10 توسعه دهنده در آن کار کنند. با ضرب این دو عدد در یکدیگر 120 = 10 * 12 مقدار Strory Point ی که تیم می تواند در اسپرینت انجام دهد بدست می آید. به این معنی که تیم در اسپرینت مذکور می تواند 120 Strory Point را به انجام برساند. البته لازم به ذکر است عدد فوق یک مثال می باشد. برای برنامه ریزی اسپرینت هیچگاه تعداد آیتم های اسپرینت به کار نمی رود بلکه از واحد Strory Point به جای آن استفاده می شود.



شکل 14-3: مدل مفهومی Story Point

تکنیک های برآورد اسپرینت

پس از تعیین حجم اسپرینت تیم، آیتم های با اولویت را یک به یک برآورد کرده و به بک لاگ اسپرینت اضافه می کند. این عمل تا جایی صورت می پذیرد که برآورد مجموع آیتم های بک لاگ اسپرینت به حجم از پیش تعیین شده اسپرینت برسد. عدد برآورد شده هر آیتم معمولاً در طول اسپرینت تغییر نمی کند. عدد برآورد شده کرانه بالای تأخیر در تکمیل یک آیتم را بیان می کند. به این معنی که نباید از این عدد عبور شود چرا که ممکن است موجب از دست رفتن زمان و برنامه ریزی شود. برآورد آیتم ها کار آسانی نیست و به مهارت و تجربه زیاد نیاز دارد.

شرایط تطبیق

این بخش که به آن شرایط پذیرش نیز گفته می شود در حقیقت نوع ساده ای از تست کیس ها می باشد. در بخش شرایط تطبیق به طور مشروح تعیین می شود که آیتم های تکمیل شده، دقیقاً چگونه باید باشند؛ به عبارت دیگر شرایطی است که باید توسط قابلیت ایجاد شده ارضا شود تا مورد پذیرش قرار گیرد. شرایط تطبیق عملاً راهنمایی برای توسعه دهندگان می باشد و به آنها نشان می دهند که یک آیتم چگونه باید ساخته شود. این شرایط در واقع جزئیات(Details) بیشتر را از قابلیت بالقوه آیتم در دست ساخت آشکار خواهد کرد. همانند تست کیس ها وظیفه نوشتن شرایط تطبیق نیز بر عهده مالک محصول می باشد. در بک لاگ محصول آیتمی برای توسعه کاندید می شود که شرایط تطبیق برای آن آماده شده باشد. شرایط تطبیق باید شفاف و بدون ابهام ارائه شود تا از برداشت اشتباه پیشگیری به عمل آید. شرایط تطبیق را معمولاً به صورت مشروح و خلاصه نوشته می شود اما تست کیس ها پیچیده تر هستند و به صورت گام به گام نوشته خواهد شد.

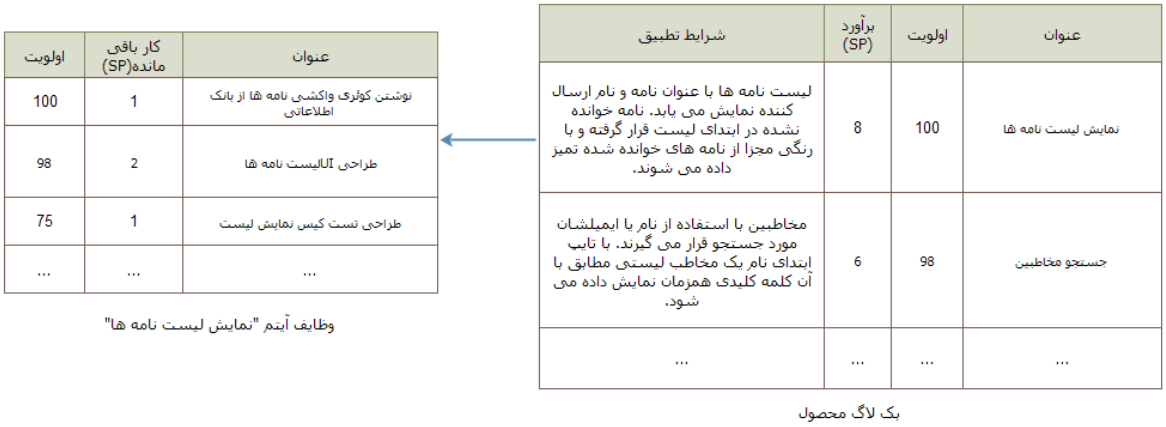
بخش هایی که توضیح داده شد قسمت های اجباری یک آیتم بک لاگ هستند. قسمت های دیگری نیز وجود دارد که اختیاری هستند. وجود آن ها می تواند در توسعه کمک کننده باشد. این بخش های اختیاری بخش دوم کتاب به تفصیل توضیح داده خواهند شد. به ازای هر گروه و هر مالک محصول، یک بک باگ محصول وجود دارد. اگر اسکرام در سطح کلان(چند تیمی) اجرا شود بدیهی است که چند بک لاگ محصول وجود نخواهد داشت. هر مالک محصول بک لاگی تحت اختیار خود و تیم خود دارد که برای توسعه از آن استفاده می کند. اما در سطح کلان نیز سندی شبیه به بک لاگ محصول وجود دارد که تحت اختیار مدیران پروژه می باشد. در این سند اهداف پروژه به صورت کلی و استراتژیک نوشته می شود. آیتم های سند بسیار کلی مطرح می شوند به عنوان مثال در پروژه سرویس دهنده ایمیل مدیران قابلیت ها و هدف هایی کلی چون پشتیبانی از نوع خاصی از تبلیغات آنلاین، ارتباط با دیگر سرویس های مشابه، توسعه برنامه هایی برای نوع خاصی از سیستم عامل های موبایل و... که انتظار دارند در Release های بعدی توسعه یابد در سند وارد می سازند این آیتم ها زمان بندی و اولویت بندی می شوند. برای شروع هر Release هر کدام از مالکین محصول طی جلسه ای با مدیران یکی از این آیتم های کلی را برای Release جاری انتخاب می نمایند. انتخاب این آیتم ها توافقی و بر اساس توانایی تیم ها انجام می شود. آیتم های چنین سندی آنقدر بزرگ کلی و بزرگ هستند که می توان از آنها به عنوان یک پروژه مستقل یاد کرد. مانند آیتم های بک لاگ محصول آیتم های این سند کلان باید دارای شرایط تطبیق باشند تا تیم توسعه روی آنچه در نهایت تحویل خواهد داد شناخت داشته باشد. پس از انتخاب یک آیتم کلی توسط مالک

محصول، او شروع به تحقیق و جمع آوری نیازها از مشتری می نماید و بک لاگ محصول خود مرتبط با خود را برای Release آماده سازد.

در بسیاری از پروژه ها، بک لاگ محصول به عنوان سند قرارداد یا ضمیمه ای از آن محسوب می شود. بک لاگ دقیقاً شرح می دهد که مشتری چه چیزی از توسعه دهندگان در پایان تحویل خواهد گرفت. بک لاگ محصول به دلیل مشروح و صریح بودن شیوه مستند سازیش به عقیده بسیاری بهترین گزینه برای محتوای قرارداد محسوب می شود. چرا که مشتری پس از تکمیل نرم افزار می تواند در صورتی که بخشی از قابلیت ها کامل نشده باشند بر اساس همان سند توافقی طلب خسارت کند. از طرف دیگر در صورتی که مشتری انتظاراتی بیش از حد آنچه ابتدا درخواست کرده بود داشته باشد تیم می تواند به تکمیلِ صرف آیتم های موجود در بک لاگ استناد نماید.

وظیفه ها

همانطور که گفته شد آیتم های بک لاگ از دیدگاه مشتری نوشته می شوند و از رویکردی تجاری برخوردار هستند. آیتم های بک لاگ محصول کلی تر از آن هستند که برای عملیات فنی مفید واقع شوند. در جلسه اسپرینت تیم معمولاً به شکستن آیتم های بک لاگ به وظایفی کوچک تر می پردازند. در این عمل سعی می شود وظایف تا حد عملیات فنی شکسته شوند. به عبارت دیگر تیم در این بخش تعیین می کند که چه کارهایی باید برای پیاده سازی یک آیتم انجام شوند(شکل 15-3).



شکل 15-3: مثال تعیین وظایف برای یک آیتم بک لاگ

وظایف می توانند کارهایی چون طراحی پایگاه داده، طراحی UI، آماده سازی تست های واحد و ... باشد. هر وظیفه معمولاً توسط یک توسعه دهنده انجام خواهد شد و معمولاً آنقدر کوچک هستند که در طول یک روز کاری قابل انجام هستند. بنابراین برای برآورد وظایف از واحد SP (و یا ساعت) استفاده می شود؛ به این دلیل که قادر هستد توسط یک نفر در طول یک روز انجام شوند. تا جای ممکن باید همه وظایف را برای آیتم ها پیش بینی کرد. بهتر است پیش بینی کمی بدبینانه باشد تا در صورت وجود کارهای پیش بینی نشده زمان بندی با مشکل مواجهه نشود. در عمل خرد کردن آیتم ها به وظایف نباید به صورت افراطی عمل شود. نیازی به تشریح دقیق آنچه که پیاده سازی می شود وجود ندارد. خاطر نشان می شود که مستند سازی زیاد در بیانیه آجیل نهی شده است و ارتباط نزدیک مشتری با مالک محصول می تواند جایگزینی مناسب برای آن محسوب شود. در ادامه سه بخش اصلی بدنه وظایف بررسی می شود.

عنوان

این بخش نیز مانند آیتم های بک لاگ محصول نامی را برای آیتم اختصاص می دهد. به دلیل این که برای آیتم های مختلف برخی کارها باید تکرار شود، بهتر است نامی برای یک وظیفه انتخاب شود که ابهام نداشته باشد. مثلاً ممکن است در همه آیتم های بک لاگ وظیفه طراحی UI وجود داشته باشد، باید عناوین این وظایف ارتباط خود را به نحوی با آیتم های بک لاگ نشان دهند.

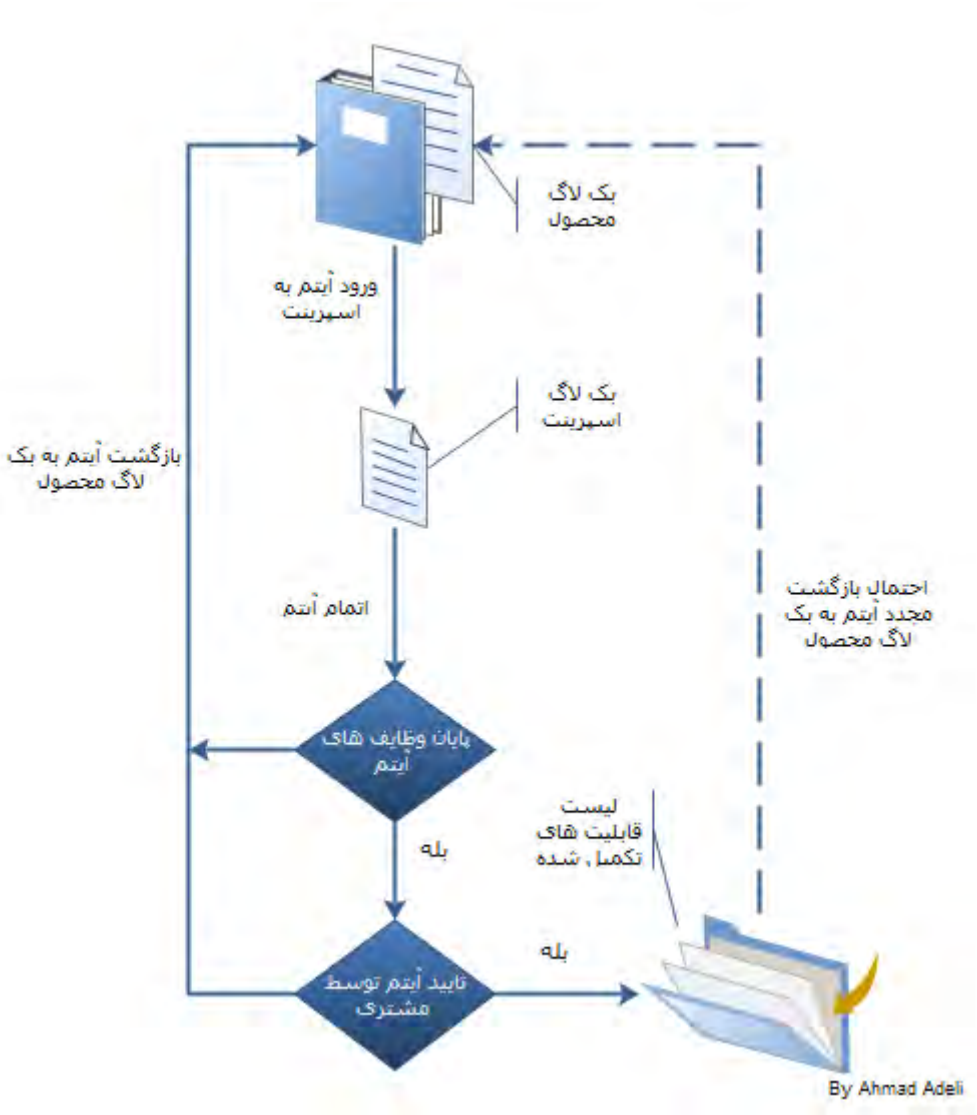
کار باقی مانده

این بخش مقدار کار باقی مانده برای هر آیتم را نشان می دهد که واحد آن همانطور که پیش از این گفته شده *ساعت* می باشد. پس از پایان روز کاری این بخش از وظایف به روز می شود. ممکن است یک وظیفه در بیش از یک روز انجام شود، به این ترتیب هر روز از میزان ساعت باقی مانده آن کاسته شده تا در نهایت تکمیل شود.

اولویت در بک لاگ اسپرینت

توسعه دهندگان معمولاً برای انجام وظایف آن ها را در یک لیست کلی قرار می دهند و به ترتیب شروع به تکمیل کردن آن ها می کنند. اغلب این گونه نیست که وظایف به تفکیک آیتم ها انجام شوند؛ لبکه وظایف آیتم های به صورت موازی انجام می شوند. به عنوان مثال ممکن است همزمان یک توسعه دهنده بخش از کد نویسی کند و توسعه دهنده دیگری ری تست واحد آیتم دیگری کار کند. پس از استخراج وظایف کامل آیتم ها، تیم آنها را به صورت یک لیست وظایف اسپرینت نگه داری می کند. بدیهی است که برخی از وظایف بر دیگری اولویت دارند. با نسبت دادن عددی(مانند آنچه در بخش اولویت بندی آیتم های بک لاگ گفته شد) می توان این اولویت بندی را اعمال نمود. لازم به ذکر است این گونه اولویت بندی از اولویت بندی آیتم های بک لاگ مجزا است. اولویت بندی آیتم های بک لاگ در زمان برنامه ریزی برای گزینش آیتم ها مورد استفاده قرار می گیرد، اما اولویت بندی وظایف برای ترتیب دادن به عملیات توسعه به کار می رود.

مفهومی در اسرام وجود دارد به نام *تکمیل شده* یا *Done*. این مفهوم به این موضوع اشاره می کند که هنگامی به یک آیتم تکمیل شده تلقی می شود که اولاً همه وظایف آن به پایان رسیده باشد و ثانیاً توسط مالک محصول و سپس مشتری تأیید شده باشد(معمولاً این عمل در جلسه بازبینی انجام می شود). اگر وظایف یک آیتم کامل شود اما آیتم مذکور توسط مشتری پذیرفته نشود این آیتم به عنوان یک آیتم تکمیل شده محسوب نمی شود و صفت Done به آن تعلق نمی گیرد. به عنوان مثال فرض کنید رابط کاربریِ آیتمی مورد پسند و پذیرش مشتری قرار نگرفته باشد. چنین آیتمی مجدداً به لیست بک لاگ محصول بازگردانده می شود، اما این بار تنها یک وظیفه انجام نشده دارد و آن هم مربوط به طراحی UI آن می شود. از این رو وظایفی که تکمیل می شود به طور کامل حذف نمی شوند و فقط به طور موقت از لیست وظایف اسپرنیت خارج می شوند تا بتوان در صورت لزوم آنها را بازیابی نمود(آیتم ها حتی پس از تأیید مشتری، هرگز به صورت دائم به حالت Done نمی روند)(شکل 16-3).

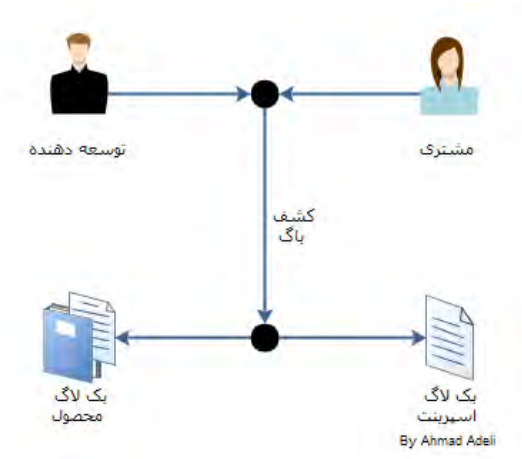


شکل 16-3: چرخه عمر وظایف

باگ

باگ ها خطاها و اشکالات نرم افزاری هستند که در طول توسعه و یا پس از آن توسعه کشف می شوند. بدیهی است که چنین خطاهایی به سرعت باید بر طرف شوند. در اسکرام برای بر طرف سازی باگ های کشف شده برنامه ریزی می شود. بدین وسیله زمانی که قرار است به تعمیر عیب نرم فزار اختصاص داده شود خللی ایجاد در زمان بندی ایجاد نمی کند، چرا که گاهی پیدا کردن و بر طرف سازی یک نقص می تواند بیش از آیتم های اصلی یک بک لاگ زمان بر باشد. برنامه ریزی باگ همانند برنامه ریزی آیتم های بک لاگ است. عملاً در اسکرام با یک باگ به عنوان یک آیتم بک لاگ برخورد می شود.

هنگامی که یک باگ توسط توسعه دهنده ای کشف می شود و یا به نحوی از سوی مشتری گزارش می شود، تیم به باگ کشف شده یک اولویت می دهد و آن را در بک لاگ وارد می سازد(شکل 17-3). به این ترتیب باگ در زمان مناسب خود (بر اساس اولویت آن) از بک لاگ استخراج شده و روی آن کار می شود.



شکل 17-3: مراحل کشف باگ

اما سوال اینجا است که باید وارد کدام بک لاگ شود، بک لاگ محصول یا بک لاگ اسپرنیت؟ پاسخ این سوال به خود تیم باز می گردد برخی از تیم ها دارای زمان بندی فشرده ای هستند به این علت هیچ آیتم جدیدی را در اسپرینت جاری نمی پذیرد و آن را در بک لاگ محصول وارد می کند تا در اسپرینت های دیگر به آن رسیدگی شود. از طرفی ممکن است بک لاگ به یک مانع در روند اجرای توسعه بدل شود و بر طرف سازی آن اولویت بالایی داشته باشد. در چنین حالتی تیم مجبور است این آیتم(باگ) را در بک لاگ اسپرینت جاری وارد کند و در همین اسپرینت آن را رفع نماید.

با وجود اینکه در عمل هر باگ یک آیتم بک لاگ می باشد اما دو تفاوت بین آنها وجود دارد؛ اول اینکه باگ ها دارای یک بخش اضافی به نام "چگونگی کشف" هستند. هر توسعه دهنده ای که باگ را می یابد باید تعیین کند که در کجا و با چه رفتاری با آن روبرو شده است. یافته ها از طریق بخش "چگونگی کشف" با دیگر توسعه دهندگان به اشتراک گذاشته می شود تا در زمان های دیگر سایر توسعه دهندگان بتوانند مجدداً وقوع باگ را بررسی نمایند. بخش متفاوت دوم شرایط تطبیق است. در این بخش مطرح می شود که چه شرایطی باید ارضا شوند تا باگ رفع شده تلقی گردد. به عبارت دیگر اگر اعمال درون شرایط تطبیق انجام شوند نباید مجدداً باگ ظاهر شوند. با استفاده از این دو بخش توسعه دهنده ای که مسئولیت رفع این مشکل را برعهده می گیرد (معمولاً کسی است که توسعه بخشی از نرم افزار را که باگ در آن شکل گرفته برعهده داشته است) در مسئله شناخت حاصل می کند و به راحتی به رفع عیب می پردازد(شکل 18-3).

عنوان	اولویت	برآورد (SP)	چگونگی کشف	شرایط تطبیق
عدم انتخاب نامه برای حذف	86	2	با انتخاب نامه با وجود علامت خوردن آن انتخابی صورت نگرفته و عمل حذف حتی پس از به روز رسانی، انجام نمی شود.	انتخاب نامه با نمایش ظاهری آن همراه است. با کلیک روی دکمه حذف نامه های انتخاب شده پس به روز رسانی صفحه حذف می شوند.
عدم اعمال پارامتر نام مخاطب در جستجو مخاطبین	79	6	در صورت استفاده از نام مخاطب برای جستجو، با وجود مخاطب مورد نظر هیچ نتیجه ای حاصل نمی شود.	با انتخاب پارامتر نام مخاطب و وارد کردن متن مورد نظر در فیلد مربوطه جستجویی بر اساس آن در نام مخاطبین صورت گرفته و نام هایی را که مورد تطبیق قرار گرفته باشد بر گشت داده می شوند.
...	...	...		...

شکل 18-3: مثال کشف باگ برای پروژه سرویس دهنده ایمیل

در بخش وظایف یک آیتم باگ باید مانند آیتم بک لاگ محصول جزئیات فنی و مراحل رفع عیب ارائه می شود. بدیهی است در صورتی بخش وظایف تکمیل می شود(Done شده فرض می شود) که راه حلی برای بر طرف

کردن مشکل پیدا شود. ممکن است پس از یافتن راه حل بخشی از وظایف پیش بینی شده انجام نشوند و یا حتی به وظایف دیگر نیاز باشد. از همین رو برخی از توسعه دهندگان (به خصوص هنگامی که باگ اولویت بسیار بالایی دارد و سریعاً باید بر طرف شود) بخش وظایف را تا یافتن یک راه حل قطعی مستند نمی کنند.

مهم است که در یافتن و اضافه کردن باگ به بک لاگ افراط نشود. به عبارت دیگر باید سعی شود هر مشکلی کوچکی را به عنوان یک باگ مطرح نکرد چرا که ممکن است زمانی برای برنامه ریزی آن مصرف می شود از زمان حل خود باگ بیشتر باشد و سربار ایجاد کند.

مانع

مانع به هر چیزی گفته می شود که در فرآیند توسعه خلل ایجاد کند و یا آن را متوقف سازد. به عنوان مثال عدم وجود راه حل برای یک مسئله، عدم تخصص تیم در یک زمینه، عدم دسترسی به نرم افزار یا سخت افزاری خاص، عدم یافتن لاگ ها بالقوه، عدم داشتن الگوریتمی بهینه و در عدم آنچه برای پیشروی روند اجرای توسعه ضروری است، می تواند نمونه ای از این نوع موانع باشند. یک مانع بدترین مشکلی است که توسعه دهندگان می توانند در یک توسعه با آن برخورد کنند. یک مانع پیشرفت پروژه را کند یا متوقف می سازد و حتی ممکن است یک پروژه را با شکست روبرو کند. حل موانع ممکن است بسیار زمان بر باشد و بهتر است برای آنها برنامه ریزی انجام شود.

در هر زمان ممکن است بیش از یک مانع وجود داشته باشد. در این صورت بهتر است برای آنها اولویت تعیین شود تا به موانع اضطراری سریع تر رسیدگی شود. برای رسیدگی به موانع نمی توان مدت زمان خاصی را تعیین نمود و هر تیم باید خود به گونه ای در زمان های خالی یا با فدا کردن بخشی از زمان اسپرینت مانع را از راه توسعه کنار بزند. برخی تیم ها از پیش زمانی را برای وقوع چنین موانعی خالی می گذارند. این گونه تیم ها عموماً هنگام مواجهه با موانع زمان زیادی را از دست نمی دهند اما باید به این نکته توجه شود که احتمال وقوع موانع بسیار پایین است و خالی گذاشتن قسمتی از زمان توسعه برای آنها می تواند در دراز مدت (به خصوص اگر بر خلاف پیش بینی مانعی به وجود نیاید) اثری سوء در زمان بندی پروژه داشته باشد. از منظر اشتراک اطلاعات، بهتر است علاوه بر عنوانی که برای آیتمِ مانع، توضیحی نیز برای آن ثبت شود تا از ایجاد ابهام در آن پیشگیری به عمل آید.

تست کیس

در فصل پیش گفته شد که تست کیس سندی است که از طریق آن صحت توسعه آیتم های بک لاگ مورد آزمایش قرار می گیرد. این سند یه صورت گام هایی نوشته می شود که آیتم بالفعل (قابلیت تکمیل شده) باید بتواند همه آنها را به خوبی پاس کند. تست کیس ها شبیه بخش شرایط تطبیق در آیتم های بک لاگ هستند با این تفاوت که شرایط تطبیق به صورت یک متن مشروح است اما تست کیس همان متن را به صورت گام هایی مجزا تعریف می کند. اجرای گام به گام تست کیس به این دلیل است که معمولاً برای آزمایش یک آیتم (قابلیت) اعمال زیادی صورت می گیرد و اگر این قابلیت در آزمایش شکست بخورد با اجرای گام به گام دقیقاً مشخص می شود چه عملی باعث شکست تست شده است. از این گذشته با این عمل بدون نیاز به توضیحات مشروح دیگری نتایج تست مستند می شود.

هر گام از دو بخش تشکیل می شود، بخش اول عملیات تست می باشد بخشی است که از توسعه دهنده خواسته می شود که در راستای آزمایش آیتم چه عملی را انجام دهد. بخش دوم نتایج مورد انتظار است؛ در این بخش نتیجه درستی هر عملی که در بخش اول قرار می گیرد نوشته می شود. اگر نتیجه مورد نظر با نتیجه ای که کاربر از عمل مربوطه بدست می آورد مغایر باشد تست شکست می خورد. به ای ترتیب دقیقاً مشخص می شود که با چه عملی مشکل ایجاد شده است. گاهی ممکن است لازم باشد داده ای در عملی که انجام می شود استفاده شود در نتیجه بهتر است برای تست این گونه قابلیت ها داده ای انتخاب شود و در تست کیس قرار گیرد. مطلوب است که در صورت توان فردی که تست کیس را می نویسد بدترین حالت برای داده ها انتخاب کند یا گام مورد نظر را با مجموعه ای از داده ها آزمایش نماید(شکل 19-3).

حذف نامه ها در لیست نمایش		
عمل	نتایج مورد انتظار	وضعیت
انتخاب نامه	این عمل باعث علامت خوردن نامه در لیست نمایش و ظاهر شدن دکمه حذف می شود.	پاس
کلیک روی دکمه حذف	اعلام پیام تاییدیه حذف	پاس
تایید حذف از سوی کاربر	حذف نامه و به روز رسانی صفحه	رد

شکل 19-3: مرحله(Step)های یک تست کیس

تست کیس ها گاهاً به عنوان یک آیتم بک لاگ محسوب می شوند. آنها پس از اولویت بندی باید درون بک لاگ قرار گیرند. بر خلاف باگ ها و آیتم های بک لاگ تست کیس ها زمان خاصی را از توسعه نمی گیرند بنابراین در زمان بندی چندان لحاظ نخواهند شد. معمولاً به گونه ای اولویت بندی می شوند که با کامل شدن آیتم ها بک لاگ همزمان باشند. برای هر آیتم بک لاگ حداقل یک تست کیس نوشته می شود. نوشتن تست کیس برای آیتم های بک لاگ امری ضروری است و بهتر است برای باگ های پیچیده نیز تست کی ایجاد شود و صرفاً به شرایط تطبیق بسنده نشود. لازم به ذکر است که تست کیس ها بر اساس شرایط تطبیق آیتم ها و توسط مالک محصول نوشته می شوند.

نوعی تست کیس وجود دارد که به کمک ابزارهای CASE به صورت خودکار اجرا می شود. ابزار CASE عملیات آزمونگر را برای گام هایی که پاس می شود ذخیره می نماید. این ذخیره سازی تا رسیدن به یک گام شکست خورده یا پاس شدن همه گام ها ادامه می یابد. حال هر بار نیاز به اجرای تست باشد کل عملیات تا رسیدن به گام شکت خورده به صورت خودکار انجام می شود و پس از رسیدن به گام مذکور اجرای تست متوقف شده و کنترل به دست آزمونگر باز می گردد. این حالت برای تست های طویل و پر زحمت بسیار مناسب است و از نظر زمانی پایین ترین هزینه را در بر دارد. برخی از ابزارها، داده های ورودی را نیز در خودکار سازی پشتیبانی می کنند. در این حالت نیز آزمونگر می تواند بدون درگیر شدن با ورود داده ها، تست ها را به کرات اجرا کند.

یکی از اشتباهاتی که پس از اجرای مکرر تست های شایع است این است که آزمونگر با فرض این که گام های پاس شده سابق بر این تست شده اند به تست مجدد آنها در دفعات دیگر اقدام نمی کند. اما این حالت می تواند مشکلات بالقوه را پنهان سازد چرا که رفع اشکال یک آیتم برای گام های شکست خورده ممکن است دیگر بخش های نرم افزار(کام های پاس شده) را به حالتی ناسازگار ببرد.

ناگفته نماند شکست در تست ها اغلب به دلیل کشف یک باگ است. اگر باگ به سرعت بر طرف نشود و نیاز به کار زیادتری داشته باشد منجر به اضافه کردن یک آیتم باگ در توسعه خواهد شد. به بیان دیگر یک تست کیس را می توان نوعی رابط بین آیتم و باگ های احتمالی درون آن تلقی نمود.

علاوه بر آیتم هایی که معرفی شدند سه آیتم نیز وجود دارند که دارای حالتی گزارشی هستند و اغلب برای ردگیری(Tracking) در توسعه مورد استفاده قرار می گیرند.

نمودار Sprint Burn Down

نمودار Sprint Burn Down یکی از پرکاربردترین آیتم ها مورد استفاده در اسکرام است. از طریق این نمودار تیم توسعه می تواند روند توسعه را به طور منظم مرور نماید. در اسکرام برای هر اسپرینت نموداری جداگانه رسم می شود(این نمودار در فصل پیش معرفی شده است)(شکل 4-3). این نمودار برای تیم توسعه چهار شناخت را ایجاد می کند که عبارت اند از:

- کارهای تکمیل شده
- کارهای باقی مانده

- رفتار تیم در توسعه آیتم
- رویکرد آتی اسپرینت

اگر خط واقعیبه بالای خط ایده آل برسد این احتمال وجود دارد که تیم مطابق زمان بندی نتواند Storty Point ها را تکمیل کند. برای این حالت باید سریعاً راه حلی ارائه شود. یکی از دلایل روی دادن این حالت می تواند کم کاری اعضای تیم باشد. اسگرام مستر باید دلیل ضعف عملکرد را در تیم بیابد و در صدد رفع آن برآید. یکی دیگر از دلایل این حالت که بسیار معمول می باشد برآورد اشتباه تیم از آیتم ها یا حجم اسپرینت می باشد.

برای حالتی که خط واقعی بالاتر از خط ایده آل است دو راه حل موجود است: یا باید با اطلاع مشتری از چنین وضعی مقداری از بار اسپرینت را سبک نمود یا کار بیشتری انجام داده شود تا کبودها جبران گردند. موانع نیز خود می تواند دلیلی برای این اتفاق باشد. حل موانع زمان گیر است و ممکن است باعث شود در چند روز کاری تیم نتواند بر روی هیچ آیتمی کار کند. امکان دارد با همه تلاش های تیم باز هم آخرین نقطه این خط در پایان با آخرین نقطه خط ایده آل برابر نشود(آیتم ها به صورت کامل تکمیل نشوند) در این حالت تیم باید همه کارهای تکمیل نشده را برای کار بیشتر به اسپرینت بعدی منتقل نماید.

علاوه بر حالت پیش ممکن است حالتی پیش بیاید که خط واقعی پایین تر از خط ایده آل قرار بگیرد. این حالت تنها و تنها یک دلیل می تواند داشته باشد و آن هم اشتباه در برآورد آیتم ها است. اغلب چنین حالتی خبر خوشی برای تیم می باشد؛ چرا که مطمئن خواهند بود که از زمان بندی عقب نخواهند افتاد. جلو زدن از زمان بندی نیز چندان حالت خوبی نمی تواند باشد. معمولاً در این حالت تیم با یک جلسه کوچک و اظطراری سعی می کند تا حد توان آیتم های اضافه تری را از بک لاگ محصول به بک لاگ اسپرینت منتقل کند تا اینکه هم آیتم های بیشتری تکمیل شوند و هم خط واقعی به خط ایده آل نزدیک شود.

ممکن است در برخی روزها علیرغم کار زیاد هیچ آیتمی تکمیل نشود. یکی از دلایل آن می تواند توسعه موازی باشد. پس از آن که آیتم ها به مراحل پایانی توسعه خود رسیدند این بار برعکس حالت قبل خط نمودار شدیداً سقوط می کند. وجود این حالت ها در اسکرام طبیعی است و تیم نباید تصمیم عجولانه ای در این رابطه بگیرد. توصیه می شود برای جلوگیری از چنین حالات ابهام انگیزی آیتم ها به صورت سریالی، یکی پس از دیگری توسعه یابند.

ایده آل این است که کل آیتم های اضافه شده در بک لاگ محصول در اسپرینت های از پیش برنامه ریزی شده تکمیل شوند. اما ممکن است گاهی به دلیل اضافه کردن برخی آیتم ها در بین Release (که عموماً اجتناب ناپذیر است) این امر محقق نشود. در این حالت یا مهلت زمانی Release افزایش یافته و یک اسپرینت اضافه می شود و یا تکمیل آیتم های باقی مانده به Release بعدی موکول می شود.

علاوه بر آیتم هایی که برای اسکرام شرح داده شدند، آیتم های کاری و مستندات دیگری نیز وجود دارند که برخی از تیم ها از آنها بهره می جویند. این آیتم ها نقش کمک کننده را در توسعه دارا هستند و استفاده از آنها برخلاف آیتم های شرح داده شده اجباری نیست. از این دست آیتم ها می توان به آیتم آمادگی تست کیس ها، روند اجرای تست اشاره کرد. پوشش همه این آیتم ها از حوصله این کتاب خارج است.

فصل پنجم: جلسه های اسکرام

اسکرام یک متولوژی جلسه محور است به جای استفاده از سندات و گزارشات و ارتباطات رسمی و غیر قابل انعطافی که در بیشتر سازمان ها وجود دارد اسکرام از جلسات برای ارتباطات درون گروهی و بین فردی استفاده می کند.

با وجود این نباید در برگزاری جلسات افراط شود. جلسات، آیتم های زمان بری هستند پس در برگزاری آنها باید دقت زیادی به خرج داد. جلسات باید برنامه ریزی شوند؛ به این معنی که باید تعیین شود جلسات در چه زمانی و با چه مدت زمانی برگزار شوند و همچنین در جلسه چه موضوعاتی مطرح شود. هر چه تعداد افرادحاضر در جلسه زیادتر باشد کنترل آن طبیعتاً مشکل تر خواهد شد. ازدیاد افراد موجود در توسعه می تواند کارایی جلسه را به خطر بیاندازد اما از طرفی بهتر است همه اعضا در جلسات شرکت داشته باشند تا بتوانند راجع موضوعات تحت اختیار خود در تصمیم گیری گروهی شرکت داشته باشند.

همانطور که پیش از این گفته شد برنامه ریزی و برگزاری جلسات کارا اغلب بر عهده اسکرام مستر می باشد. او باید تضمین کند که همه جلسات به نحو احسن برگزار خواهد شد. تکرار برخی جلسات معمولاً موجب آزردگی افراد می شود پس بهتر است زمان آن در حد معمول باشد و سبک و سیاق آن نیز متناسب با ذائقه افراد تعیین شود. در اسکرام پنج نوع جلسه وجود دارد که هر کدام در طول توسعه تکرار خواهند شد. این پنج نوع جلسه عبارت اند از جلسه برنامه ریزی Release، جلسه برنامه ریزی اسپرینت، جلسه برنامه ریزی اسپرینت، جلسه تحویل قابلیت و جلسه بازبینی که در ادامه این فصل به تفصیل شرح داده خواهد شد(شکل 9-2).

جلسه برنامه ریزی Release

در این جلسه همانطور که از نام آن پیدا است به مسائل مرتبط با یک Release پرداخته می شود. این جلسه چندان بین تیم های اسکرام مرسوم نیست و ممکن است برای برخی گروه ها اجرا نشود. این جلسه با حضور نمایندگان مشتری، مالک محصول و اسکرام مستر (به عنوان نماینده تیم) انجام می شود. حضور سایر اعضای تیم (توسعه دهندگان) در این جلسه الزامی نیست. در جلسه برنامه ریزی Release بر سر آنچه در انتهای Release به عنوان یک نرم افزار کارکننده تحویل داده می شود، بحث می شود.

تاریخ انتهای Release را معمولاً مشتری پیشنهاد می کند چرا که دریافت به موقع و زمان بندی شده نرم افزار برای مشتری اهمیت بسیاری دارد. پس از آنکه نظرات اسکرام مستر و مالک محصول ارائه شد سعی می شود بر سر یک تاریخ مشخص توافق شود.

پس از تعیین تاریخ پایان Release بر سر آنچه در پایان باید تحویل داده شود مذاکره می گردد. قابلیت ها و ویژگی هایی که مطرح می شود در قالب انتظاراتی که مشتری از نرم افزار دارد بیان می شود. این انتظارات به صورت کلی و استراتژیک مطرح می شود و جزئیات چندانی رونمایی نمی شود. مالک محصول بسیار راغب است که این بخش تا حد ممکن شفاف سازی شود تا از ابهام و بروز مشکلات آنی در ادامه توسعه پیشگیری به عمل آید.

گاهی ممکن است مشتری در انتظاراتش با توجه به بازه زمانی محدود کمی افراط کند. مالک محصول که به خوبی می داند که با عدم تحویل آنچه در این جلسه روی آن توافق می شود، او است که باید پاسخ گو باشد سعی می کند به کمک اسکرام مستر که نماینده توسعه دهندگان می باشد، انتظارات مشتری را به سطحی قابل اجرا تنزل دهد. در این بین حتی ممکن است بر سر آنچه باید در نهایت توافق شود کشمکش هایی نیز بین دو طرف صورت پذیرد. بهترین راه حل چنین شرایطی شفاف سازی و ارائه اطلاعات فنی به مشتری است.

علاوه بر دو مورد ذکر شده، در این جلسه درباره تعداد اسپرینت ها، حجم احتمالی اسپرینت ها، مسائل کلی توسعه پیش بینی هایی صورت می گیرد. این پیش بینی ها معمولاً ضمانت اجرایی نخواهند داشت و بیشتر برای تسهیل در برنامه ریزی انجام خواهد شد.

جلسه برنامه ریزی اسپرینت

این جلسه پیش از اجرای هر اسپرینتی باید انجام پذیرد. اعضای این جلسه کل اعضای تیم اسکرام یعنی مالک محصول، اسکرام مستر و همه توسعه دهندگان می باشد. در این جلسه به دقت تعیین می شود در اسپرینت چه باید کرد و چه در انتهای تحویل داده خواهند شد(شکل 10-2).

پیش از اجرای این جلسه مالک محصول باید لیست آیتم های بک لاگ محصول را تا جای ممکن تکمیل کرده باشد. کامل سازی کل لیست به خصوص در ابتدای Release امری دشوار و گاهی غیر ممکن است. از آن جایی

که لیست بک لاگ محصول یک لیست اولویت دار است، مالک محصول باید حداقل تعدادی از آیتم های ابتدایی را که پیش بینی می کند در اسپرینت جاری گزینش خواهند شد؛ آماده کند. منظور از آماده سازی این است که آیتم کاملاً شناسایی شده باشد و همه بخش های آن به خصوص شرایط تطبیق کامل و قابل ارائه باشد. مالک محصول اصولاً باید از ارائه آیتمی که برای او و یا مشتری ابهام دارد خودداری نماید. او می تواند اولویت چنین آیتمی را پایین آورده تا از دست رس توسعه دهندگان موقتاً دور بماند.

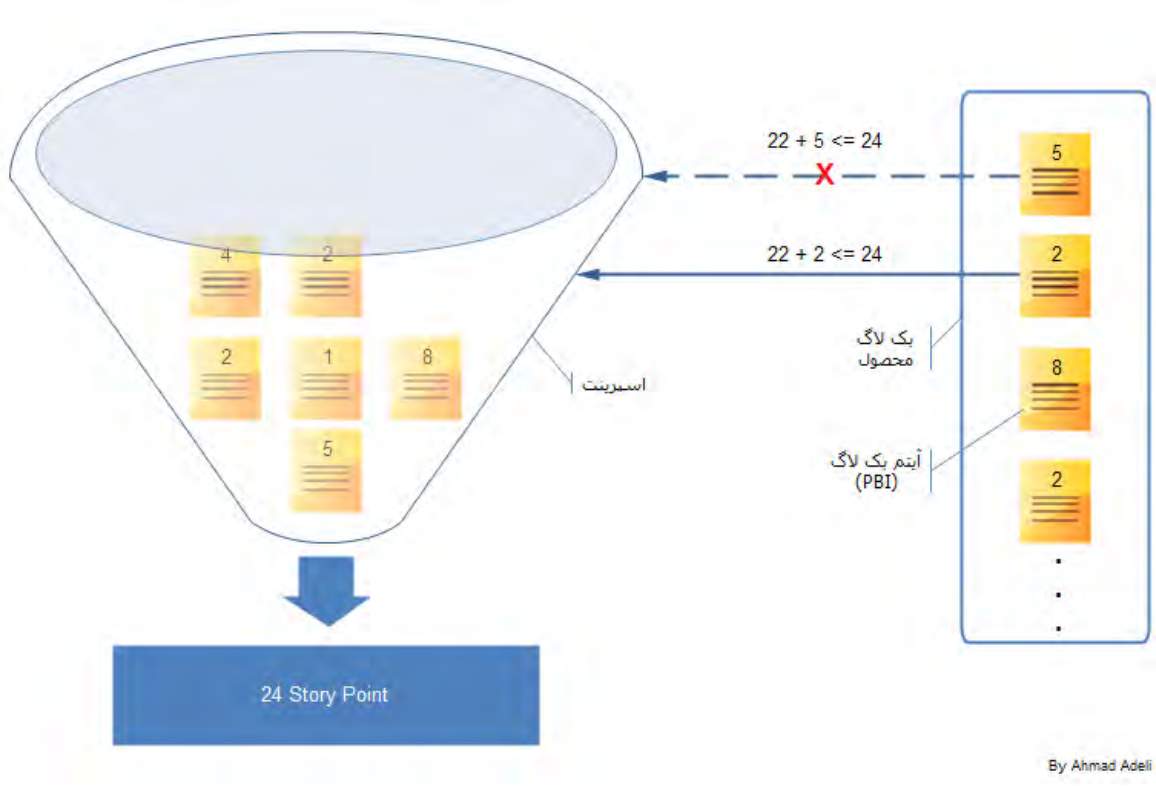
تیم توسعه و به ویژه اسکرام مستر نیز باید تخمین خود را از حجم اسپرینت پیش از اجرای جلسه انجام دهند. حجم اسپرینت پس از موافقت مالک محصول نهاییتاً معین خواهد شد. اعضای تیم توسعه هر کدام باید به شخصه کل روزهای حضور خود را در اسپرینت پیش بینی کرده و اعلام نمایند. پس از آن اعضای جلسه باید نسبت به افزودن توسعه دهنده کمکی یا حتی افزایش ساعت کار توسعه دهندگان مبنی بر پیش بینی اعلام حضور اعضای تیم اسکرام تصمیم گیری نمایند. بدیهی است که تصمیم گیری باید قبل از تخمین حجم اسپرینت انجام شود چرا که اعضای حاضر در تیم در طول اسپرینت یکی از فاکتورهای تخمین حجم اسپرینت می باشد. توصیه شده که اعضای تیم برای اسپرینتی که آیتم های درون آن دانش فنی خاصی را طلب می کند آمادگی لازم را داشته باشند. این آمادگی باعث می شود تیم با اطمینان بیشتری آیتم های فنی تر را برگزیند.

اسکرام مستر نیز به نوبه خود باید پیش از اجرای جلسه برنامه ریزی اسپرنیت کاستی های تجهیزات و امکانات و موانع احتمالی سد راه توسعه را بررسی کند. این گونه مسائل باید در این جلسه مطرح شده تا دید اعضا نسبت به برنامه ریزی واقعی تری باز کند. اسکرام مستر همچنین باید از آگاهی داشتن همه اعضا از زمان و البته مدت زمان جلسه اطمینان حاصل کند.

با شروع جلسه ابتدا مالک محصول به ارائه آیتم های بک لاگ محصول می پردازد. او سعی می کند از آنچه که او انتظار دارد در اسپرینت انجام شود سایرین اطلاع داشته باشند. مالک محصول از بالای لیست شروع به تشریح آیتم ها می کند. او برای تیم توضیح می دهد که هر آیتم چیست؟ چه اولویتی دارد؟ از نظر تجاری دارای چه ارزشی می باشد؟ و مهمتر از همه باید چگونه باشد و با چه شرایطی قابل پذیرش است؟ پس از آن که آیتم ها به صورت کامل تشریح شدند، اعضای تیم در صورت داشتن ابهام شروع به اخذ آگاهی از جانب مالک محصول می کنند. مالک محصول باید درباره هر آیتمی که ارائه می دهد اطلاعات کاملی داشته باشد، در غیر اینصورت تیم از وی می خواهد که یا آیتم حذف شود یا اولویت آن برای بررسی و کسب اطلاعات بیستر تنزل دهد.

اسکرام مستر نقش هدایت کننده را در این اشتراک اطلاعات بر عهده دارد. او نباید اجازه دهد تیم با طرح مسائل فنی زمان جلسه را به هدر دهد. همچنین باید مطمئن شود که مالک محصول از زیر بار پاسخ به سوالات و ابهامات شانه خالی نمی کند. اسکرام مستر با طرح سوالاتی می توان اطمینان حاصل کند که تیم برداشت اشتباهی از توضیحات مالک محصول نداشته باشد و انتقال مفهوم کامل و صحیح انجام شده است.

پس از شناخت آیتم های کاندید تیم شروع به گزینش این آیتم ها می کند. هر آیتم توسط تیم توسعه باید برآورد شود. اعضای تیم در حضور مالک محصول به گفتگو راجع به این آیتم ها می پردازد و سعی می کند به برآورد و پیش بینی های درستی دست پیدا کنند. حضور یک عضو با تجربه به همراه اسکرام مستر می تواند به برآوردهایی بسیار واقعی منتج شود. اغلب اوقات در تیم هایی که تجربه کار با یکدیگر را دارند عمل برآورد به سادگی و پس از کمی مذاکره صورت خواهد پذیرفت. تیم باید از آیتم های با اولویت در لیست شروع کند چرا که به نظر مالک محصول این آیتم ها مهمتر می باشند و اسرع وقت باید تکمیل شوند. پس از برآورد هر آیتم که بر اساس Story Point می باشد تیم به حجم اسپرینت نگاه می کند اگر مجموع Story Point آیتم جاری و آیتم هایی که پیش از این انتخاب شده اند کمتر از حجم از پیش تعیین شده اسپرینت باشد آیتم انتخاب می شود. اما اگر این عدد از حجم اسپرینت بیشتر شود، این آیتم قابل پذیرش نمی باشد(شکل 1-5).



شکل 1-5: نحوه بستن بک لاگ محصول

با چنین حالتی مالک محصول از اعضای تیم می خواهد در برآوردشان تجدید نظر کنند چرا که برای پر کردن بک لاگ اسپرینت تیم با پس زدن این آیتم به سراغ آیتم بعدی در لیست خواهد رفت که ممکن است چندان به مذاق مالک محصول خوش نیاید. اگر اولویت های این آیتم و آیتم بعدی یکی باشد به شرط مناسب بودن آیتم بعدی مشکل با انتخاب آن حل خواهد شد اما اگر اولویت آن بسیار مهم باشد تیم از مالک محصول می خواهد که آیتم را به آیتم های کوچک تر بشکند تا قسمتی از آیتم را حداقل بتوان در اسپرینت انجام داد. شکستن آیتم در مواقعی دیگر مانند هنگامی که تیم در برآورد دچار مشکل می شود راه حل مناسبی است. گزینه دیگر برای جای دادن یک آیتم در بک لاگ اسپرنیت جا به جایی اولویت آیتم مذکور با آیتم های بالایی است. همچنین این مشکل با افزایش حجم اسپرینت نیز قابل حل می باشد. البته افزایش حجم احتمالاً با اعتراض اعضای تیم همراه می شود، چرا که برای آنها کار بیشتری را در پی خواهد داشت.

برای حل این مسئله می توان از شیوه تحویل مشروط استفاده کرد. بر این اساس تیم تحویل آیتم هایی را که بیش از حجم اسپرینت هستند به شرط وجود زمان خالی در انتهای اسپرینت می پذیرد. به عبارتی دیگر بر خلاف دیگر آیتم ها که تحویل آنها باید ضمانت شود برای تحویل آیتم های مشروط هیچ تضینی ارائه نمی شود اما تیم حداکثر سعی خود را برای تحویل این آیتم ها می کند. برآورد آیتم ها کار بسیار حساسی است. اما اگر این عمل گروهی انجام شود ریسک برآورد اشتباه در تیم به حداقل می رسد. برخی آیتم ها به توسعه دهندگان خاصی مربوط می شود؛ با این حال بهتر است سایر توسعه دهندگان نظرات خود را نیز بیان کنند تا دید بهتری از برآورد بدست آید. برخی برای برآورد زمان بیشتری نسبت به سایرین صرف می کنند، برخی نیز تاره کار هستند و ممکن است برآورد نه چندان درستی را ارائه کنند. همه این موارد در یک جلسه برنامه ریزی امری است طبیعی. در تیم هایی که در آنها ترکیبی از افراد ماهر و تازه کار وجود دارد ممکن است تصمیمات تیم حول محور افراد ماهر گرفته شود. به این علت که به دلیل اعتماد اعضا به توسعه دهنده مذکور هر آنچه او برای برآورد نظر دهد با همسو شدن نظر سایرین با او همراه می شود.

برای جلوگیری از چنین وضعی بهتر است افراد ماهر نظرات خود ر پس از همه ارائه کنند یا نشرات به صورت غیر شفاهی اعلام شوند. اسکرام مستر با نظارت بر برآورد آیتم های تیم باید اطمینان حاصل کند که برآورد سهواً یا عمداً اشتباه انجام نمی پذیرد. او می تواند از اعضای تیم دلیل نظرشان و چگونگی استدلالشان را جویا شود. ممکن است گزینش آیتم ها بر خلاف انتظارات مشتری صورت پذیرد. در این صورت تیم باید اطلاعات مناسبی را از دلایل این گزینش در اختیار مالک محصول قرار دهد تا او بتواند بازخورد مناسبی را به مشتری منتقل کند.

پس از گزینش آیتم ها و تکمیل کردن بک لاگ اسپرینت شروع به استخراج وظایف می کند. این عمل می تواند در جلسه ای جداگانه نیز انجام شود. حضور مالک محصول در این جلسه الزامی نیست اما با این حال توصیه می گردد که در عمل استخراج وظایف مالک محصول نیز حضور داشته باشد. چرا که در این عمل آیتم ها با جزئیات بیشتری مورد بررسی قرار خواهند گرفت وحضور مالک محصول می تواند به حل ابهام های احتمالی کمک شایانی کند.

برای عمل استخراج تیم، هر کدام از آیتم ها را به ترتیب اولویت انتخاب می کند. سپس اعضای تیم روی آنچه که باید برای این آیتم انجام شود نظر می دهند. وظایفی که استخراج می شوند نباید خیلی کلی و نه خیلی جزئی باشند. 4 تا 10 وظیفه برای هر آیتم کافی به نظر می رسد. اگر آیتمی بیش از بزرگ است بهتر است آیتم با اطلاع مالک محصول شکسته شود. تعریف وظیفه برای آیتم های شکسته شده به مراتب عمل راحت تری است. اگر تعداد وظایف برخی آیتم ها زیاد شود بهتر است وظایف مرتبط به یکدیگر با هم ادغام شوند. آنچه مسلم است این است که وظایف باید از عملیات فنی درون توسعه استخراج شوند، در نتیجه آنها نباید شبیه به آیتم های بک لاگ تجاری محور باشند. برای هر وظیفه باید برآورد انجام شود و میزان زمان را که یک توسعه دهنده باید بر روی آن تلاش کند تعیین گردد.

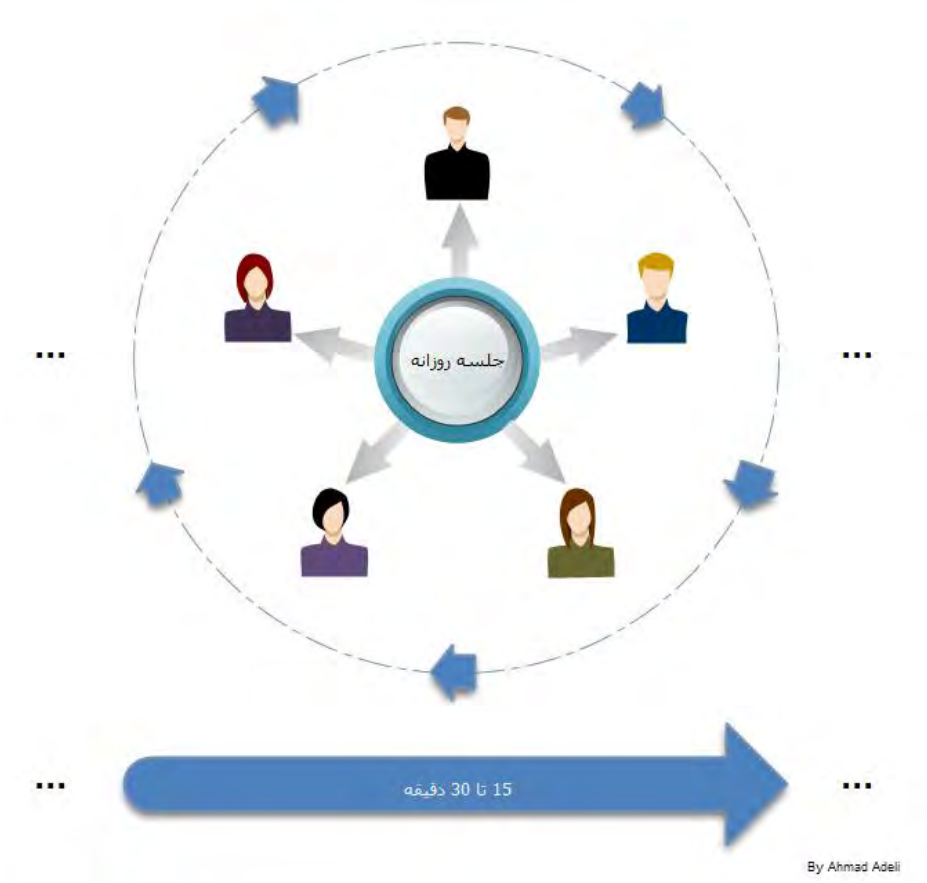
نکته بسیار مهمی وجود دارد این است مجموع برآوردهای وظایف یک آیتم باید در نهایت با برآورد خود آیتم برابر باشد. در این بین اسکرام مستر در طول مدت استخراج وظایف باید به این موضوع نظارت کند چرا که عدم هماهنگی برآوردها با وظیفه ها یشان ممکن است به شکست در برنامه ریزی منجر شود. (در فصل پیش مشاهده شد که اشتباه در برآورد باعث فاصله گرفتن خط واقعی از خط ایده آل می شود). اگر تیم در خصوص وظایف آیتمی مطمئن نیست بهتر است آن آتیم موقتاً رها شود و استخراج وظایف آن پس از کسب اطلاعات بیشتر انجام شود. توجه داشته باشید استخراج کردن وظایف یک آیتم با تأخیر، بسیار بهتر از اشتباه تعیین کردن آن می باشد.

بک لاگ اسپرینت یک لیست باز است و آیتم های آن می توانند در طول اسپرینت تغییر کنند. پس بهتر است در جلسه برنامه ریزی اسپرینت چند آیتم کاندید برای چنین وضعی برآورد شود تا گزینش احتمالی آنها با مشکل مواجه نشد.

جلسه روزانه اسپرینت

توسعه دهندگان همواره باید سعی کنند با یکدیگر ارتباط بر قرار کنند. استقلال در توسعه توسط اعضای می تواند نرم افزار را از یکپارچگی خارج سازد. گروهی کار کردن تنها نمی تواند توسعه را در مسیر درستی قرار دهد؛ بلکه اعضای توسعه باید همیشه ارتباط نزدیکی با یکدیگر داشته باشند. آنها باید تصمیمات را با یکدیگر اتخاذ کنند، به صورت مشترک روی راه حل مسائل بیاندیشند، یکدیگر را از موانع و مشکلات به وجود آمده مطلع کنند. اطلاعات فنی شان را با یکدیگر به اشتراک بگذارند و حتی به جای یکدیگر به کار بپردازند. بدین وسیله می توان سرعت و چالاکی توسعه را افزایش داد و از انحرافات احتمالی در توسعه پیشگیری نمود.

با این حال در این نوع ارتباطات نباید افراط شود چرا که نه تنها یاعث تلف شدن زمان توسعه می شود بلکه احتمال تداخل وظایف درون گروه را بالا می برد. نیازی نیست که برای اخذ هر تصمیم کوچکی همه افراد درون گروه دخالت نمایند به دلیل اینکه ازدیاد جلسات در سطح گروه، باعث آزردگی خاطر اعضا می شود و مسلماً در کارایی آنها تأثیری منفی خواهد گذاشت. اسکرام مستر در طول توسعه باید سعی کند این ارتباطات را در حد تعادل نگه دارد. یکی از بهترین راه های انجام این کار جلسات روزانه اسکرام می باشد(شکل 2-5).



شکل 2-5: جلسه روزانه اسکرام

این جلسه همانطور که از نامش پیدا است به صورت روزانه انجام می شود. کل تیم اسکرام به جز مالک محصول باید در این جلسه حضور داشته باشند. برگزار کننده این جلسه کوتاه شخص اسکرام مستر می باشد. در این جلسه یک ارتباط اجباری و کوتاه بین اعضا ایجاد می شود. هدف این جلسه صرفاً اشتراک اطلاعات می باشد. به هر عضو حاضر در این جلسه زمان کوتاهی داده می شود تا در مورد وظایفشان صحبت کنند. هر فرد باید در ارائه خود به چهار مورد بدون اتلاف وقت رسیدگی کند: وظایفی که روز قبل انجام شده، وظایفی که امروز انجام می شود، مشکلات و موانعی در طول توسعه با آنها مواجهه شده و پیشنهادهای و راهکارهای احتمالی. البته دو مورد آخر نسبت به دو مورد اول کمتر اتفاق خواهند افتاد. همه افراد چنین عملی را باید انجام دهند. در پایان نیز اسکرام مستر در صورت وجود تصمیمات سطح بالا و کلان بدین آنها را به تیم ابلاغ می کند.

مهمترین نکته در کارایی جلسه روزانه اسکرام کوچک نگه داشتن آن است. توسعه می شود که برای تیم های کوچک این جلسه نباید از 15 دقیقه فراتر رود. برای تیم های بزرگ 30 دقیقه می تواند زمان مناسبی باشد. باز هم تعیین دقیق مدت زمان جلسه به خود تیم باز می گردد. یکی دیگر از نکات این جلسه زمان برگزاری آن می باشد. از آنجایی که جلسه روزانه اسپرینت به طور منظم انجام می شود بهتر است زمان ثابتی داشته باشد تا اسکرام مستر برای هر بار برگزاری آن مجبور به اطلاع رسانی به اعضا نشود. عرف بر این است که اعضای تیم این جلسه را به صورت ایستاده انجام دهند. حتی در برخی منابع از این جلسه به عنوان جلسه ایستاده اسپرینت یاد می کنند. ایستاده برگزار کردن جلسه دو حسن بزرگ دارد: اول این که نیازی به آماده سازی تدارکاتی مثل اتاق، میز و صندلی برای جلسه نمی باشد بنابراین هم در هزینه و هم در زمان صرفه جویی خواهد شد. حسن دوم آن این است که اعضا به علت خستگی ناشی از ایستاده بودن، جلسه را چندان طول نمی دهند که این باعث می شود زمان جلسه تا حد ممکن کوچک نگه داشته شود.

جلسه تحویل قابلیت

قابلیت هایی که در طول یک اسپرینت تکمیل می شوند در جلسه تحویل قابلیت به مشتری تحویل داده می شوند. این جلسه در پایان هر اسپرینت با حضور مشتری، مالک محصول، اسکرام مستر و اعضای تیم توسعه

برگزار می شود. هدف این جلسه ارائه قابلیت های اضافه شده به نرم افزار برای مشتری و اخذ تاییدیه وی نسبت به آن ها می باشد. این جلسه یکی از مهمترین جلسات اسکرام است و انجام آن ضروری است.

پیش از این در مورد مفهوم "تکمیل شده" مطالبی ذکر شد. گفته شد که هنگامی می توان چنین صفتی را به یک آیتم منتسب کرد که در وهله اول آیتم توسط تیم توسعه تکمیل شده باشد و پس از آن صحت کارکرد آیتم در مقایسه با آنچه مشتری انتظار دارد تایید شود(شکل 16-3). این تاییدیه توسط نمایندگان مشتری در جلسه تحویل قابلیت صورت می پذیرد. پیش از انجام جلسه تیم باید آیتم ها را بدون کم و کاستی تکمیل کرده باشد و آنها را با تست هایی که پیش از این توسط مالک محصول آماده شدند آزمایش نماید. آیتم های ناقص به هیچ وجه نباید به جلسه تحویل قابلیت راه پیدا کنند. اگر در پایان اسپرینت آیتم های تکمیل نشده ای وجود داشته باشند باید ضمن اطلاع مشتری از این حالت به همان صورتی که وجود دارند به اسپرینت بعدی منتقل شوند. پس از آماده سازی آیتم ها مالک محصول پیش از جلسه باید آنها را یک بار بررسی کند. البته توصیه می شود که مالک محصول در طول توسعه بر روی آماده سازی آیتم ها نظارت کامل را داشته باشد. به طور کلی آیتمی به جلسه تحویل قابلیت راه می یابد از قبل توسط مالک محصول تایید شده باشد. چرا که در صورت رضایت بخش نبودن قابلیت های ارائه شده توسط مشتری، مالک محصول باید پاسخگوی کاستی های به وجود آمده باشد. گاهی انتظارات مشتری غیر واقع بینانه است. حضور تیم فنی توسعه برای پاسخگویی به شبهات و سوالات مشتری می تواند در طول جلسه مفید واقع شود.

با آغاز جلسه یکی از اعضای تیم (به عنوان مثال اسکرام مستر) شروع به ارائه آیتم های اخذ شده از بک لاگ محصول می نماید. علاوه بر آن معادل هر آیتم اخذ شده قابلیت اضافه شده به نرم افزار را نیز نشان می دهد و پاره ای اوقات آنها را تشریح می کند. هر توسعه دهنده ای که آیتم منتخب را تکمیل نموده در صورت لزوم به تشریح مسائل فنی آن خواهد پرداخت. مالک محصول نیز به نوبه خود با افزودن توضیحاتی از منظر تجاری و از دیدگاه مشتری ارائه را تکمیل می کند.

در راستای اخذ باز خورد بر اساس تجربه کاربری مشتری، بهتر است شخص مشتری گاهاً قابلیت های تکمیل شده را آزمایش نماید و با نرم افزار به کار بپردازد. شیوه کار کردن مشتری با نرم افزار خود می تواند بازخورد مناسبی از نرم افزار برای تیم توسعه قلمداد شود. مشتری نیز با این تجربه کاربری دید بهتری نسبت به انتظارات و نیازهای آتی خود پیدا خواهد کرد.

پایان یک اسپرنیت بدون انجام جلسه تحویل قابلیت ممکن است توسعه را وارد شرایط بغرنجی نماید. در این صورت تیم در حالی اسپرینت بعدی را آغاز می کند که از تایید آیتم های قبلی توسعه داده شده مطمئن نیست. این شرایط حتی در جلسه برنامه ریزی اسپرینت هم می تواند تأخیر خود را نشان دهد. گاهی به دلایلی مشتری مایل به حضور در چنین جلسه ای نمی شود. اسکرام مستر باید مشتری را از مزایای این جلسه آگاه کرده و با برنامه ریزی مناسب باید بتواند کارایی جلسه را تضمین کند.

جلسه بازبینی

آخرین جلسه ای که برای یک اسپرینت قابل انجام است جلسه بازبینی می باشد. در این جلسه اعضای تیم در این جلسه به بازبینی عملکرد خود در اسپرینت گذشته می پردازند. جلسه بازبینی تنها از اعضای توسعه به همراه اسکرام مستر تشکیل می شود و نیازی به حضور مالک محصول یا نماینده مشتری در آن وجود ندارد. هدف این جلسه بهبود عملکرد تیم در اسپرینت های آتی است. اعضای توسعه در پایان هر اسپرینت در این جلسه نقاط ضعف و قوت خود را بررسی می کنند و سعی می کنند آنها را در اسپرینت های آینده به ترتیب کاهش و افزایش دهند.

جلسه بازبینی از دو بخش تشکیل می شود: بررسی عملکرد در توسعه و پیشنهاد عملکرد در آن. در بررسی عملکرد، تیم به تحلیل عمکرد خود در اسپرینت قبلی می پردازد. بررسی عملکرد خود نیز به دو بخش تقسیم می شود: کار(آیتم)های انجام شده و کار(آیتم)های انجام نشده. به عبارت دیگر در آیتم های تکمیل شده نقاط قوت را می یابیم و در آیتم های تکمیل نشده نقاط ضعف را. برای این کار کافی است نگاهی به آیتم های تکمیل شده در اسپرینت قبل بیاندازیم و از خود بپرسیم که به چه علت این آیتم ها به خوبی تکمیل شده اند. این پرسش باید به صورت عمومی از تیم پرسیده شود. همه اعضا باید نظرات خود را راجع به این موضوع بیان کنند. در دل این پاسخ ها می توان نقاط قوت تیم را جستجو کرد.نقاط قوت باید در تیم حفظ شوند و آنها را به همان صورتی که هستند باید در سایر اسپرنیت ها تکرار کرد.

گاهی ممکن است تیم، خود تاثیر مثبت برخی کارهای انجام شده را نداند که باعث می شودبه طور مستمر آنها را انجام ندهد. با استخراج این نقاط قوت تیم خود را مجبور می کند که آنها را مجدداً در آینده تکرار کند. از آن طرف نیز تیم با دقت نظر در آیتم های شکست خورده می تواند ضعف های خود را در توسعه پی ببرد و در صدد رفع آنها برآید. رفع نقاط ضعف به مراتب از اهمیت بیشتری برخودار است. از آنجایی که نقاط ضعف ممکن است به شکست آیتم ها بیانجامد، بر طرف سازی آنها در اسپرینت آتی در اولویت است. لازم به ذکر است برخی نقاط قوت نیز نقشی حیاتی در توسعه ایفا می کنند و عدم استمرار آنها نیز می تواند باعث بروز مشکلاتی در توسعه شود و حتی به افزایش نقاط ضعف در تیم منجر شود(شکل 3-5).

بررسی عملکرد در توسعه	
کار(آیتم)های انجام شده(نقاط قوت)	کار(آیتم)های انجام نشده(نقاط ضعف)
Buildها به صورت متوالی اجرا شدند.	موانع ایجاد شده باعث اختلال در زمانبندی از پیش تعیین شده شدند.
همه تست کیس ها طراحی شده اجرا شدند.	به دلیل عدم کد زنی تمیز (Clean) ایجاد تغییر در برخی بخش ها با مشکل مواجحه شد.
به دلیل اجرای منظم جلسه روزانه اسپرینت، موانع بسیار سریع تشخیص داده شدند.	رابط(Interface)ها در برخی زیر سیستم ها ناهمخوان پیاده سازی شدند.
....	...

شکل 3-5: مثال بررسی عملکرد در توسعه

تیم در بخش بررسی به طور اخص روی استخراج نقاط ضعف و فدرت تمرکز می کند. ابتدا نظرات همه اعضا در جایی به صورت موقت ثبت می شود. از آنجایی که نمی توان همه نظرات را اعمال کرد هر کدام از باید نظرات به صورت جداگانه بررسی شوند.

بخش دوم جلسه معمولاً با این عنوان خوانده می شود "چه کارهایی باید متفاوت انجام شوند". این بخش راه حل هایی برای بهبود عملکرد تیم در فرآیند توسعه را نتیجه می دهد. از آنچه در بخش قبل مورد بررسی قرار گرفت برای ارائه راه حل ها استفاده می شود. در این قسمت تیم باید پیشنهاد کند که چگونه می تواند نقاط قوت را در توسعه نگه دارد. همچنین از چه راه های می توان نقاط ضعف را بر طرف ساخت و عملکرد بهتر را جایگزین آن ساخت. در جلسه بازبینی ممکن است راه حل های زیادی به یکباره ارائه شود اما بدیهی است که کل این راه حل ها را نمی توان یکجا در اسپرینت آینده اعمال نمود. توصیه می شود که فقط بخشی از راه حل ها که دارای اولویت بالاتری هستند اعمال شوند و سایر راه حل ها به مرور در طول اسپرینت های بعدی به راه حل های پیشین اضافه شوند.

بسیار مهم است که اسکرام مستر عملیاتی نظارتی را در طول یک اسپرینت روی اجرای چنین راه حل هایی انجام دهد. راه حل های اضافه شده در یک اسپرینت، باید برای اطمینان از صحت عملکردشان در جلسات بازبینی بعدی مورد بررسی قرار گیرند و بازخورد آنها تحلیل شود. چرا که ممکن است خود راه حل، به طور نامناسبی اعمال شده باشد و یا فی نفسه اشتباه باشد. در این حالت ممکن است راه حل نه تنها بهبودی ایجاد نکند بلکه مشکلاتی را نیز به مشکلات قبلی توسعه اضافه کند. تیم در اعمال راه حل ها باید واقع بین باشد. بهبود هایی باید اعمال شوند که قابلیت اجرایی داشته باشند. افراط در اعمال بهبود ها می توانند در روند عادی توسعه خلل ایجاد کند و زمان پروژه را تلف کند.

گاهی ممکن است که به دلایلی چون کمبود زمان، کسالت و ... اعضای تیم میل به انجام چنین جلسه ای نباشند. یا این که ممکن است تیمی به دلیل کارایی بالا در توسعه نیازی به برگزاری این جلسه احساس نکند. اما طولی نمی کشد که به دلیل عدم شناخت از عوامل کارا در توسعه کیفیت عملکرد آن کاهش می یابد. در این گونه مواقع حتی کشف آنچه سابق بر این درست انجام می شده نیز دشوار می شود (چنین شرایطی با اصول بیانیه آجیل در تضاد است). این شرایط ممکن است حتی به شکست کل پروژه بیانجامد. از آنجایی که

بهبود کارایی تیم یکی از وظایف اسکرام مستر می باشد، او باید تیم را همواره به برگزاری جلسه بازبینی تشویق نماید و با اعمال نظارت در جلسه از صحت برگزاری آن اطمینان حاصل کند(شکل 4-5).

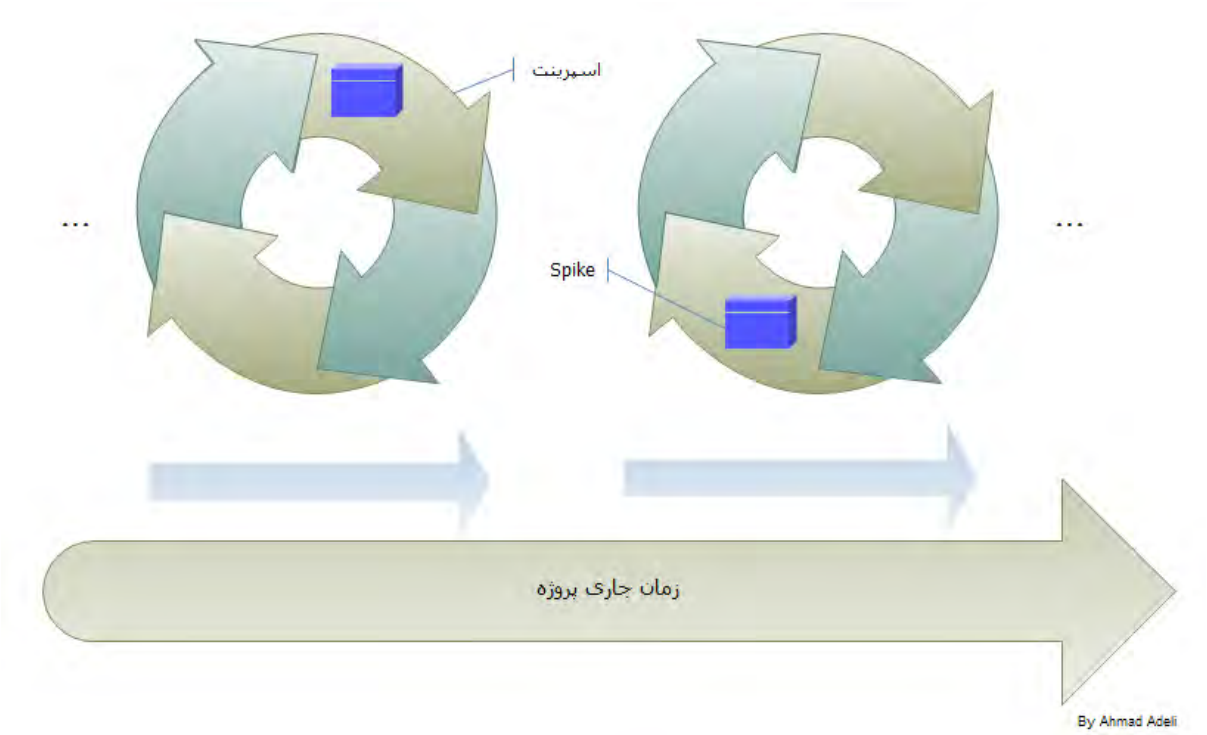
پیشنهاد عملکرد (راه حل های با اولویت)
برای اجرای تست کیس ها باید به مانند آیتم های بک لاگ برنامه ریزی صورت گیرد تا از اجرای همه آن ها تضمین کردد.
برای همه برنامه نویسان باید دوره های آموزشی Clean Codeگذاشته شود تا از با کیفیت بودن کدهای نرم افزار اطمینان حاصل شود.
به دلیل اجرای منظم جلسه روزانه اسپرینت، موانع بسیار سریع تشخیص داده شدند.
....

شکل 4-5: مثال پیشنهاد عملکرد

Spike

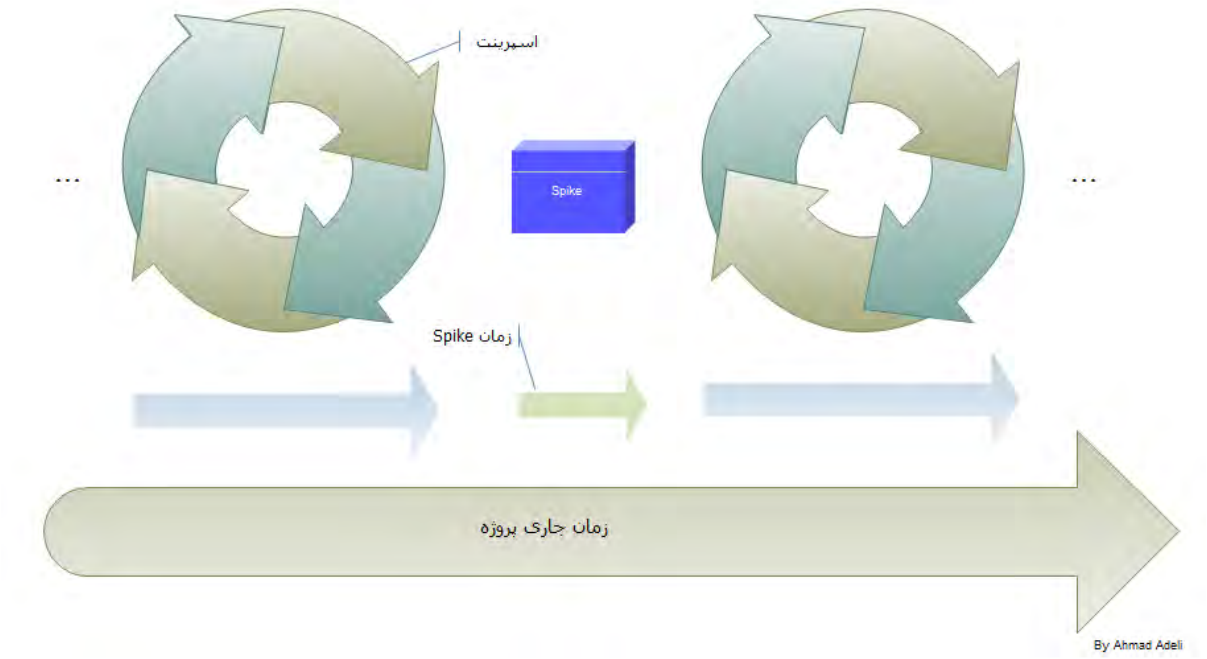
Spike اساساً یک جلسه محسوب نمی شود بلکه یک دوره در طول توسعه می باشد. گاهی لازم است در میان توسعه تیم، روزهایی را به آموزش و تحقیقات اختصاص دهد. به این مجموعه روزهای پیوسته Spike گفته می شود. بهانه برگزاری Spike اصولاً ارتقاع سطح مهارت و دانش تیم توسعه نمی باشد(اگر چه از طرایق آن میسر می شود). Spike از نیازهای پروژه نشأت می گیرد و در راستای کشف راه حل هایی برای موانع موجود صورت خواهد پذیرفت. تیم در بین توسعه ممکن است برای حل مسائلی با فقدان دانش خود مواجه شود. از طرفی عدم رسیدگی به موقع به چنین مسائلی ریسک رویارویی با موانع را در توسعه افزایش می دهد. از این رو تیم یک دوره کوتاه(چند روزه) را برای افزایش میزان مهارت و دانش فنی خود به تحقیقات و آموزش اختصاص می دهد. این تحقیقات و آموزش کاملاً هدفمند بوده و در راستای ارضای نیازهای توسعه شکل خواهند گرفت. در طول اجرای Spike تیم روی هیچ کدام از آیتم ها کار نمی کند و قابلیتی به نرم افزار اضافه نمی کند.

به طور کلی برای اجرای Spike دو رویکرد وجود دارد. در رویکرد اول تیم، Spike را در یک اسپرینت انجام می دهد. در این رویکرد قسمتی از زمان (روزهای) اسپرینت برای اجرای Spike اختصاص داده می شوند. در این روزها تیم به امور فنی توسعه نمی پردازد. لازم نیست که همه اعضا در Spike شرکت داشته باشند. برخی توسعه دهندگان همچنان می توانند به کار خود در توسعه بپردازند(شکل 5-5).



شکل 5-5: مدل مفهومی Spike در اسپرینت

در رویکرد دوم، Spike بین اسپرینت ها انجام می شود. در این نوع Spike به دلیل این که اسپرنیتی وجود ندارد همه اعضا می توانند همزمان در Spike شرکت داشته باشند(شکل 5-6). انتخاب نوع Spike به عهده تیم توسعه می باشد. معمولاً نوع اول برای حل موانع حساس و اضطراری استفاده می شود و نوع دوم برای مسائلی است که حل آنها زمان بیشتری طلب می کنند.



شکل 5-6: مدل مفهومی Spike بین اسپرینت

اصولاً بهتر است Spike را نیز مانند دیگر آیتم ها برنامه ریزی نمود به خصوص اگر یک Spike قرار است درون یک اسپرینت انجام شود. در این صورت حجم اسپرینت با توجه به Spike موجود در آن به درستی تخمین زده خواهد شد و آیتم ها به طور کامل در اسپرینت انجام می شوند. برنامه ریزی برای Spike هایی که بین اسپرینت ها اجرا

می شوند هم دارای اهمیت است. برنامه ریزی Spike ها باعث می شود که در تعداد اسپرینت ها و زمان Release اختلالی ایجاد نشود. چرا که برخی از Spike های که بین اسپرنیت ها اجرا می شوند بخشی از زمان Release را به خود اختصاص خواهند داد. با این وجود گاهی حساسیت و ضرب العجل برخی موانع باعث می شود برخی Spikeها خارج از برنامه اجرا شوند. احتمال اجرای چنین Spikeهایی بسیار پایین است اما به هر صورت تیم باید خود را برای مواجهه با هر شریطی آماده سازد.

جلسات موجود در اسپرینت جلساتی تکراری هستند. بدین معنی که در طول توسعه به کرات و به صورت پریودیک انجام قابل انجام اند. علاوه بر جلسات شرح داده شده در این فصل جلسات دیگری نیز وجود دارد که کمتر انجام می شوند و برگزاری آنها اختیاری است. این جلسات عموماً توسط خود تیم ها ابداغ، تنظیم و در اسکرام سفارشی می شوند. گاهاً جلسات مرسوم در اسکرام مانند جسله برنامه ریزی اسپرنیت، جلسه روزانه و ... نیز توسط برخی تیم ها در جهت بهبود کارایی سفارشی می گردند.

فصل ششم:

تخمین و برآورد

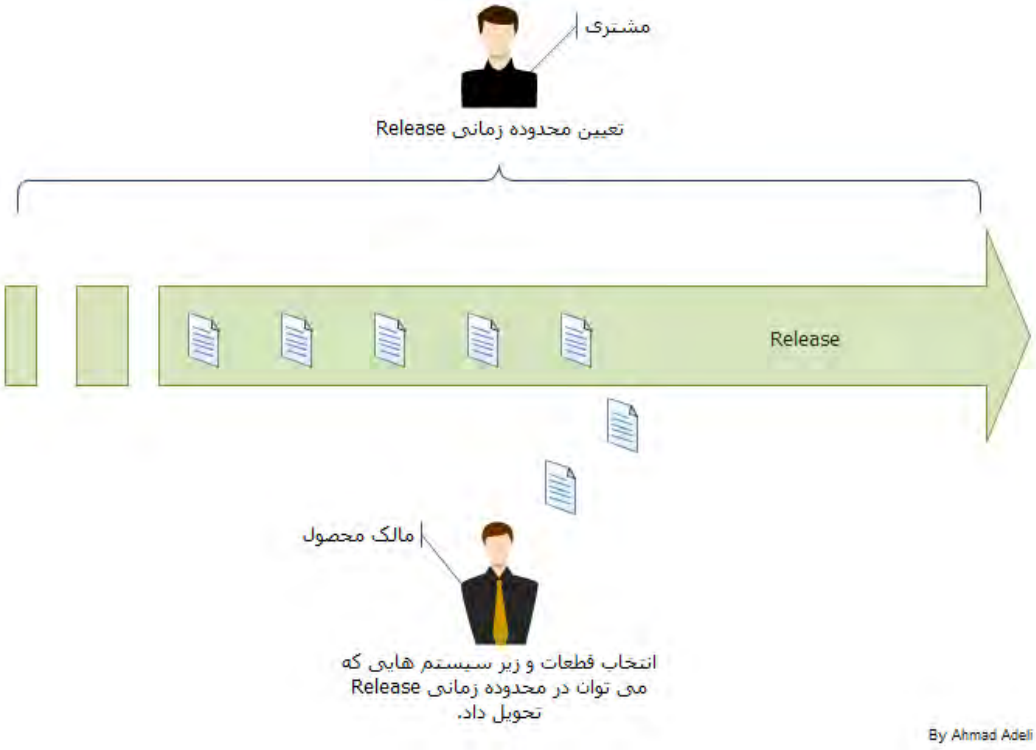
برای صحیح انجام دادن هر کاری برنامه ریزی دقیقی نیاز است. این برنامه ریزی دقیق شامل برآورد زمان انجام آن کار نیز می باشد. اجرای یک پروژه با استفاده از اسکرام نیز از این قائده مستثنی نیست. به دلیل عدم قطعیت در پروژهای نرم افزاری همواره زمان بندی و برنامه ریزی دشوار بوده و اغلب از عمل تخمین استفاده می شود. تخمین به معنی گمانه زنی در رابطه با موضوعی است و هر چه این گمان دقیق تر باشد طبیعتاً ضمانت اجرایی بیشتری هم خواهد داشت. توسعه دهندگان هیچ گاه نمی توانند به طور دقیق امور مربوط به یک پروژه را تعیین کنند از طرفی بدون تعیین آنها نیز پروژه قابل اجرا نمی باشد. توسعه دهندگان برای این کار اغلب از تخمین به جای تعیین استفاده می کنند.

در تخمین سعی می شود تا حد ممکن آنچه به واقع اتفاق خواهد افتاد پیش بینی شود. برای دست یابی به یک تخمین مناسب بهتر است از تکنیک هایی که پیش از این در پروژه های دیگر آزمایش شدند استفاده نمود تا ضمن کمینه کردن زمان، ریسک خطا در آن نیز به حداقل برسد. توسعه دهندگان در اسکرام در بسیاری از مواقع از تخمین و برآورد کمک می گیرند. به طور کلی چهار بخش مهمی که باید برای آنها برآورد صورت بگیرد عبارت اند از زمان Release، حجم اسپرینت، برآورد آیتم های بک لاگ و روند اجرایی اسپرینت می باشد. در ادامه تکنیک هایی در رابطه با این چهار دسته ارائه می شوند:

تخمین Release

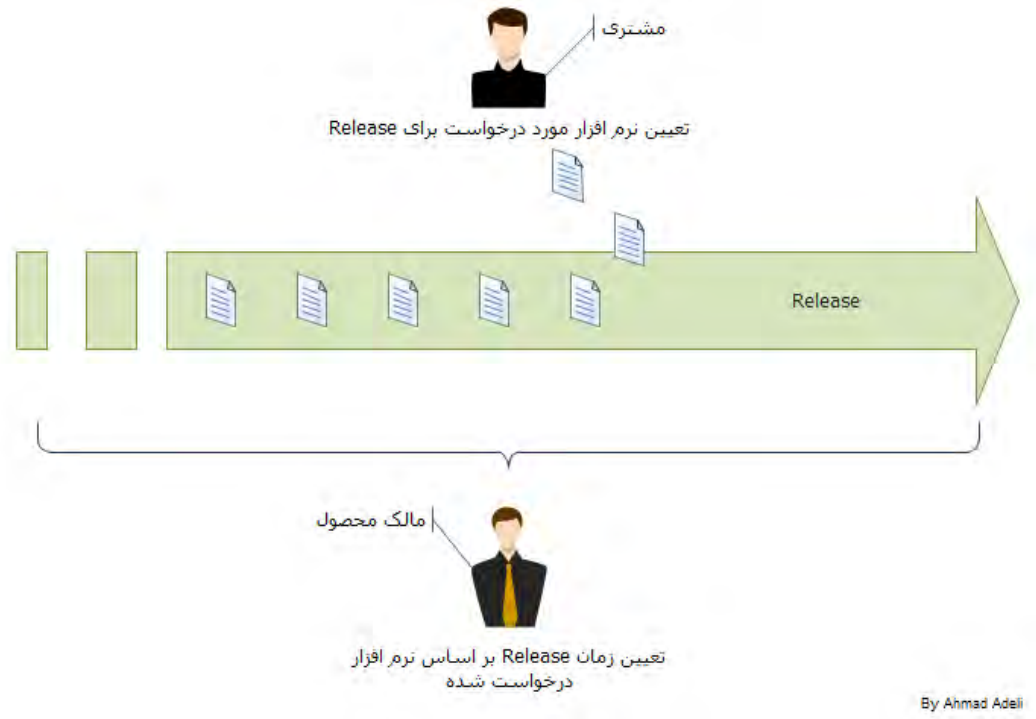
همانطور که پیش تر گفته شد Releaseها تکرارهایی هستند که بخشی از نرم افزار کار کننده در آنها ساخته می شود. بدیهی است این تکرارها در زمان محصور هستند. این زمان در Releaseهای مختلف ممکن است به صورت متفاوتی انتخاب شود. زمان Release معمولاً تابع مشتری است و با هماهنگی او انتخاب می شود.

Releaseها معمولاً به دو صورت برنامه ریزی می شوند: *زمان تعیین شده، نرم افزار درخواست شده*. در صورت اول، معمولاً مشتری زمانی را برای تحویل نرم افزار تعیین می کند(شکل 1-6). این زمان ممکن است در ابتدای پروژه برای همه Releaseها انتخاب شود. تیم اسکرام موظف است بر اساس زمان تعیین شده نرم افزار(یا بخشی از نرم افزار) کار کننده را ارائه کند. این زمان همچنین می تواند برای هر Release متفاوت باشد و با هر بار اجرای یک Release انتخاب شود. مالک محصول باید بر اساس این زمان تعیین شده آیتم ها و قابلیت هایی را که تیم می تواند آماده کند در بک لاگ محصول وارد سازد. در ابتدای Release بدیهی است که این عمل چندان کامل و با جزئیات صورت نمی پذیرد و اغلب به مسائل کلی منتج می شود. این مسائل کلی می توانند زیر سیستم های یک سیستم نرم افزاری باشند. برآورد این زیر سیستم ها کار آسانی نبوده و به اجرای برخی تکنیک ها نیازمند است(با این تکنبک ها در ادامه فصل آشنا خواهید شد).



شکل 1-6: برنامه ریزی Release به روش زمان تعیین شده

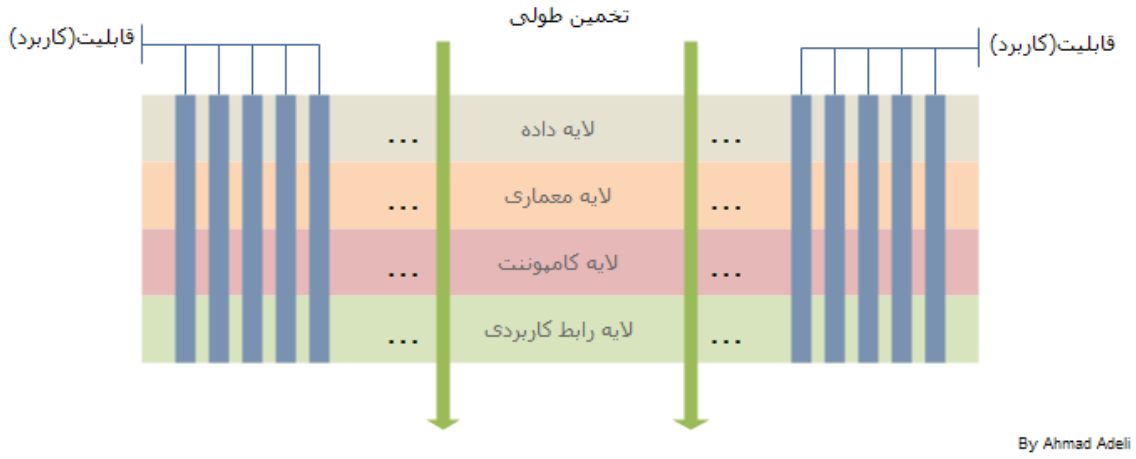
صورت دوم برنامه ریزی Releaseها، نرم افزار درخواست شده می باشد(شکل 2-6). در این شکل برنامه ریزی مشتری بدون تعیین زمانی خاص، قابلیت هایی را در انتهای تکرار بعدی به عنوان نرم افزار کار کننده طلب می کند. این قابلیت ها مانند حالت قبل به صورت کلی ارائه می شوند. این بار عکس حالت قبل تیم اسکرام وظیفه تعیین زمان Release را بر عهده دارد. ناگفته پیدا است که این زمان نباید از حد و حدود مورد نظر مشتری فراتر رود. این نوع برنامه ریزی Release با استفاده از تکنیک هایی قابل انجام است.



شکل 2-6: برنامه ریزی Release به روش نرم افزار درخواست شده

تخمین طولی

همانطور که می دانید یک نرم افزار از لایه هایی چون بانک داده، لایه معماری، لایه کامپوننت، لایه رابط کاربردی و ... تشکیل شده است. این لایه ها به صورت مجازی روی هم قرار دارند. هر قابلیتی که در نرم افزار پیاده سازی می شود از همه لایه استفاده می کند. هر کدام از قابلیت ها برای پیاده سازی از اولین لایه به سمت آخرین لایه در طول حرکت می کند(شکل 3-6). به تخمینی که بر اساس قابلیت ها صورت گیرد تخیمن طولی گویند. در تخمین Release همانطور که گفته شد با جزئیات و به صورت قابلیت صورت نمی گیرد. در این نوع تخمین زیر سیستم ها و قطعات بزرگ توسط تیم برای تعیین Release (زمان یا حجم آن) برآورد می شوند. دو تکنیک برای چنین برآوردی وجود دارد: میانگین برآورد ها و طبقه بندی برآوردها.



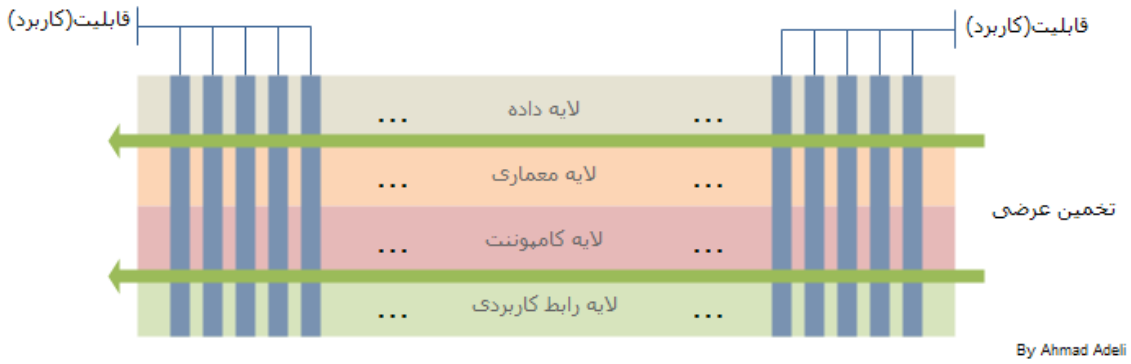
شکل 3-6: تخمین طولی (بر اساس قابلیت)

اولین تکنیک بر این اساس است که اعضای گروه ابتدا هر کدام به صورت مستقل نظر خود را راجع به برآورد قطعات و زیر سیستم ها (که به صورت مجزا انجام می شود) ارائه می کنند. بهتر است برای حفظ استقلال نظرات آنها به صورت نوشتاری ارائه گردند تا نظر یک توسعه دهنده بر دیگری اثر گذار نباشد. امید است که این برآوردها نزدیک به یکدیگر باشند؛ تا با میانگین گیری آن بتوان به برآوردی واحد رسید. اما در صورتیکه برآوردها به هم نزدی نباشد حتماً باید با استفاده از تکنیک شفاف سازی به درک واحدی از موضوع مورد بحث رسید. این روش برای اغلب قابلیت ها و زیر سیستم ها قابل اعمال است.

در تکنیک اول هر توسعه دهنده بر اساس دیدگاه خویش برآوردی آزادانه را پیش خواهد گرفت. این موضوع باعث می شود اتفاق نظر روی یک برآورد به دشواری امکان پذیر باشد. در تکنیک طبقه بندی برآورد ها، هر یک از اعضای تیم فقط می توانند برآوردهای مشخصی را گزینش کنند. این برآورد از پیش تعیین شده می باشند به طور مثال برآوردها برای یک تیم ممکن است به صورت 1، 2، 4، 9، 13، 18 و ... تعیین شوند. این کار باعث می شود برآوردها دارای حد و حدودی مشخص باشند و از دقتی افراطی در تعیین آنها پرهیز شود. در تکنیک نیز با استفاده از شفاف سازی می توان به راحتی برآوردی واحد را برای قابلیت مورد بحث ارائه کرد.

تخمین عرضی

این نوع تخمین عکس حالت قبل به صورت عرضی نرم افزار مورد انتظار را مورد بررسی قرار می دهد. برای زمان هایی که تعداد و پیچیدگی قابلیت ها زیاد باشد می توان از روش عرضی برای برآورد یک Release استفاده نمود. در این روش هر لایه به صورت جداگانه مورد بررسی قرار می گیرد و برای آن برآوردی جداگانه ارائه می شود(شکل 4-6). حجم هر لایه بر اساس نرم افزاری که باید پیاده سازی شود پیش بینی می شود. نتیجه این بررسی ها برآورد نرم افزار در یک Release می باشد.



شکل 4-6: تخمین عرضی(بر اساس لایه ها)

محدودیت این روش در این است که برای برآوردی صحیح نیاز به تجربه زیادی در زمینه توسعه از سوی اعضای اسکرام می باشد چرا که در این روش قابلیت ها به وضوح مشاهده نمی شوند تا بتوان بر اساس آنها تخمینی صحیح را صورت داد. این روش به ویژه برای سیستم هایی که چندان جدید نیستند بسیار کارآمد است.

برآورد ریسک ها و موانع

هنگامی که به یک توسعه دهنده به یک قابلیت یا یک زیر سیستم نگاه می کند در حقیقت به چگونگی پیاده سازی آن می اندیشد. توسعه دهندگان بر اساس مهارت و تجربه خود اغلب برای حل مسائل از راه حل های مشخص و آزموده شده ای استفاده می کنند. این راه حل ها تا زمان پیاده سازی چندان معتبر نیستند و بر اساس حدس و گمان ارائه می شوند. اما با رویکردی واقع بینانه اغلب این راه حل به همین صورت پیاده سازی می شوند. از آنجایی که تخمین قابلیت ها بر اساس همین راه حل ها انجام می شوند بسیار مهم است که ریسک استفاده از آنها نیز پیش بینی شود. ریسکی که ممکن است منجر به ایجاد یک مانع شود و مانعی که در صورت عدم استفاده از آن راه حل ظاهر نمی شد. بنابراین یا باید روش و زمان این مانع نیز در برنامه ریزی لحاظ شود و یا با تخمین مجدداً با استفاده از راه حلی جدید صورت پذیرد.

هر تخمینی ممکن است با اشتباه روبرو شود؛ چرا که همانطور که گفته شد تخمین روش دقیقی نیست و اساس مستحکمی برای اجرا ندارد. پس در نتیجه بهتر است راهی برای جبران این اشتباهات در Release تعبیه شود. به عبارتی دیگر این مهم است که قدری بیش از آنچه که برآورد شده برای تخمین ارائه شود. این باعث می شود که در صورت خطا در برآوردها با کمبود زمان مواجهه نشویم.

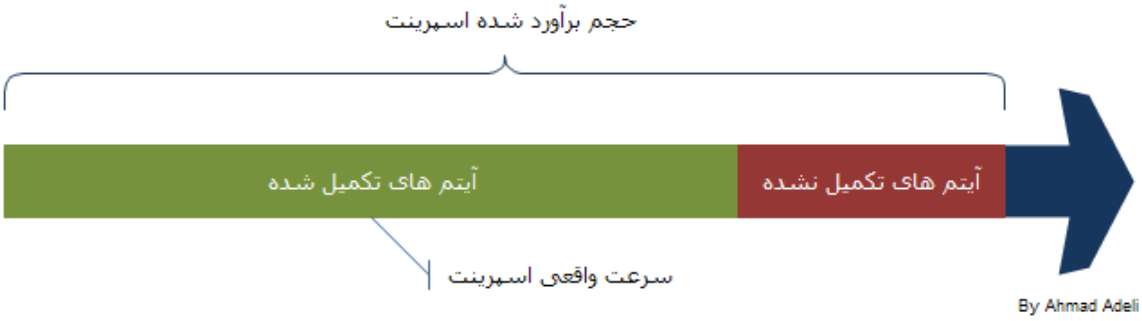
تخمین Release به هر صورتی که انجام شود وابسته به نیازها و نظر مشتری است؛ زیرا اوست که در نهایت در انتهای Release نرم افزار تحویل می گیرد. اما از طرفی ممکن است عدم اطلاع او از چگونگی توسعه باعث انتظار غیر واقعی او از تخمین Release شود. به هر حال به دلیل دو طرفه بودن این موضوع همواره باید بالانسی بین آنچه مشتری انتظار دارد و آنچه تیم به صورت واقعی می تواند انجام دهد برقرار شود.

تخمین حجم اسپرینت

اسپرینت ها همانند Releaseها دارای حجم می باشند. به این معنی که چند آیتم بک لاگ در آنها قابل انجام است. اما باید توجه شود که آیتم های بک لاگ متفاوت هستند و نباید بر اساس تعداد آنها اسپرینت ها را برنامه ریزی کرد. آیتم های بک لاگ همانطور که پیشتر گفته شد دارای واحدی هستند به نام Story Point که معادل نیرو-روز می باشد. از آنجایی که آیتم های بک لاگ بر اساس این واحد سنجیده می شوند، حجم اسپرینت را نیز باید بر این اساس برآورد نمود(بهترین تمثیل اسپرینت می تواند یک ترم درسی در دانشکاه باشد. در یک ترم می توان به اندازه 20 واحد درس اخذ کرد و ممکن است تعداد درس های مختلفی در این 20 واحد قرار بگیرند). در حقیقت با تخمین حجم اسپرنیت تعیین می کنیم با چند نیرو، چند روز می توانیم در یک اسپرینت نوعی روی توسعه انرژی بگذاریم. هر آیتم نیرو و زمان خاصی را طلب می کند و تا زمانی که اسپرینت گنجایش داشته باشد آن آیتم ها برای یک لاگ آن اسپرینت اخذ می شوند. حال باید دید که چگونه می توان حجم اسپرینت را تخمین زد. برای این نوع تخمین تکنیک های زیر پیشنهاد می شوند:

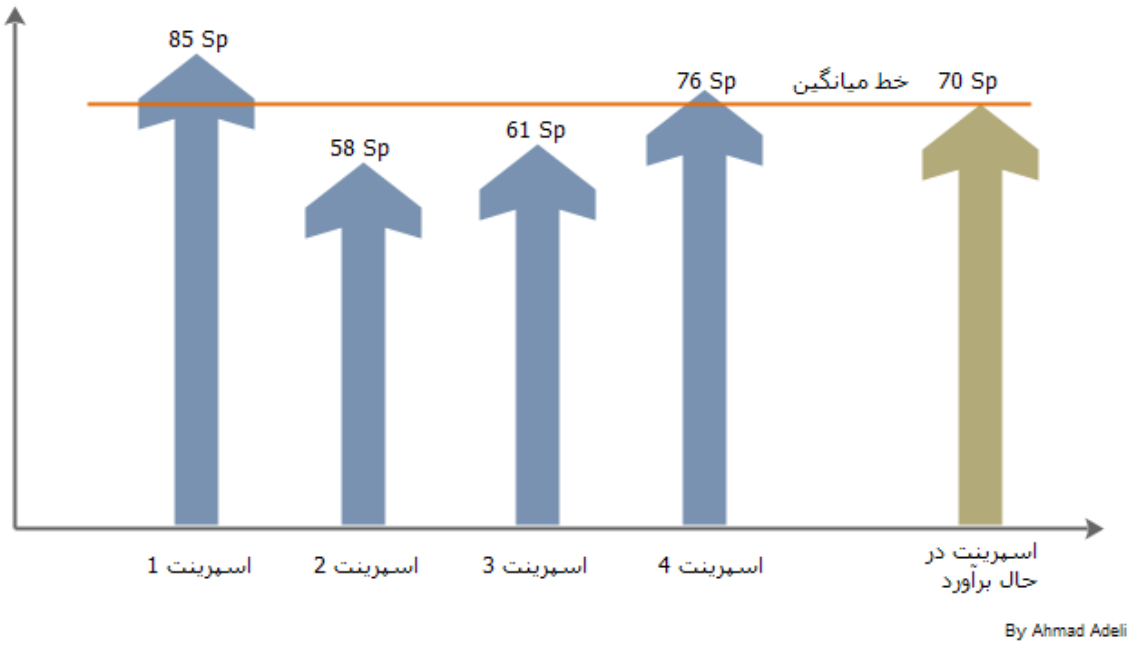
تخمین بر اساس میانگین سرعت(Velocity)

هر اسپرینت دارای حجمی است که پیش از اجرای آن تعیین می شود. اجرای یک اسپرنیت ممکن است تکمیل همه Story Point های تعیین شده در ابتدای اسپرینت را پشتیبانی نکند. به این معنی که امکان دارد در پایان اسپرینت بخشی از Story Point ها تکمیل نشوند و مقدار کل Story Point های تکمیل شده از مقدار برآورد شده ابتدایی کمتر باشد. طبق تعریف سرعت اسپرینت به معنی اجرای واقعی اسپرینت بر اساس Story Point است که می تواند برابر یا کمتر از حجم برآورد شده ابتدایی یک اسپرینت باشد(شکل 5-6). اعضای اسکرام (با تسامح) اغلب مایلند این سرعت را همواره بهبود ببخشند. اما این میل نباید باعث شود که حجم اسپرنیت ها سیر صعودی به خود بگیرد؛ چرا که اگر تیم در اسپرینت های قبل نتوانست حجم بالایی از Story Point را به اتمام برساند احتمالاً در این اسپرینت نیز نخواهد توانست به آن فائق آید.



شکل 5-6: حجم و سرعت یک اسپرینت

در این نوع تخمین منطق بر این است که با گرفتن میانگینی از سرعت همه اسپرینت های انجام شده تیم خواهد توانست به حجم مناسبی برای اسپرینت جاری دست پیدا کند(شکل 6-6). این تکنیک تخمین دو پیش شرط دارد: اول اینکه باید حداقل سه اسپرنیت وجود داشته باشد که بتوان از آنها میانگین گرفت. به عبارت دیگر این تکنیک برای اسپرنیت های ابتدایی قابل استعال نمی باشد. دوم اینکه اگر سرعت اسپرینت های گذشته به هم نزدیک نباشد گرفتن میانگین چندان راه گشا نخواهد بود. اختلاف زیاد سرعت اسپرینت ها گذشته ممکن است ناشی برآورد ناشیانه و یا بروز مشکلات جدی در توسعه باشد که این نیز به نوبه خود می تواند روند توسعه را با چالش هایی جدی روبرو سازد.



شکل 6-6: تخمین اسپرینت بر اساس میانگین سرعت(Velocity)

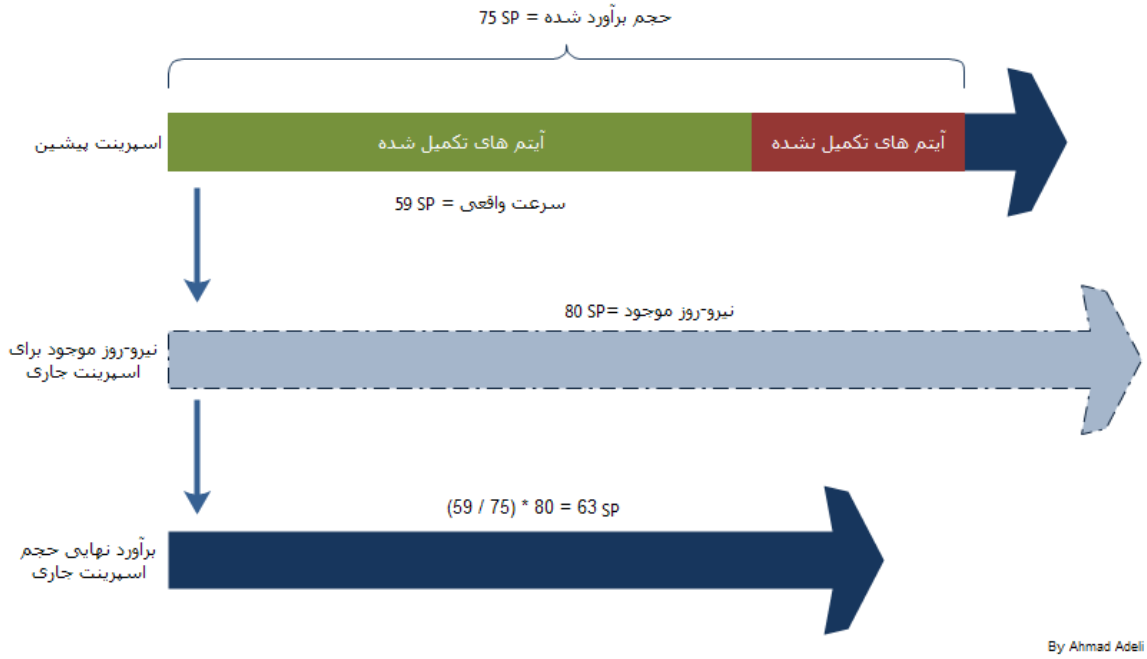
تخمین بر اساس سوابق اسپرینت ها

آنچه تکنیک پیش چندان بدان توجه نمی کند سیر اتفاقاتی است که در اسپرینت های اخیر رخ داده است. فرض شود در ابتدای پروژه به دلیل عدم هماهنگی افراد یا بیرون رفتن ناگهانی برخی سرعت اسپرینت نسبت به حجم آن بسیار پایین باشد. اما با ادامه روند توسعه یکپارچگی نیروی انسانی افزایش یافته و متعاقب آن سرعت اسپرنیت ها بیشتر شده است. در مثالی دیگر ممکن است در روند اسپرنیت پیشین حادثه ای باعث خسران دیدن منابع فیزیکی سازمان(مانند کامپیوترهاو ...) شود. بدیهی است که این گونه مسائل با عوض شدن اسپرینت حل نخواهند شد و همچنان به قوت خود باقی هستند. با در نظر گرفتن این گونه مسائل اگر تخمین بر اساس سرعت اسپرینت های اخیر انجام شود که تیم شانس بیشتری برای یکسان کردن حجم پیش بینی شده و سرعت واقعی انتهای اسپرینت دارا خواهد شد. نکته مهمی که برای تخمین حجم اسپرینت وجود دارد این است که تخمین باید بر اساس سرعت واقعی اسپرینت های گذشته صورت پذیرد و نه حجم پیش بینی شده آنها؛ چرا که حجم صرفاً یک پیش بینی از اسپرینت است و سرعت اجرا واقعی آن اسپرینت می باشد.

برای اجرا این تکنیک لازم است که نیرو و زمان اسپرینت در حال برآورد با نیرو و زمان اسپرینت ماقبل برابر باشد(که اغلب این چنین است). در غیر این صورت باید تبدیلی در این بین انجام داد تا اشکالی در برآورد حجم اسپرینت پیش نیاید. برای این عمل بهتر است فرمول ساده زیر استفاده نمود:

نیرو – روز اسپرینت جاری*(حجم پیش بنی شده اسپرینت قبل / سرعت واقعی اسپرینت قبل)=تخمین حجم اسپرنیت جاری

بخش اول فرمول نسبت سرعت به حجم اسپرینت قبل می باشد. این نسبت عددی بین 0 تا 1 می باشد و به عنوان ضریبی برای نیرو-روز اسپرینت واقعی استفاده می شود. اگر این عدد 1 باشد به این معنی است که حجم با سرعت اسپرنیت برابر بوده و هر آنچه نیرو-روز برای اسپرینت جاری داریم برای تخمین استفاده می کنیم. اما اگر کمتر از یک بود، باعث می شود قدری کمتر از استعداد موجود استفاده کنیم(شکل 7-6).



شکل 7-6: تخمین اسپرینت بر اساس سوابق اسپرینت های پیشین

در این تکنیک باید کمی با احتیاط عمل کرد؛ چرا که ممکن است اسپرینت قبل به دلایلی اسپرینت مطلوبی نباشد و تاثیر بد خود را بر روی اسپرینت جاری نیز بگذارد. در این گونه مواقع بهتر است از میانگین دو اسپرینت قبل و یا از اسپرینت های قبل تر برای برآورد استفاده کرد.

تخمین بر اساس تجربه

هر دو تکنیک قبل دارای یک اشکال بودند و آن هم این است که برای اسپرینت های ابتدایی و شرایط خاص قابل استفاده نیستند. حتی با وجود این تکنیک ها باز هم در تیم های اسکرام اعضا تجربه خود را در تخمین هرچه صحیح تر حجم اسپرینت تأثیر می دهند. در اسپرینت های ابتدایی هیچ سابقه کاری برای تیم وجود ندارد، از این رو باید بر اساس تجربه تیم در پروژه قبلی تخمینی مناسب را ارائه داد. گاهی نیز به دلیل ایجاد شرایطی خاص (مثلاً بیرون رفتن چند تن از اعضا در میانه اسپرینت) تیم اسکرام مجبور به تغییر برآورد حجم اسپرینت می شود.

ناگفته پیدا است که این روش برای تیم هایی که دارای تجربه و هماهنگی درون تیمی می باشند بهتر قابل اجرا است. اما تیم هایی که از این تجربه بی بهره هستند احتمالاً در اسپرینت های ابتدایی دچار مشکل برآورد می شوند که این از نظر فنی امری است طبیعی و معمول. در این گونه مواقع بهتر است تیم از حجم پایین در اسپرینت های ابتدایی استفاده کند تا ضمن کسب تجربه کافی از هماهنگی درون تیمی نیز برخوردار شود.

برآورد آیتم های بک لاگ

در فصل پیش دیدیم این برآورد بک لاگ در جلسه برنامه ریزی اسپرینت انجام می شود. این برآورد به منظور آماده سازی بک لاگ اسپرینت انجام می شود. این عمل با حضور اعضای تیم توسعه و مالک محصول انجام می شود. حضور مالک محصول برای چنین عملی الزامی است. مالک محصول باید آیتم هایی که در بک لاگ محصول قرار دارند را یک به یک برای تیم تشریح کند و موشکافانه آنها را مورد بحث قرار دهد. پس از این عمل معمولاً اعضا هر یک به چگونگی ساخت آیتم می اندیشند. در این بین اگر ابهامی وجود داشته باشد می توانند مستقیماً از مالک محصول کمک بگیرند. بهتر است بررسی آیتم ها با بحث و گفتگو میان اعضا همراه باشد تا همه ابعاد آیتم برای تیم شفاف شود. معمول بر این است راه حل ها ابتدا توسط اعضا ارائه شوند و سپس در حضور جمع بررسی شوند. در نهایت برآورد بر اساس راه حلی که پیشنهاد می شود، صورت خواهد پذیرفت.

تخمین آیتم ها

تخمین آیتم همانند تخمین زیر سیستم ها در Release به دو صورت میانگین و طبقه بندی برآوردها انجام می پذیرد. ابتدا هر یک از اعضا به صورت جداگانه و مستقل برآورد خود را از آیتم بیان می کند. مجدداً تأکید می شود بهتر است از ذکر شفاهی و ترتیبی نظرات خودداری شود چرا که اثر گذاری نظرات دیگران بخ خصوص افراد ارشد بر دیگران اجتناب ناپذیر است. در صورتی که برآوردها نزدیک به یکدیگر باشد (که احتمالاً با شفاف سازی انجام شده چنین خواهد بود) با گرفتن میانگینی از برآوردها می توان به برآورد واحدی رسید. اما اگر برآوردها چندان به هم نزدیک نباشد بهتر است از برآوردهای طبقه بندی شده و از پیش تعریف شده استفاده کرد.

مزیت این روش در این است که از ریز شدن افراطی در برآورد توسط افراد جلوگیری می شود و احتمال یکسان شدن برآوردها بیشتر می شود. در این صورت نیز ممکن است برآورد ها چندان به یکدیگر نزدیک نباشد. این خود می تواند دلیلی برای مشابه نبودن دیدگاه اعضای تیم در خصوص یک آیتم باشد. بدیهی است که هر برآوردی که توسط فردی انجام شود دارای توجیه خاص خود می باشد. به اشتراک گذاشتن این توجیحات خود می تواند موجبات رسیدن به یک برآورد واحد را مهیا سازد.

تخمین تجربی آیتم ها

همانطور می دانیم اغلب تصمیمات در مهندسی نرم افزار بر اساس تفکرات انتزاعی انجام می شود. به علت این که ماهیت مهندسی نرم افزار قابل فرموله شدن نمی باشد، چندان نمی توان از قوانین کم انعطاف و محاسباتی در آن استفاده کرد. در بسیاری از تصمیمات در اسکرام نقش تجربه و نوع تفکر بسیار مهم جلو داده می شود. اعضای اسکرام نیز معمولاً تجربه توسعه خود را در برآورد آیتم ها تاثیر می دهند. گاهی با استفاده از همین روش بدون استفاده از هیچ تکنیکی و صرفاً با ارائه نظری معتبر از سوی یکی از اعضا برآورد به سرعت انجام می شود. برای تیم های با تجربه حتی گاهی پیش خواهد آمد که با نگاه کردن به یک آیتم به راحتی برآورد صحیحی از آن بدست می آید. اما در طرف دیگر ماجرا در صورت افراط در استفاده از روش های تجربی ممکن است اشتباهات جبران ناپذیری در برآورد آیتم ها رخ دهد. پس بهتر است از هر دو روش به شکلی متوازن استفاده شود تا ضمن سرعت عمل در برآورد آیتم ها، ریسک برآورد اشتباه نیز پایین باشد.

شکستن آیتم ها

گاهی آیتم هایی که توسط مالک محصول انتخاب می شوند بسیار بزرگ هستند. این امر خود باعث کاهش توانایی تیم در رسیدن به یک برآورد مطلوب می شود. در صورتی تشخیص داده شود یک آیتم با شکسته شدن به آیتم های کوچکتر بهتر برآورد خواهد شد، باید از مالک محصول درخواست شود که این عمل را بدون لطمه

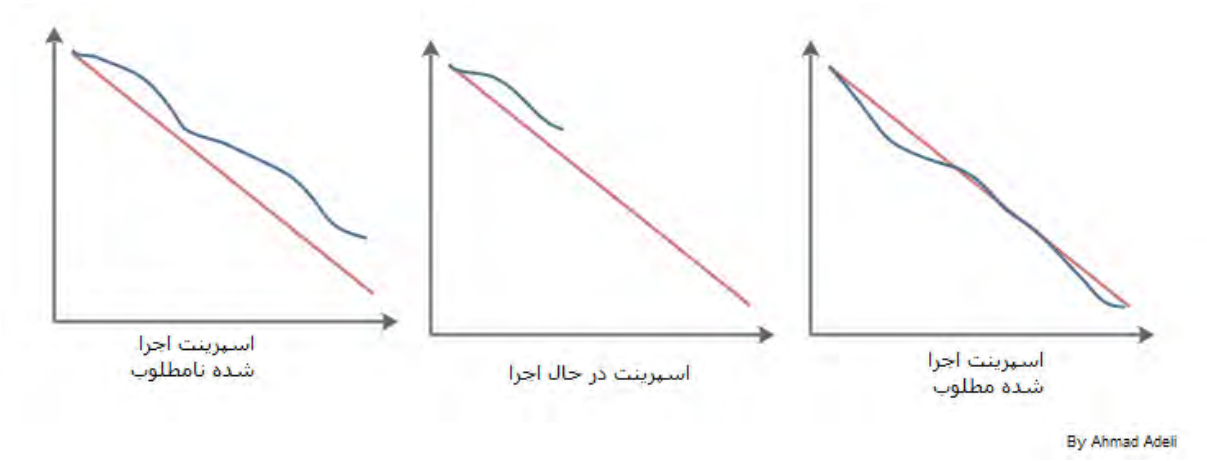
دیدین یا از بین رفتن کاربردهای آن آیتم بزرگ انجام دهد. یادآوری می شود که تنها فردی می تواند این در بک لاگ محصول تغییری ایجاد کند شخص مالک محصول می باشد. حالت عکس آن نیز وجود دارد گاهی آیتم هایی آنقدر کوچک هستند که برآورد مستقل آنها ناکارآمد و غیر ضروری است. بهتر است در این گونه مواقع با ترکیب کردن آنها آیتم های بزرگتری ایجاد کرد و برآورد مناسب تری را به آن نسبت داد.

پیش بینی روند اجرایی اسپرینت

یکی از دغدغه های اصلی توسعه دهندگان در راستای توسعه، چگونگی روند اجرایی اسپرینت می باشد. این دغدغه همواره حتی در طول اجرای یک اسپرینت نیز وجود دارد. تیم توسعه همیشه باید وضعیت روزهای آتی یک اسپرینت را پیش بینی کند. چرا که در مشکلی و مانعی قابل حل است که پیش از وقوع آن تشخیص داده شود. گاهی در توسعه آیتم زودتر از آنچه تصور می شود تکمیل می شوند و گاهی نیز اسپرینت پایان می یابد و همچنان آیتم هایی وجود دارند که حتی کار بر روی آنها شروع هم نشده است. ممکن است در ادامه با پیش گرفتن رویکردی مانعی پیش آید یا برخی توسعه دهندگان به دلایلی قادر به ادامه کار نباشند. همه این مشکلات برای یک پروژه امری است اجتناب ناپذیر؛ اما آنچه مهم است شیوه رویارویی تیم توسعه با آنها و نحوه حل کردن آنها است.

اکثر مشکلات دارای راه حل هستند. تنها نگرانی در مورد آنها زمان تشخیص آنها است. اگر تیم بتواند به خوبی پیش بینی کند که در زمان آتی چه مشکلاتی به وجود خواهند آمد، حتی این توان وجود دارد که از وقوع آنها نیز پیشگیری به عمل آید. مشکلاتی که در آینده قرار است به وجود آیند اولین اثر خود را در زمان حال خواهند گذاشت. به این معنی که اگر به خوبی روند اجرایی اسپرینت در طول اجرای آن را مطالعه کنیم به به اطلاعات بسیار با ارزشی راجع به اتفاقات آینده اسپرینت خواهیم رسید. یکی از این نوع مطالعات (و به جرأت مهمترین آن) Sprint BurnDown می باشد. با تغییر وضعیت خط واقعی که نسبت به خط ایده آل سنجیده می شود، به راحتی می توان آینده یک اسپرینت را پیش بینی کرد و برای بهبود آن اقداماتی صورت داد. این اقدامات می تواند اضافه کردن نیروی انسانی بیشتر، افزایش ساعتی کار روزانه و یا حتی اضافه کردن آیتم های بیشتر به بک لاگ اسپرینت باشد.

یکی دیگر از روش مقایسه الگوی اجرای اسپرینت با اسپرینت های انجام شده مشابه می باشد. در این روش نحوه تکمیل آیتم ها با و چگونگی پیشرفت آن با یک اسپرینت انجام شده مقایسه می شود. توجه شود که اسپرینت انجام می تواند یک اسپرینت خوب یا ضعیف باشد. در این حالت تنها نوع مقایسه تفاوت می کند. به طور کلی اگر اسپرینت مورد مقاسیه اسپرینت موفق بود تیم باید سعی کند که اسپرینت جاری مانند آن و یا بهتر اجرا شود. و اگر اسپرینت مذکور اسپرینتی موثر و رضایت بخش نبود از مشابهت الگوی آن با روند اجرایی اسپرینت جاری جلوگیری به عمل آید. باید همواره به یاد داشته باشیم که اتفاقات رخ داده در گذشته با احتمال زیاد در آینده هم ظهور خواهند داشت. بنابراین بهتر است از شرایط پیش آمده در گذشته و حال برای بهبود آینده استفاده کنیم(شکل 8-6).



شکل 8-6: مقایسه الگوی اجرای اسپرینت با اسپرینت های انجام شده

بخش دوم

فصل هفتم: معرفی TFS

یکی از مهمترین آیتم های هر متدولوژی ابزارها می باشند. اجرای یک متدولوژی بدون استفاده از ابزار می تواند مشکل و در مواردی غیر ممکن باشد. برای اسکرام نیز مانند بسیاری از متدولوژی مدیریت پروژه ابزارهایی وجود دارد. هر کدام از این ابزارها مزایا و معایب خاص خود را دارند. اما به طور کلی می توان آنها را به دو نوع فیزیکی و مجازی تقسیم بندی کرد. ابزارهای فیزیکی در حقیقت به ابزارهایی گفته می شود که از کامپیوتر بهره نمی برند. این گونه ابزار ها از نوشت افزارهایی چون کاغذ، وایت برد و کارت تشکیل می شوند. در مقابل این ابزارها، ابزارهای مجازی وجود دارد که به مدیریت متدولوژی به کمک کامپیوتر اشاره دارد. دراین نوع ابزارها کل اعمال مدیریتی در اجرا متدولوژی توسط نرم افزار صورت خواهد گرفت.

در مقام مقاسیه، ابزارهای فیزیکی نسبت به ابزارهای مجازی سنتی تر هستند. به عبارت دیگر می توان گفت ابزارهای فیزیکی عملاً از مزایای CASE بهره نمی برند. استفاده از این گونه ابزار اغلب زمان بر، ناکارآمد و پر هزینه است. اشتراک اطلاعات به وسیله این ابزارها بسیار ضعیف انجام می شود. ابزارهای فیزیکی همانطور که از نامشان پیدا است نیاز به یک محیط فیزیکی دارند و توسط آن محیط محدود می شوند. ردگیری پروژه توسط این ابزار بسیار ناکارآمد و کند صورت می گیرد.

اما در آن سو ابزارهای مجازی به واقع وضعیتی عکس ابزارهای فیزیکی را دارا هستند. آماده سازی این ابزار ارزان تر و راحت تر می باشد. از آنجایی که ابزارهای مجازی نرم افزار می باشند، می توانند با دیگر ابزار CASE در توسعه مانند ابزار زبان UML ارتباط بر قرار کنند. بدیهی است که اعمالی چون بایگانی، گزارش گیری و ردگیری از طریق نرم افزار آسان تر انجام خواهند پذیرفت. از طریق ابزار مجازی می توان به وسیله محیط هایی مثل اینترنت به اشتراک اطلاعات پرداخت و همچنین بسیاری از کارها را به صورت خودکار انجام داد. به طور کلی می توان گفت ابزارهای مجازی گزینه مناسب تری برای چالاک سازی توسعه نرم افزار می باشند. به علت برتری این ابزارها در ادامه کتاب صرفاً به معرفی این دسته خواهیم پرداخت.

یکی از کامل ترین ابزارهایی که تا به امروز برای اسکرام معرفی شده Team Foundation Server می باشد. TFS یک هسته عملیاتی مدیریت توسعه نرم افزار است که وظیفه آن یکپارچه سازی سایر نرم افزارهای مدیریت پروژه می باشد. عملیاتی که توسط TFS پشتیبانی می شوند عبارت اند از: مدیریت پروژه، دریگیری آیتم های کاری، کنترل نسخه، مدیریت تست کیس، خودکار سازی Buildها، گزارش گیری، مدیریت آزمایشگاه مجازی(Virtual Lab).

TFS خود شامل بخش های متنوعی می باشد اما آنچه را در این کتاب به آن خواهیم پرداخت آموزش اسکرام در TFS می باشد. TFS علاوه بر اسکرام از متدولوژی های دیگری چون MSF نیز پشتیبانی می کند. TFS ابزاری است بسیار کامل و مجهز جهت اجرای بی عیب و نقص اسکرام. هر آنچه در فصل های پیش راجع به اسکرام گفته شد در این بخش به صورت عملی ارائه می گردند. یادگیری TFS بسیار آسان است و مشاهده خواهد شد که استفاده از آن حتی بدون آموزش قبلی نیز میسر خواهد بود.

TFS داری ابزار و محیط مستقلی نمی باشد و محیط کاربری دیگر ابزارها برای اجرا استفاده می کند. سه از ابزارهایی که توسط آنها می توان از TFS استفاده نمود عبارت اند از: Visual Studio، Excel و Share Point. ابزار Share Point با استفاده از یک پورتال TFS را ارائه می دهد. به وسیله Excel نیز می توان با TFS همانند یک صفحه گسترده پیشرفته کار کرد. اما کامل ترین ابزاری که برای کار با TFS موجود است Visual Studio می باشد. از آنجایی که Visual Studio یک نرم افزار برنامه نویسی است بیشترین یکپارچگی را با TFS دارد. Visual Studio از طریق رابط کاربری خود محیط بسیار کارآمدی را برای کار با TFS در اختیار کاربران قرار می دهد. در این کتاب برای کار کردن با TFS ابزار Visual Studio انتخاب شده است و در ادامه آموزش ها مبتنی بر این بزار ارائه می شوند.

توجه شود که نیاز به نصب کامل Visual Studio وجود ندارد. برای اجرا شدن TFS در Visual Studio صرفاً نیاز به محیط Team Explorer می باشد. Visual Studio یکی از محصولات شرکت مایکروسافت می باشد و توسعه نرم

افزار توسط آن مبتنی بر پلت فرم های این شرکت صورت می گیرد. اما این موضوع دلیلی بر عدم توانایی توسعه دهندگان دیگر پلت فرم ها در استفاده از ترکیب TFS و Visual Studio نمی باشد. به عبارت دیگر هر تیم توسعه ای فارغ از نوع پلت فرم نرم افزار تحت توسعه خود می تواند برای مدیریت پروژه از ترکیب فوق استفاده نماید.

TFS از همه سیستم عامل های خانواده ویندوز به خوبی پشتیبانی می کند. TFS در دسته کلاینت ها روی ویندوز هایی چون XP، Vista و 7 قابل نصب است. گفتنی است که اگر TFS به صورت کلاینت نصب شود از برخی امکانات آن مثل گزارش گیری و یکپارچه سازی Share Point نمی توان استفاده کرد. در دسته سرور، TFS روی ویندوزهای 2003 و بالاتر قابل نصب است. اساساً استفاده اصلی TFS برای مدیریت متمرکز می باشد و بهتر است به جای کلاینت روی یک سرور نصب شود. با نصب TFS روی یک ویندوز سرور می توان به کل قابلیت های آن دست یافت.

پیش نیازهای نصب

IIS: پیش از نصب TFS ابتدا باید روی ویندوز IIS نصب شده باشد. نسخه IIS مورد نیاز حتماً باید 6 و بالاتر انتخاب شود. لازم به ذکر است که IIS در ویندوزهای سرور به صورت پیش فرض وجود دارد.

(Visual Stadio) Team Explorer: TFS برای مدیریت پروژه محیطی کاربری مجزایی ندارد و همانطور که گفته شد از Visual Stadio برای اجرا آن استفاده می شود. در صورت عدم تمایل به نصب Visual Studio می توان از Team Explorer به جای آن بهره برد. Team Explorer محیطی رایگان است و آن را می توان به راحتی از سایت مایروسافت دانلود نمود. همچنین عموماً در پکیج TFS یک نسخه از Team Explorer موجود است.

SQL Server 2008: TFS به علت استفاده از داده های زیاد و پیچیده نیاز به یک DBMS بسیار قوی دارد. برای این منظور می توان از یکی از نسخه های SQL SERVER استفاده نمود. برای مدیریت داده های TFS می توان از نسخه Express این نرم ابزار(SQL SERVER) استفاده کرد. این نسخه که به صورت رایگان عرضه می شود، شامل همه امکانات SQL SERVER نمی باشد اما برای راه اندازی TFS کافی است. باید توجه داشت که در نسخه Express قابلیت Reporting Srevice وجود ندارد در نتیجه نمی توان از قابلیت های گزارش گیری TFS استفاده نمود. برای استفاده از همه قابلیت های TFS باید از نسخه Standard و بالاتر استفاده شود. با نصب این نسخه بر روی یک ویندوز سرور، می توان به صورت کامل از همه امکانات TFS استفاده نمود.

Excel: یکی از ابزارهایی که می تواند به عنوان محیط کاری از آن استفاده کرد Excel می باشد. این نرم افزار یک برنامه مدیریت صفحه گسترده می باشد که درون مجموعه MS Office قرار دارد. حتی اگر از Visual Studio استفاده می شود، از Excel می توان به عنوان نرم افزاری جانبی جهت تحلیل داده و گزارش گیری استفاده کرد.

Share Piont: این ابزار یک فضای کاری است که در توسعه نرم افزار های سازمانی مورد استفاده قرار می گیرد. Share Point می تواند از طریق یک پروتال تحت وب مدیریت TFS را تسهیل بخشد. Share Point یکی از بهترین ابزارهایی است که برای مدیریت پروژه های بزرگ با تیم مختلف استفاده می شود. برای اجرای Share Point حتماً باید از ویندوز سرور استفاده شود.

تذکر: در صورتیکه از VS 2010 استفاده می شود، پیش از نصب TFS حتماً باید سرویس پک 1 آن نصب شود. این سرویس پک باعث می شود Visual Studio با TFS همخوانی بیشتری داشته باشد و کاراتر اجرا شود.

راهنمای نصب و راه اندازی TFS

اولین گام نصب TFS این است که فایل ISO آن از سایت مایکروسافت دانلود شود. نسخه ای دانلود می شود Trial می باشد و 90 روز اعتبار دارد. در صورت خرید لایسنس این محصول می توان به صورت نامحدودی از این نرم افزار استفاده نمود.

فایل ISO دانلود شده از سه پوشه مهم تشکیل می شود. اولین پوشه Team Explorer می باشد. در محتویات این پوشه فایل مورد نیاز برای نصب Team Explorer قرار داده شده است. با اجرا کردن فایل Visual Studio-ts.exe ویزارد نصب Team Explorer اجرا می شود.

در پکیج TFS دو نسخه TFS قرار دارد: نسخه 32 بیتی و نسخه 64 بیتی. پوشه ها هر یک فایل ها نصب این دو نسخه را به شکل مجزا در بر دارند. با اجرا کردن فایل Setup.exe موجود در یکی از این دو پوشه ویزارد نصب TFS اجرا می شود.

برای نصب TFS بهتر است مسیر نصب پیشنهادی پذیرفته شود. لازم به ذکر است که TFS برای نصب به حدود 800 MB فضای خالی نیاز دارد.

چهار گزینه هنگام نصب TFS وجود دارد که به طور پیش فرض همه آنها انتخاب شده اند(شکل 1-1):

Team Foundation Server: شامل تمامی کامپوننت های مود نیاز نصب یک نمونه جدید TFS می باشد.

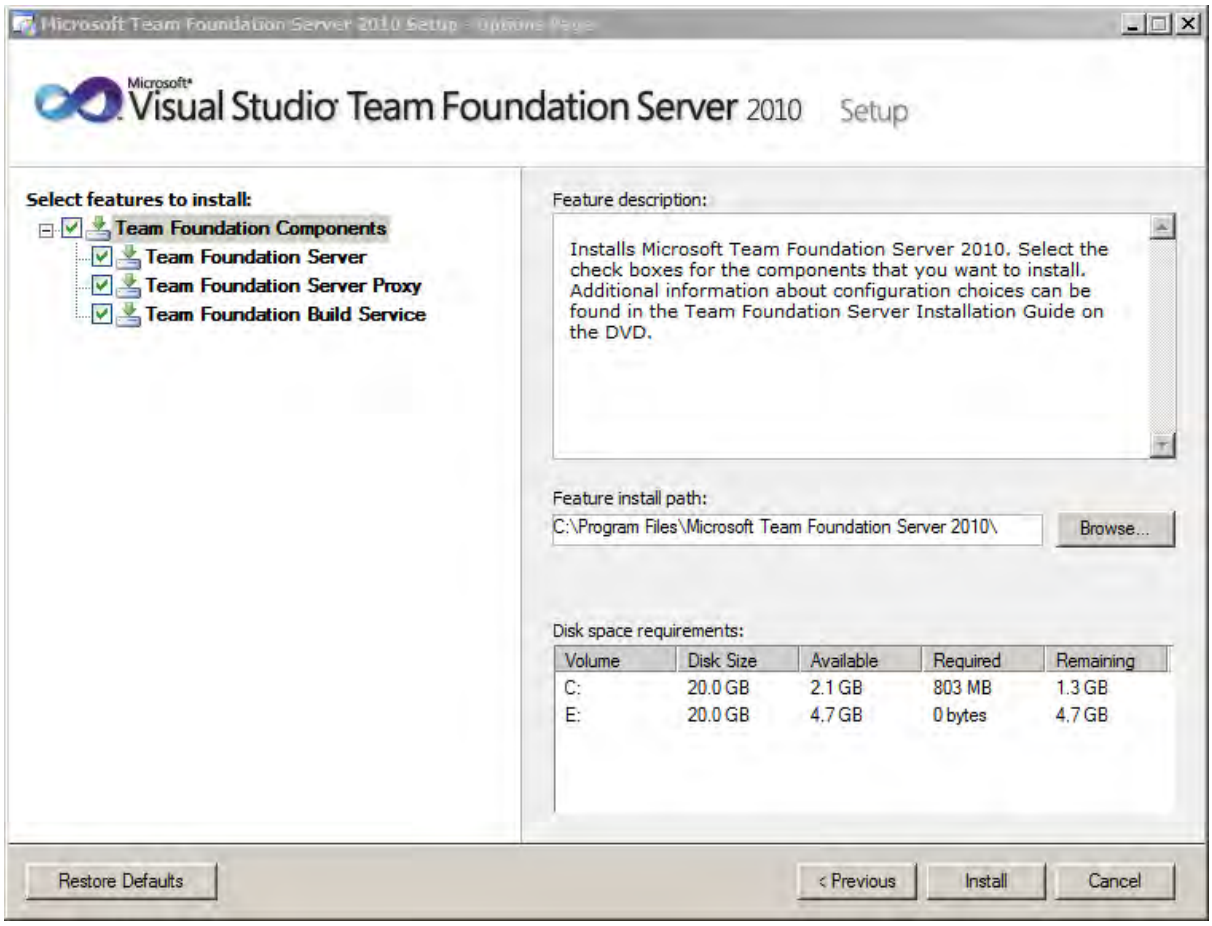
ExTeam Explorernsions for SharePoint Products and Technologies: اگر از Share Piont برای اجرای TFS استفاده می شود باید این زینه انتخاب شود. این گزینه شامل پیکربندی های Share Point بر روی سرور می باشد. اگر از سرور برای TFS استفاده نمی شود(حالت Local) نیازی به علامت دار بودن این گزینه نمی باشد.

Team Foundation Server Proxy: شامل کامپوننت هایی برای پیکربندی سرور ماشینی است که TFS روی آن نصب می شود.

Team Foundation Build Service: این گزینه برای نصب کامپوننت های پیکربندی ماشینی استفاده می شود که میزبانِ کنترل کننده Build در TFS می باشد.

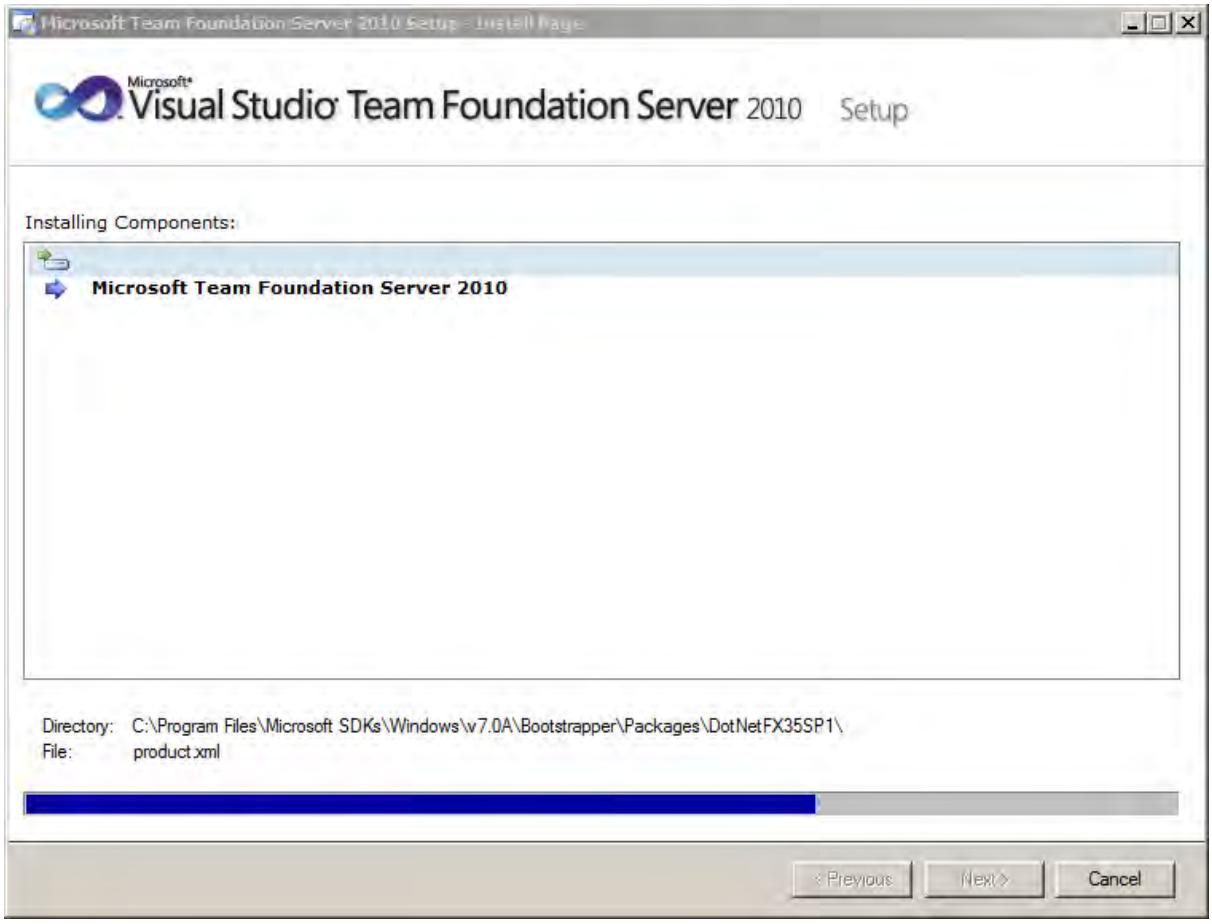
تذکر: اگر TFS روی یک ویندوز کلاینت مثل 7 نصب شود به علامت دار بودن ExTeam Explorernsions for SharePoint Products and Technologies و Team Foundation Server Proxy نمی باشد.

با زدن دکمه Install فرآیند نصب انجام می شود(شکل 1-7).



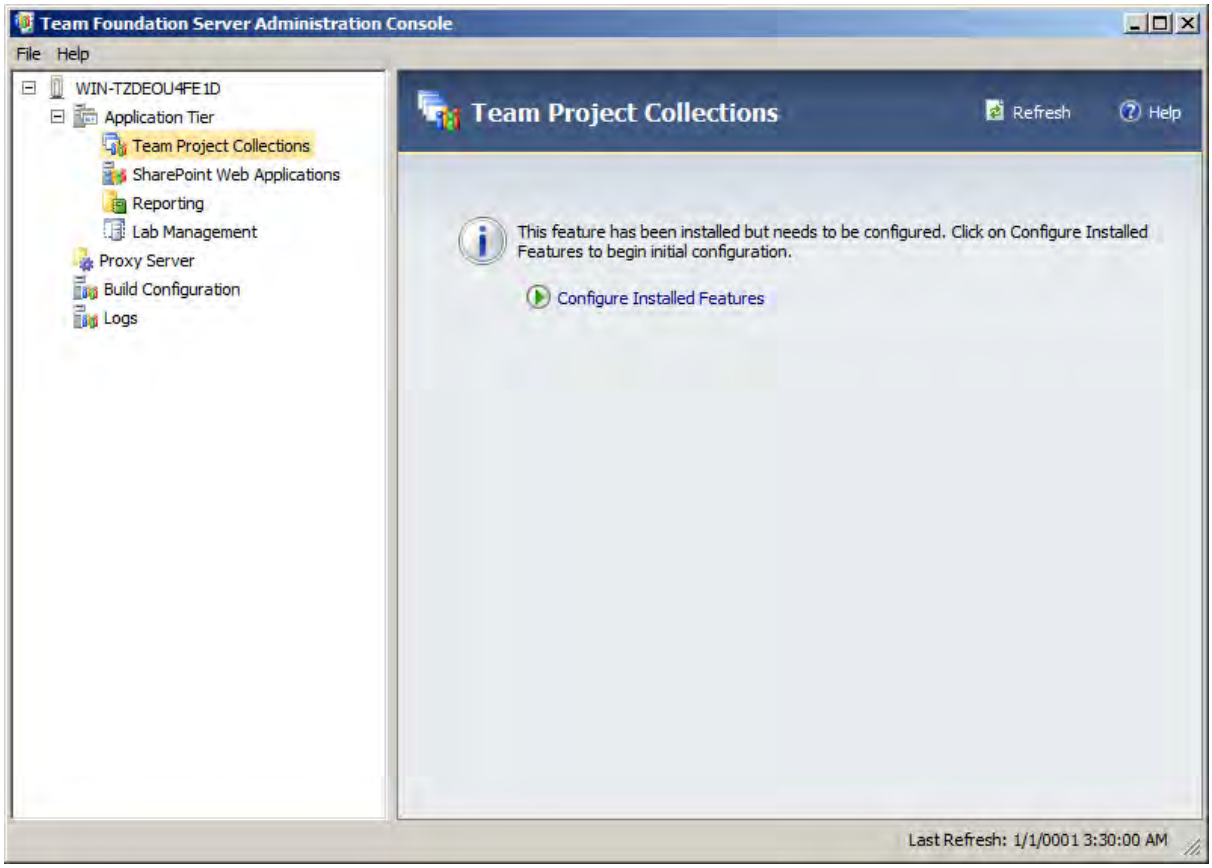
شکل 1-7: تنظیمات نصب TFS

پس از پایان یافتن نصب در صورتی که Net Framwork. بر روی سیستم عامل از پیش نصب نشده باشد سیستم یک بار Restart می شود(شکل 2-7).



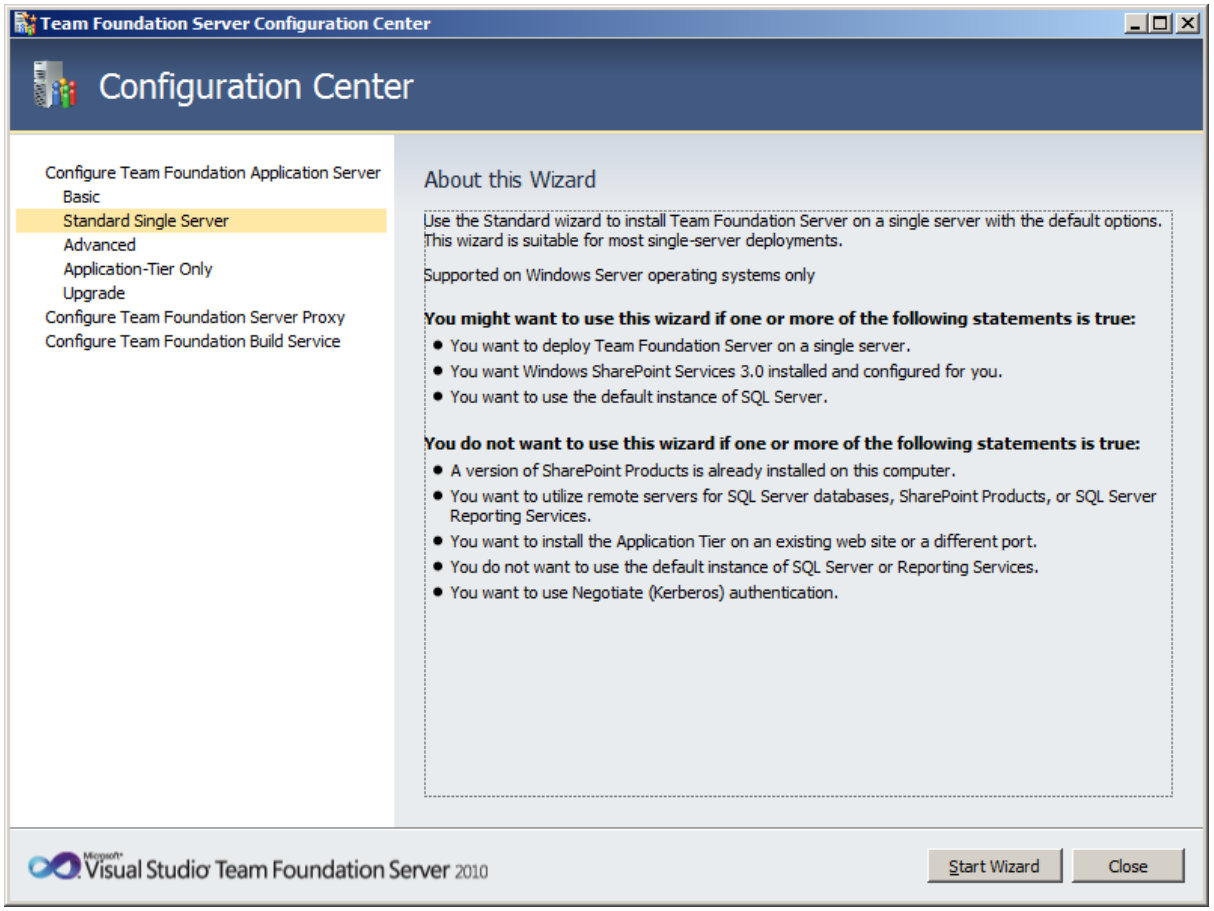
شکل 2-7: فرآیند نصب TFS

پس از بوت شدن مجدد سیستم وارد فاز پیکربندی TFS خواهیم شد. پیش از استفاده از TFS باید آن را پیکربندی نمود. برای اجرای ابزار پیکربندی TFS باید در مکان نصب TfsMgmt.exe را یافته و آن را اجرا نمود(شکل 3-7).



شکل 3-7: کنسول مدیریت TFS

این صفحه کنسول اصلی مدیریت و پیکربندی TFS می باشد. برای تنظیم اولین پیکربندی باید بر روی Team Project Collections کلیک شود و Configure Install Features انتخاب شود. با این عمل صفحه Configuration Center گشوده خواهد شد(شکل 4-7).



شکل 4-7: ویزارد پیکربندی TFS

هشت نوع پیکربندی مختلف در این پنجره وجود دارد:

Basic: این نوع سریع ترین و راحت ترین آماده سازی TFS برای اجرا می باشد. این نوع پیکربندی معمولاً برای تیم های کوچک استفاده می شود. در این نوع پیکربندی می توان همه قابلیت های TFS به جزء سرویس های گزارش گیری و یکپارچه سازی Share Point را استفاده کرد. دو قابلیت استثنا شده از طریق نصب TFS روی ویندوز سرور و استفاده از نسخه استاندارد SQL SERVER قابل استفاده می باشد.

Standard: در این نوع پیکربندی یک سرور برای اجرا آماده می کند. سایر ویژگی های این پیکربندی مانند Basic می باشد.

Advanced: این نوع حداکثر انعطاف پذیری را ارائه می دهد. اغلب قابلیت های آن مانند گزینه Standard با این تفاوت که Advanced اجازه تعریف یک سیستم Remote را برای Share Point، SQL SERVER و SQL SERVER Reporting Service را می دهد. همچنین این ویزارد امکان پیکربندی Kerberos Authertication برای استفاده از سایر نسخه Share Point ارائه می دهد، از دیگر امکانات این پیکربندی امکان غیر فعال کردن Reporting Service و Share Point InTeam ExplorergraTeam Explorerd می باشد.

Application-Tire Only: از این نوع پیکربندی برای زمانی که می خواهیم یک TFS یا یک سرور را تغییر دهیم استفاده می شوند. همچنین از آن می توان برای اجرای سناریوی DesasTeam Explorerr-Recovery استفاده شود.

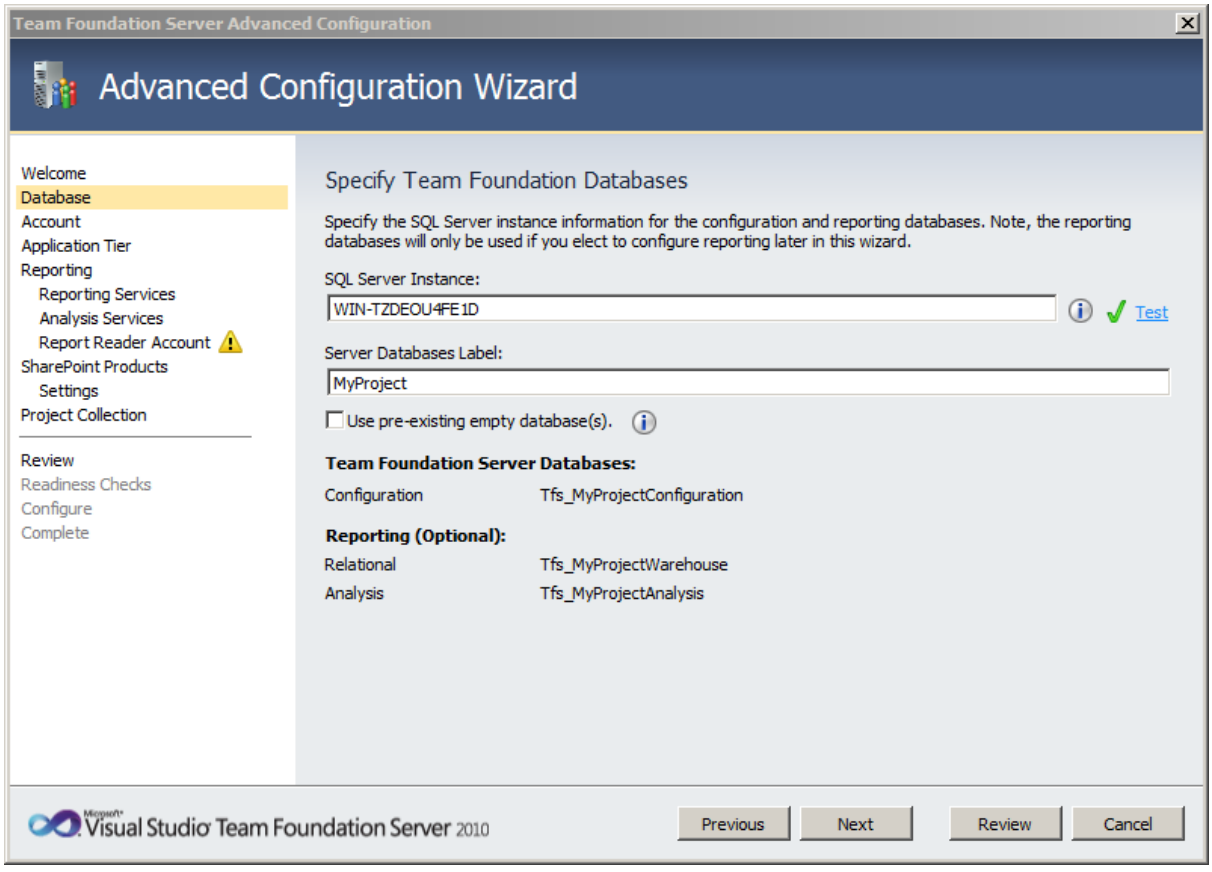
Upgrade: از این گزینه برای به روز رسانی TFS 2005 یا TFS 2008 استفاده می شود.

Configure Team Foundation Server Proxy: از این گزینه برای پیکربندی ماشین به عنوان یک Team Foundation Server Proxy استفاده می شود.

Configure Team Foundation Build Service: از این گزینه هنگامی استفاده می شود که بخواهیم ماشین را به عنوان یک Build Control پیکربندی کنیم.

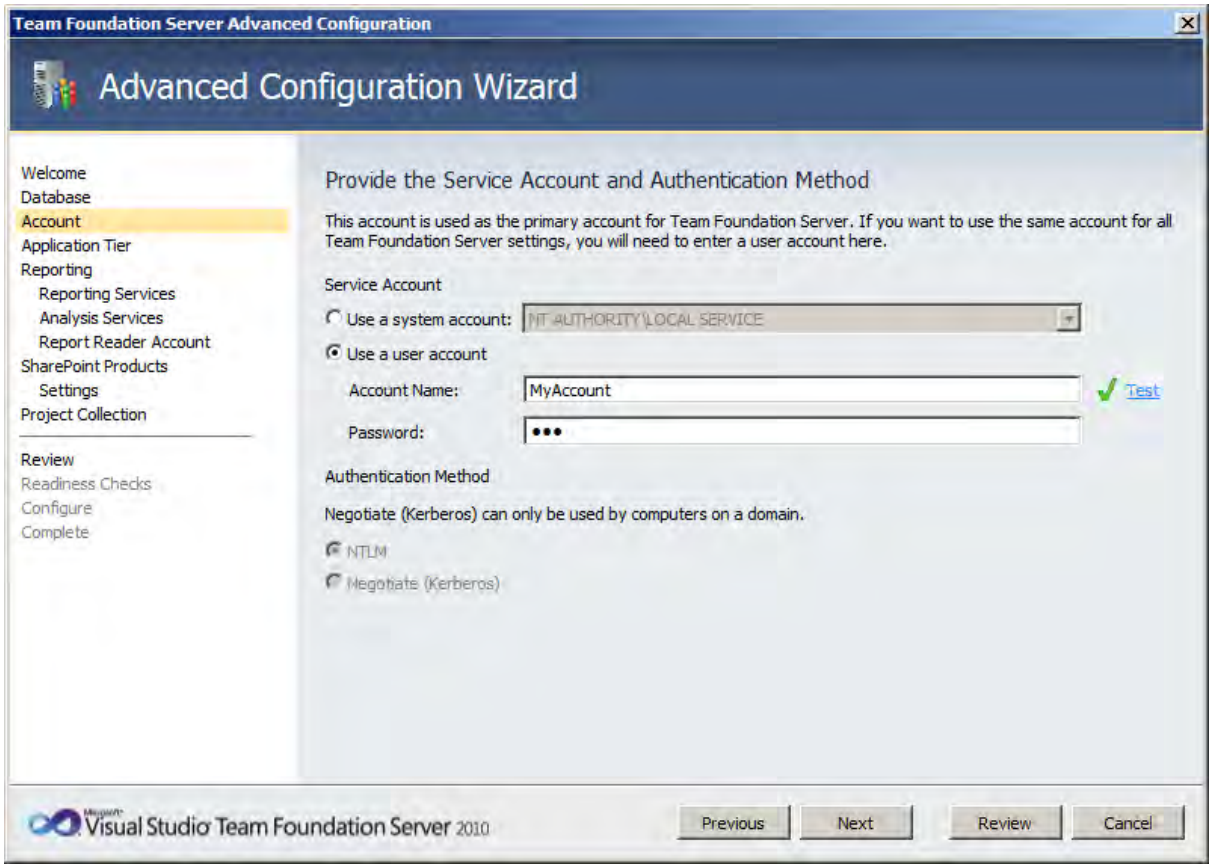
Configure ExTeam Explorerntion for Share Point Product: این گزینه نیز هنگامی استفاده می شود که قصد پیکربندی TFS را برای Share Point InTeam ExplorergraTeam Explorerd داشته باشیم.

برای استفاده از همه قابلیت ها در TFS بهتر است از گزینه Advaned استفاده شود. کلیک کردن روی Start Wizard باعث باز شدن پنجره Advanced Configuration Wizard می شود. این ویزارد از تعدادی مرحله تشکیل شده است. صفحه آغازین صفحه Wellcome می باشد که اطلاعاتی در خصوص ویزارد را ارائه می دهد. با کلیک روی دکمه Next وارد بخش Database می شویم. این بخش شامل دو قسمت می باشد. بخش اول Instance را مشخص می کند. اگر نسخه از SQL SERVER از پیش نصب شده باشد، نام Instance آن به صورت خودکار در این قسمت قرار می گیرد. اگر نام Instance به صورت دستی وارد شود با استفاده از دکمه Test می توان صحت آن را آزمایش کرد. بخش دوم یک برچسب به عنوان نام دیتابیسی که قرار است برای TFS ساخته شود اخذ می گردد. این برچسب نباید قبلاً استفاده شده باشد و باید یکتا باشد. اگر گزینه Use pre-existing empty database(s) علامت خورده باشد، TFS از یک دیتابیس خالی که قبلاً ساخته شده است استفاده می کند. در پایین این صفحه نیز اطلاعاتی چون نام واقعی دیتابیس نمایش داده می شود(شکل 5-7).



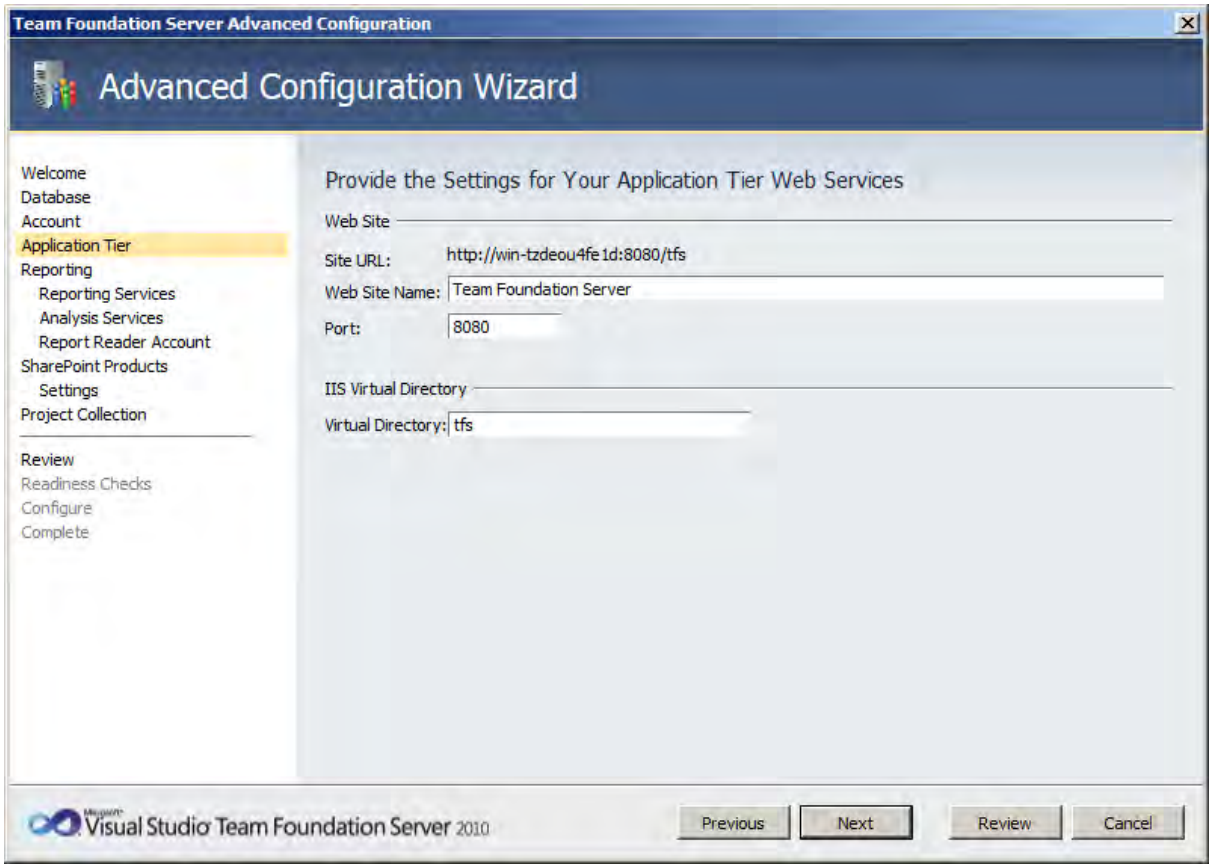
شکل 5-7: بخش معرفی دیتابیس در ویزارد پیکربندی TFS

با کلیک روی دکمه Next وارد مرحله بعد یعنی تعیین Acount می شویم. این حساب به عنوان حساب اصلی TFS مورد استفاده قرار می گیرد. برای تعیین این حساب هم می توان از حساب های سیستمی استفاده کرد و هم حساب های کاربری. اگر یک حساب کاربری معرفی شود می توان از آن برای همه تنظیمات TFS استفاده کرد. در غیر این صورت در بخش های دیگر نیاز به معرفی یک حساب کاربری جداگانه می باشد. با استفاده از دکمه Test می توان صحت حساب و رمز عبور آن را پیش از رفتن به بخش های دیگر چک کرد(شکل 6-7).



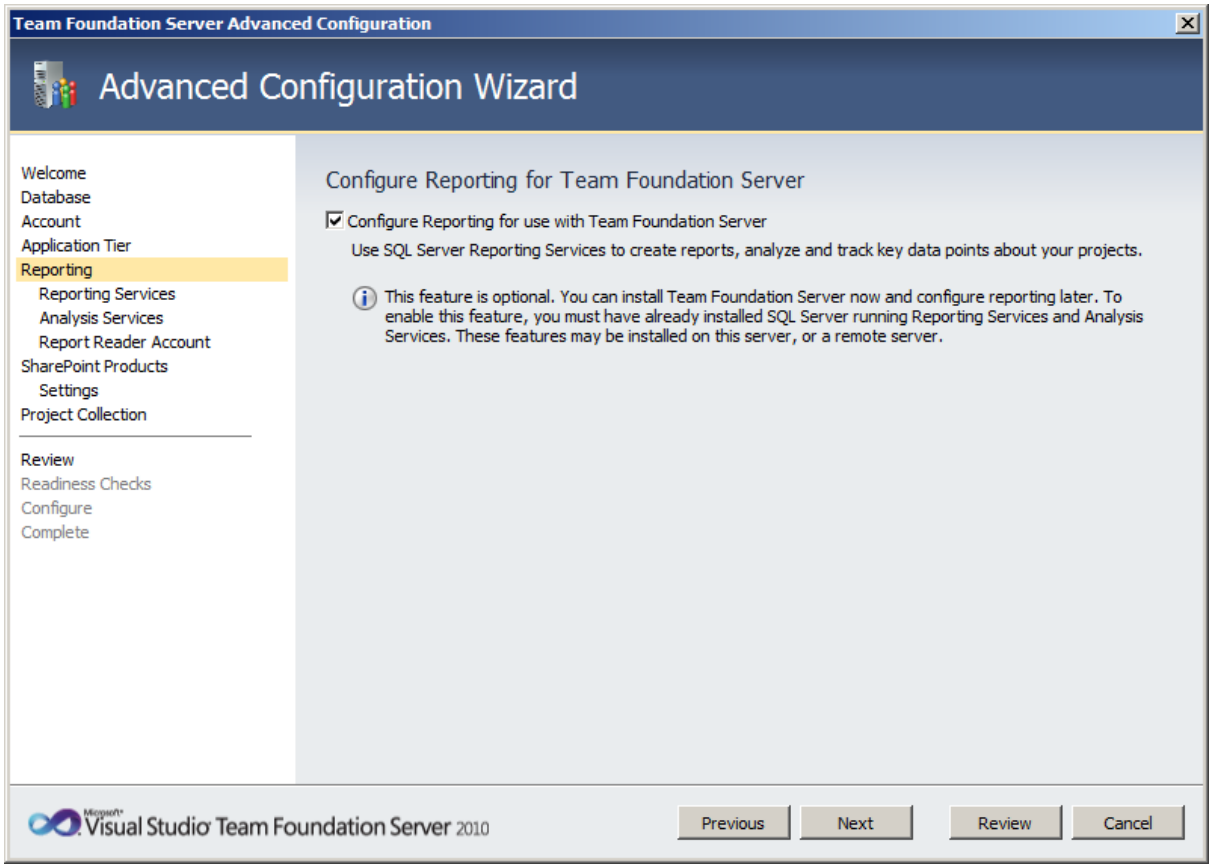
شکل 6-7: بخش معرفی اکانت در ویزارد پیکربندی TFS

مرحله بعد Application Tier نام دارد. در این بخش اطلاعات لازم برای وب سرویس Application Tier تعیین می گردد. این اطلاعات از قبیل نام وب سایت، پورت و مسیر مجازی IIS می باشد. توصیه می شود مقادیر پیش فرض این اطلاعات به همان صورتی که هستند پذیرفته شود(شکل 7-7).



شکل 7-7: بخش اپلیکیشن سرویس وب در ویزارد پیکربندی TFS

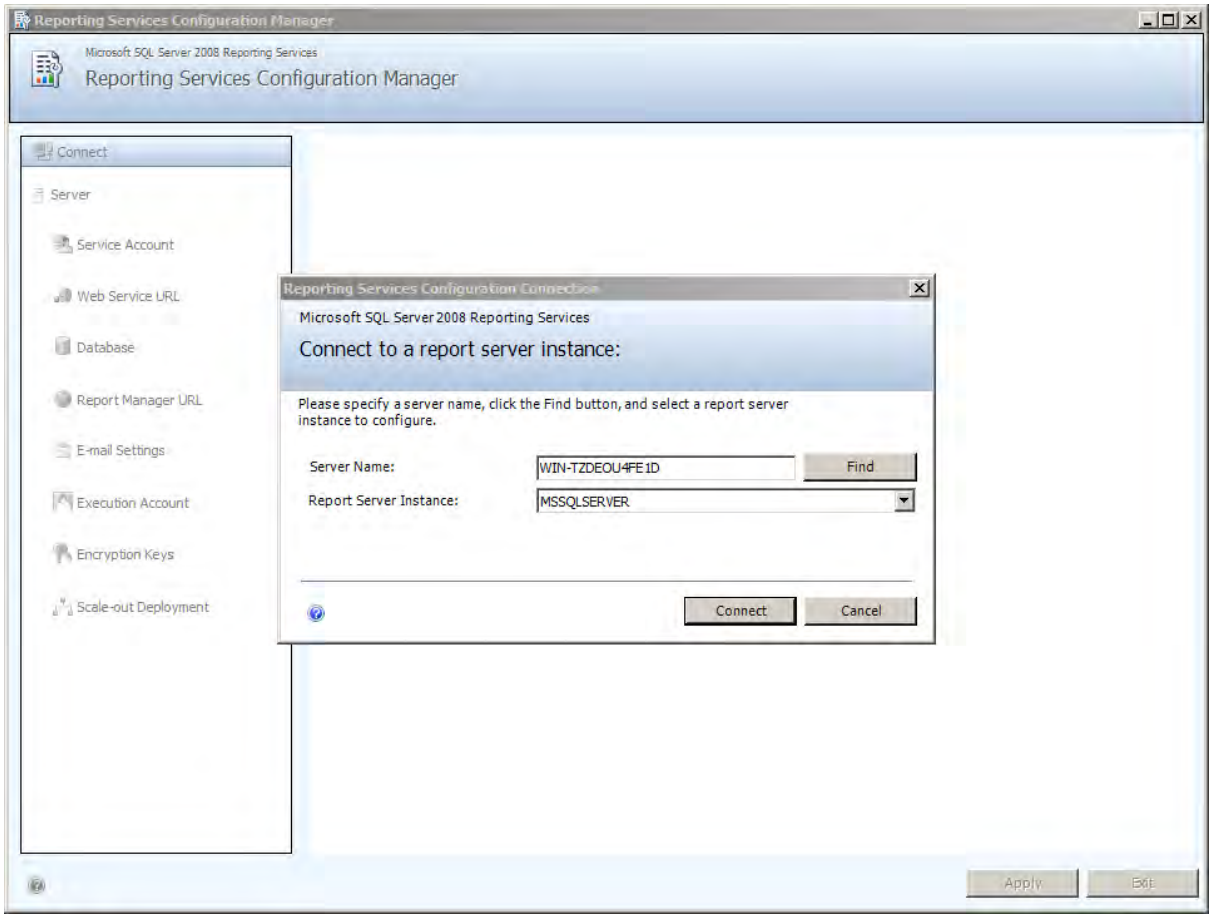
مرحله بعد مربوط به تنظیمات گزارش گیری می باشد. در بخش Report می توان تعیین نمود که قابلیت گزارش گیری فعال باشد یا خیر(شکل 8-7).



شکل 8-7: بخش گزارش گیری در ویزارد پیکربندی TFS

برای فعال کردن این قابلیت باید گزینه Configure Reporting for use with Team Foundation Sever علامت بخورد. با این عمل سه زیر بخش مربوط به تنظیمات قسمت گزارش گیری نمایش داده می شود.

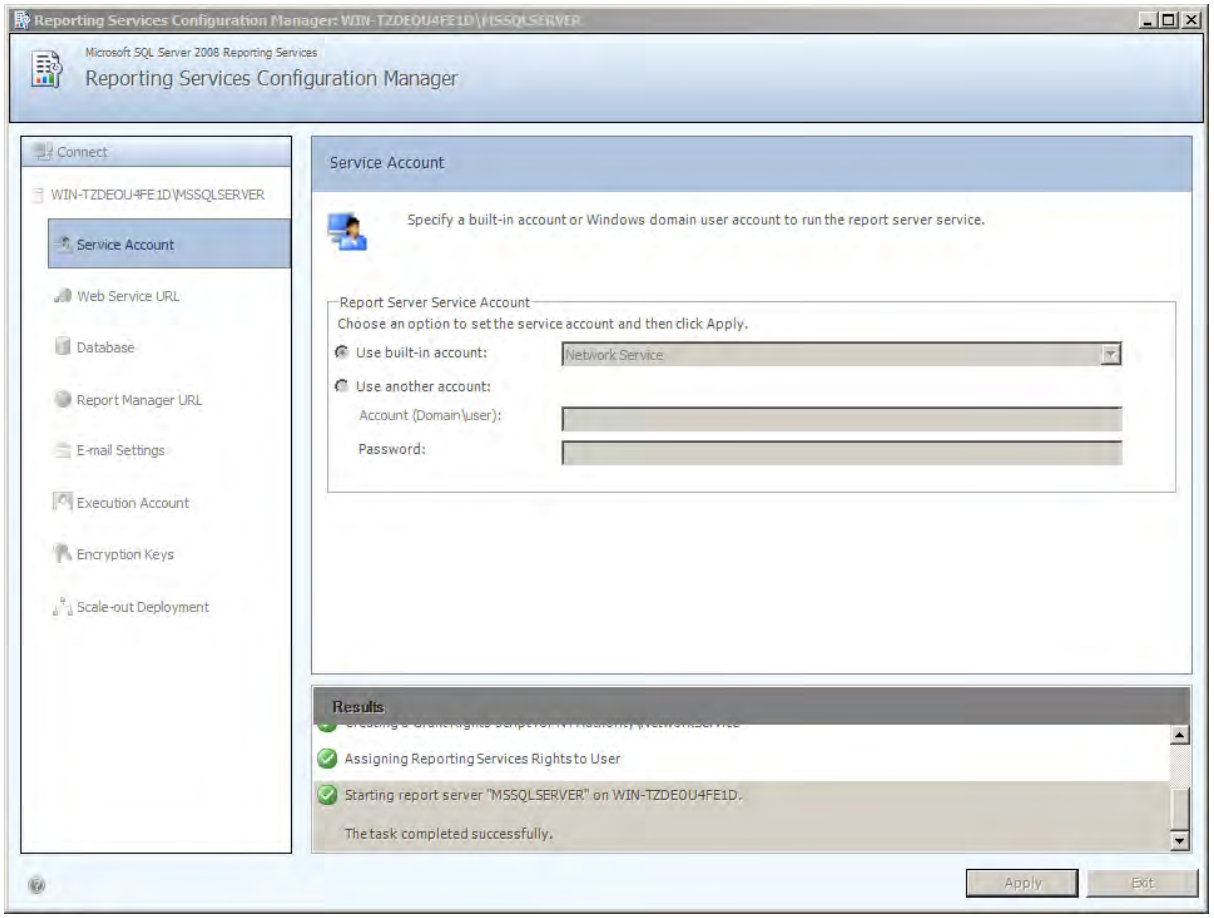
برای ادامه اعمال تنظیمات گزارش گیری ابتدا باید برخی پیکربندی ها در Reporting Service Configuration Management(Reporting Services Configuration Manager) مرتبط با SQL SERVER انجام شود. با اجرای Reporting Service Configuration Managment از محل نصب SQL SERVER صفحه Connection آن باز می شود(شکل 9-7).



شکل 9-7: اتصال به مدیریت پیکربندی سرویس گزارش گیری

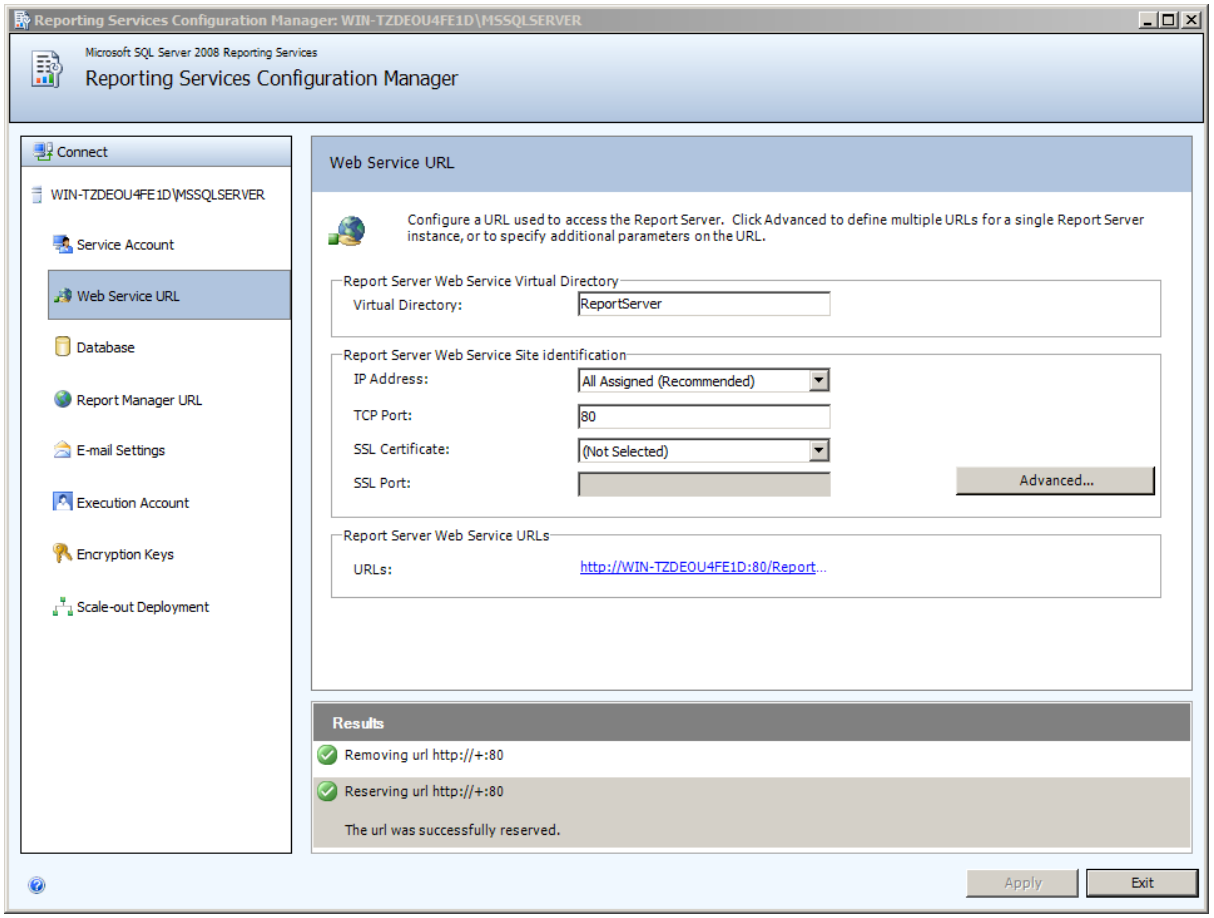
این صفحه از کاربر می خواهد که نام سرور و نام Instance سرور گزارش گیری را وارد کند. پس وارد کردن اطلاعات خواسته شده با زدن کلید Connect وارد بخش اصلی Reporting Service Configuration Managment می شویم. به صورت پیش فرض اطلاعات موجود در Reporting Services Configuration Manager اعمال نشده هستند و باید به صورت دستی این بخش ها را Apply نمود.

در بخش Service Acount حسابی که سرویس گزارش گیری استفاده می کند تعیین می گردد. این حساب می تواند سیستمی و یا کاربری باشد. پس تعیین حساب حتماً باید کلید Apply کلیک شود. لازم به ذکر است که این کلید برای هر بخش به صورت جداگانه عمل می کند(شکل 10-7).



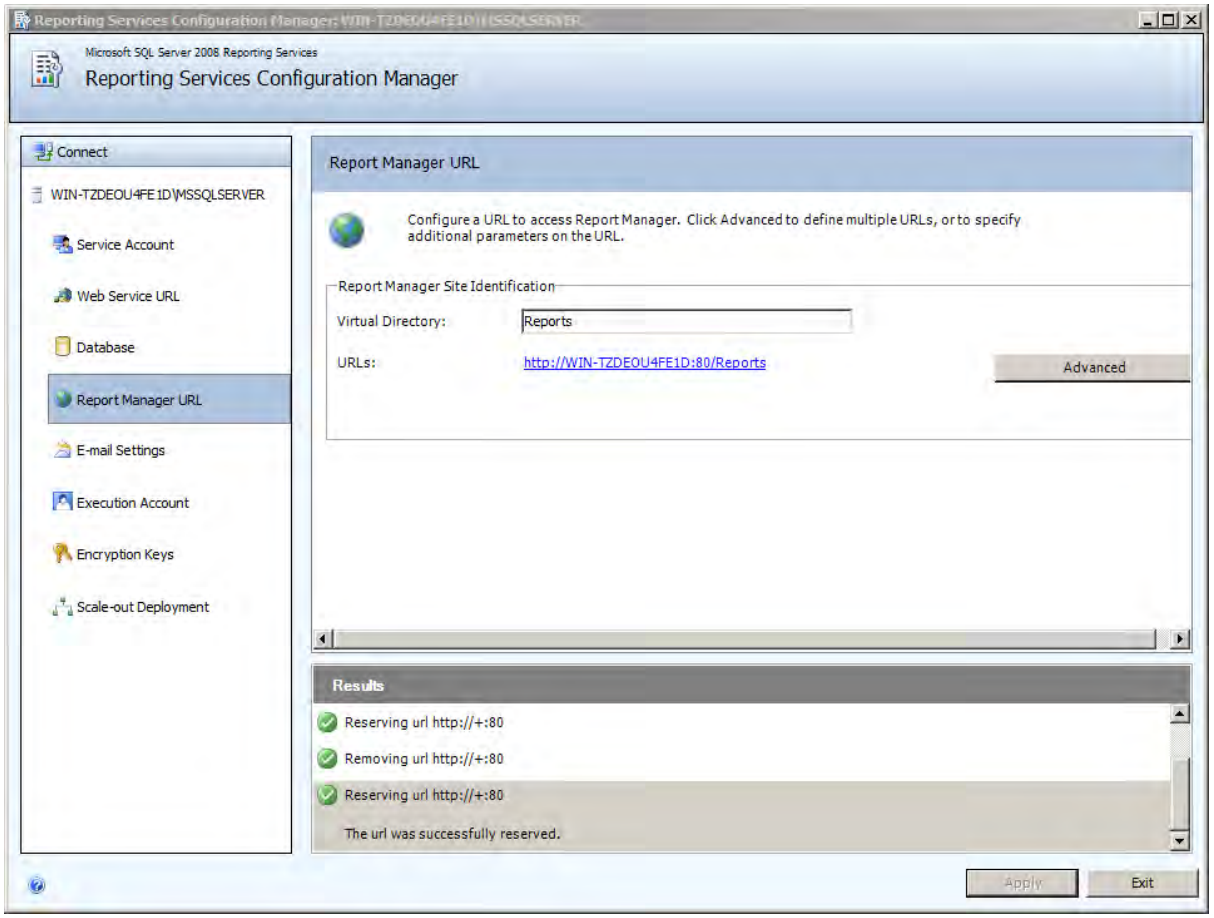
شکل 10-7: معرفی اکانت سرویس به سرویس گزارش گیری

در بخش Web Service URL مسیر مجازی Report Server تعیین می شود. همچنین باید اطلاعات تعیین هویت وب سرویس گزارش گیری نیز مشخص شود. برای این امر لازم است آدرس IP و پورت TCP مشخص شود. بهتر است برای این امر از گزینه اول Address IP یعنی All Assigned استفاده شود. همچنین بهتر است برای پورت TCP نیز مقدار 80 انتخاب شود. به خاطر داشته باشید که در پایان حتماً باید روی دکمه Apply کلیک شود(شکل 7-11).



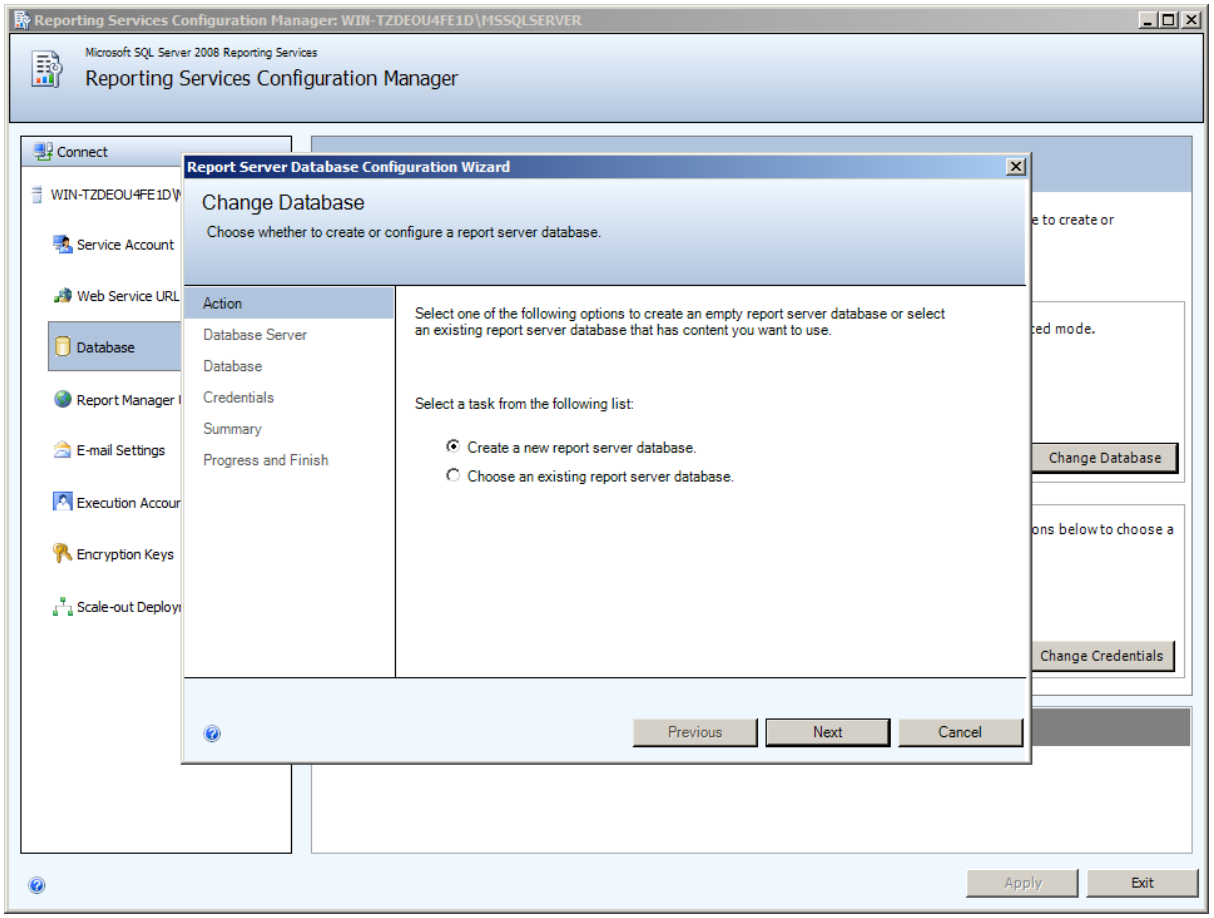
شکل 11-7: تنظیمات Web Service URL در سرویس گزارش گیری

برای بخش Report Manager URL نیز باید یک مسیر مجازی برای Report Manager تعیین شود. با Applyکردن این بخش مسیر مجازی در بخش URL به صورت لینک به نمایش در خواهد آمد(شکل 12-7).



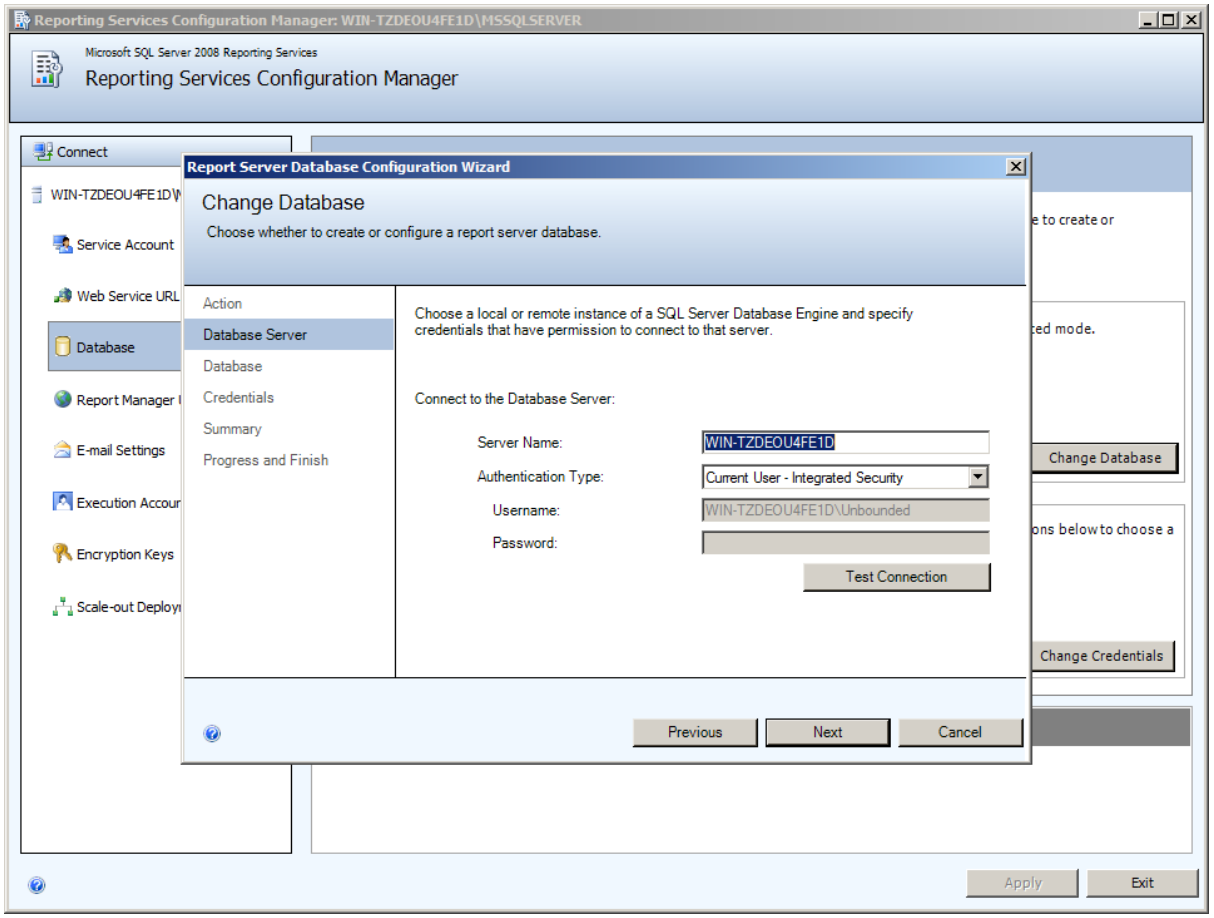
شکل 12-7: تنظیمات Manager URL در سرویس گزارش گیری

هر کدام از بخش های تنظیم شده در Reporting Services Configuration Manager مربوط به یک بخش از تنظیمات گزارش گیری TFS می باشد. ممکن است پس از اعمال همه تنظیمات در Reporting Services Configuration Manager و TFS باز هم در بخش پیکربندی برای دسترسی به وب سرویس مشکلی ایجاد شود. در این حال باید دیتابیس سرور گزارش گیری عوض(Change) نمود. برای این کار باید در بخش Database از Reporting Services Configuration Manager صورت پذیرد. کافی است روی دکمه Change Database از Current Report Server Database کلیک شود. این عمل باعث باز شده ویزارد Database Change خواهد شد. در صفحه اول این ویزارد تعیین می شود که دیتابیسی جدید ساخته شود یا از دیتابیس های موجود استفاده شود. با توجه به مشکل یاد شده بهتر دیتابیس جدید ساخته شود(شکل 13-7).



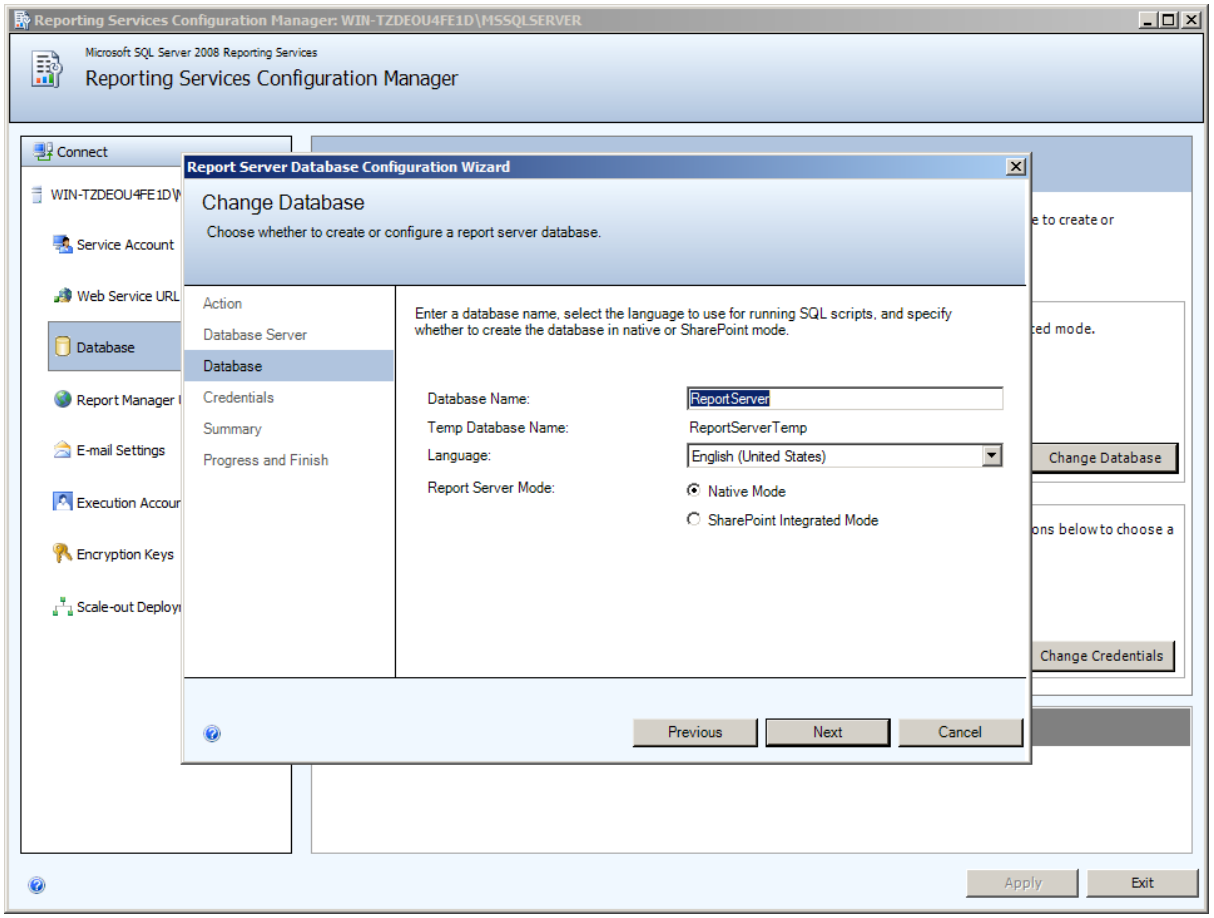
شکل 13-7: ویزارد ایجاد دیتابیس در سرویس گزارش گیری

با انتخاب گزینه اول و کلیک روی دکمه Next به مرحله بعد یعنی Database Server خواهیم رفت. در این بخش نام سرور و نوع تشخیص هویت تعیین می شود(شکل 14-7).



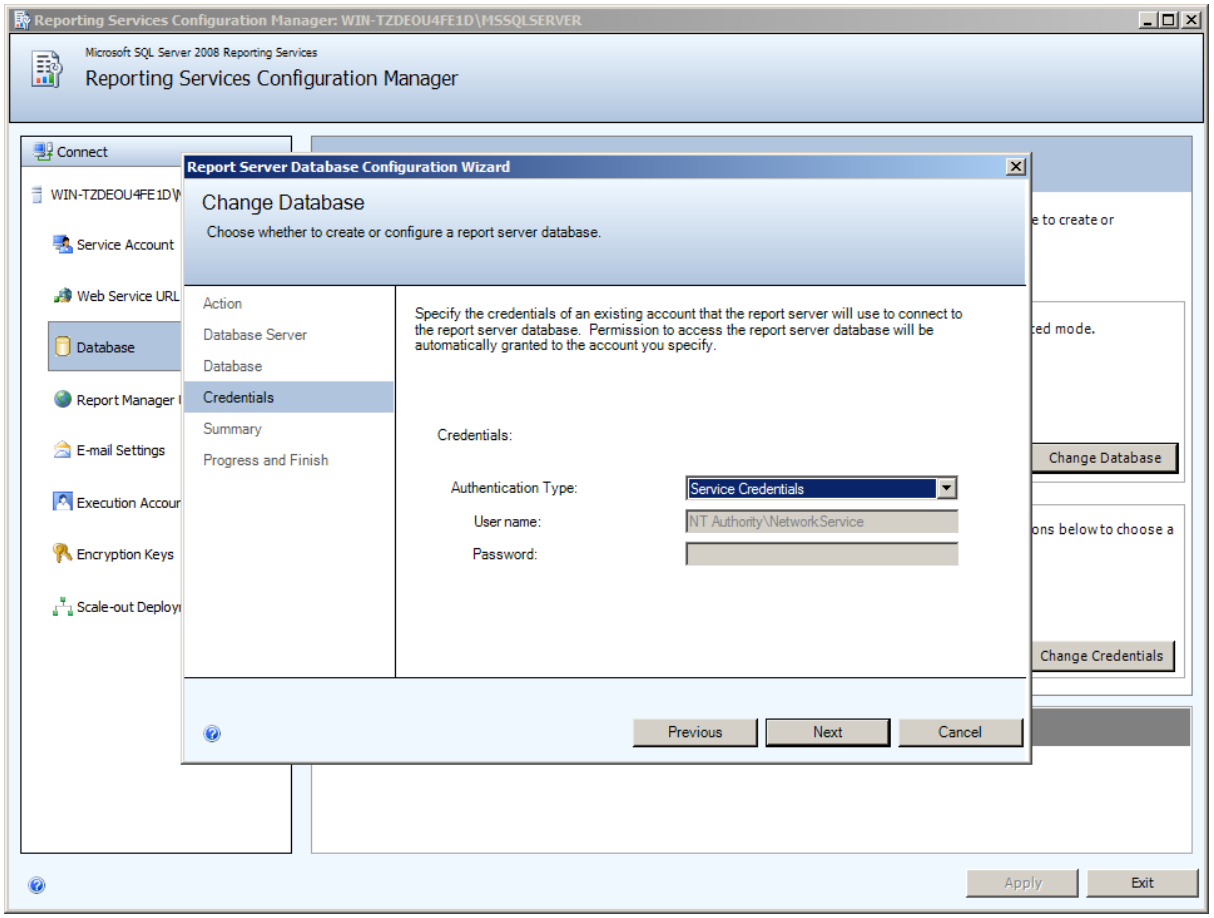
شکل 14-7: ورود نام سرور سرویس گزارش گیری

مرحله بعد در رابطه با اطلاعات خود دیتابیس نظر نام آن و زبان مورد استفاده آن از کاربر پرسیده می شود(شکل 15-7).



شکل 15-7: ورود نام دیتابیس جدید سرویس گزارش گیری

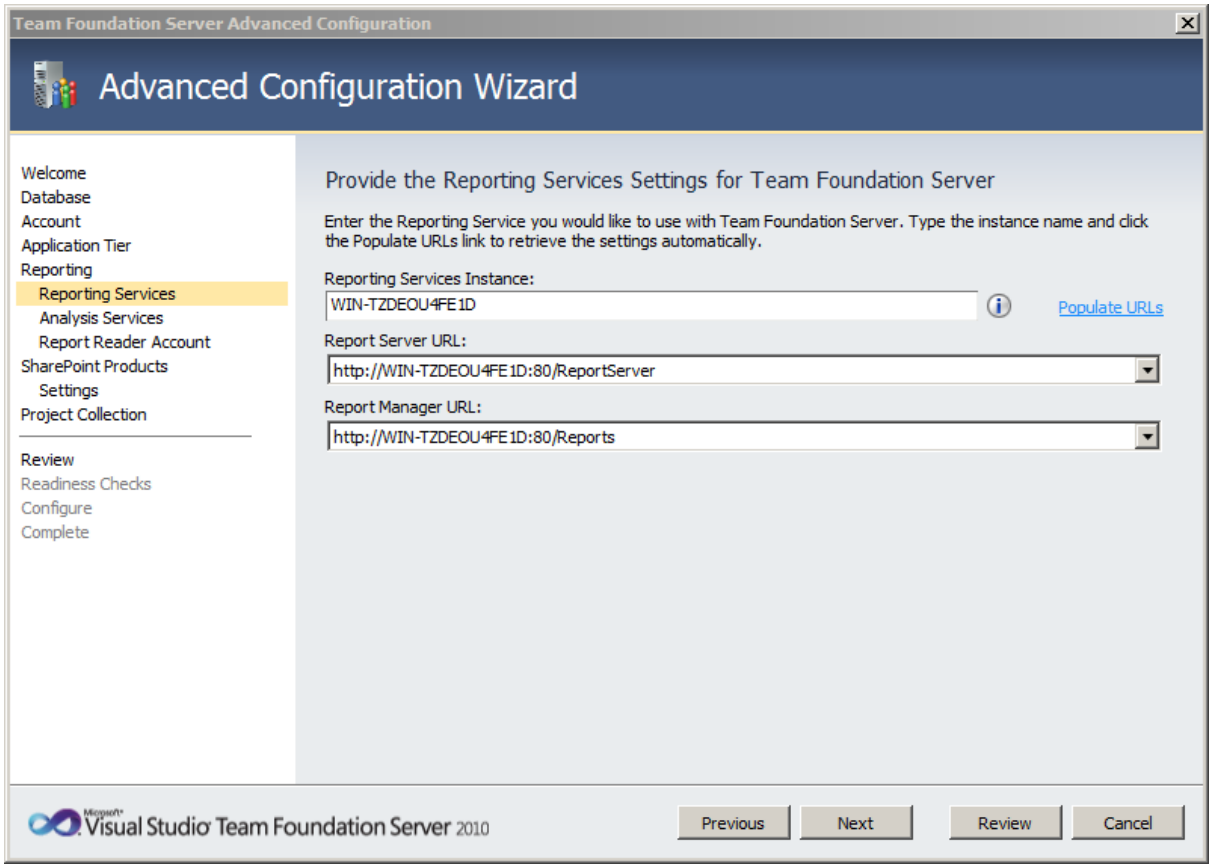
مرحله بعد نیز اطلاعاتی در رابطه با Credentials درخواست می شود که بهتر است بدون تغییر رها شود(شکل 16-7).



شکل 16-7: تنظیمات اعتبارسنجی سرویس گزارش گیری

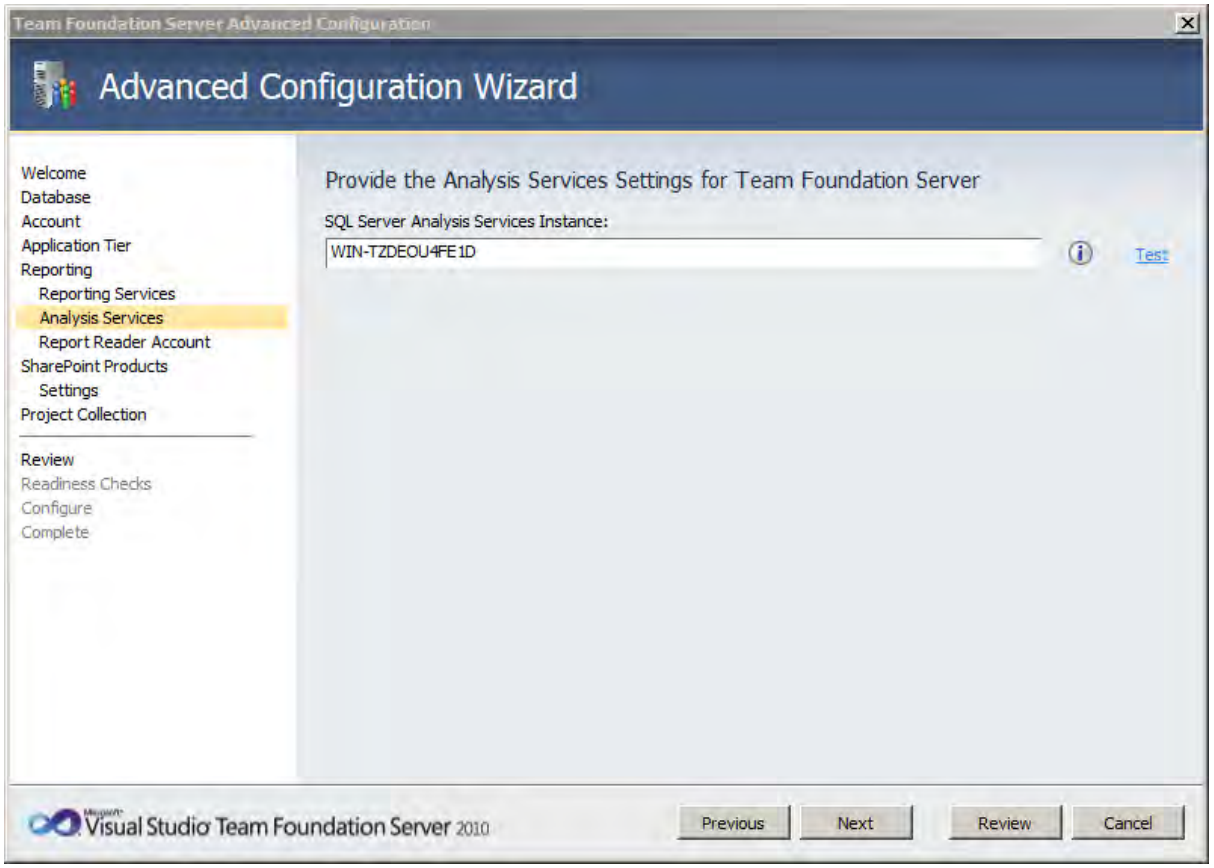
در مرحله آخر نیز خلاصه از تنظیمات تعیین شده نمایش داده می شود. با کلیک روی Next عملیات ایجاد دیتابیس جدید انجام می شود. به خاطر داشته باشید در این بخش(Database) نیاز به Apply کردن تنظیمات وجود ندارد. سایر تنظیمات موجود در Reporting Services Configuration Manager ارتباطی با پیکربندی TFS ندارند.

پس از اعمال تنظیمات در Reporting Services Configuration Manager می توان پیکربندی بخش گزارش گیری TFS را بدون مشکل دنبال کرد. بخش Reporting Service برای مهیا کردن سرویس گزارش گیری برای TFS باید پیکربندی شود. این بخش دارای سه فیلد است. فیلد اول سرور گزارش گیری را مشخص می کند. در صورت نصب Reporting Service در SQL SERVER این فیلد به صورت خودکار مقدار دهی می شود. دو فیلد بعدی Report Server URL و Report Manager URL را مشخص می کنند. این دو فیلد نیاز به اعمال تنظیمات در Reporting Services Configuration Manager دارند. به صورت پیش فرض هر دو فیلد بدون مقدار هستند. برای مقدار دهی این فیلد ها باید بر روی PopulaTeam Explorer URLs کلیک شود. باید این عمل به صورت خودکار فیلدها مقدار دهی می شوند(شکل 17-7).



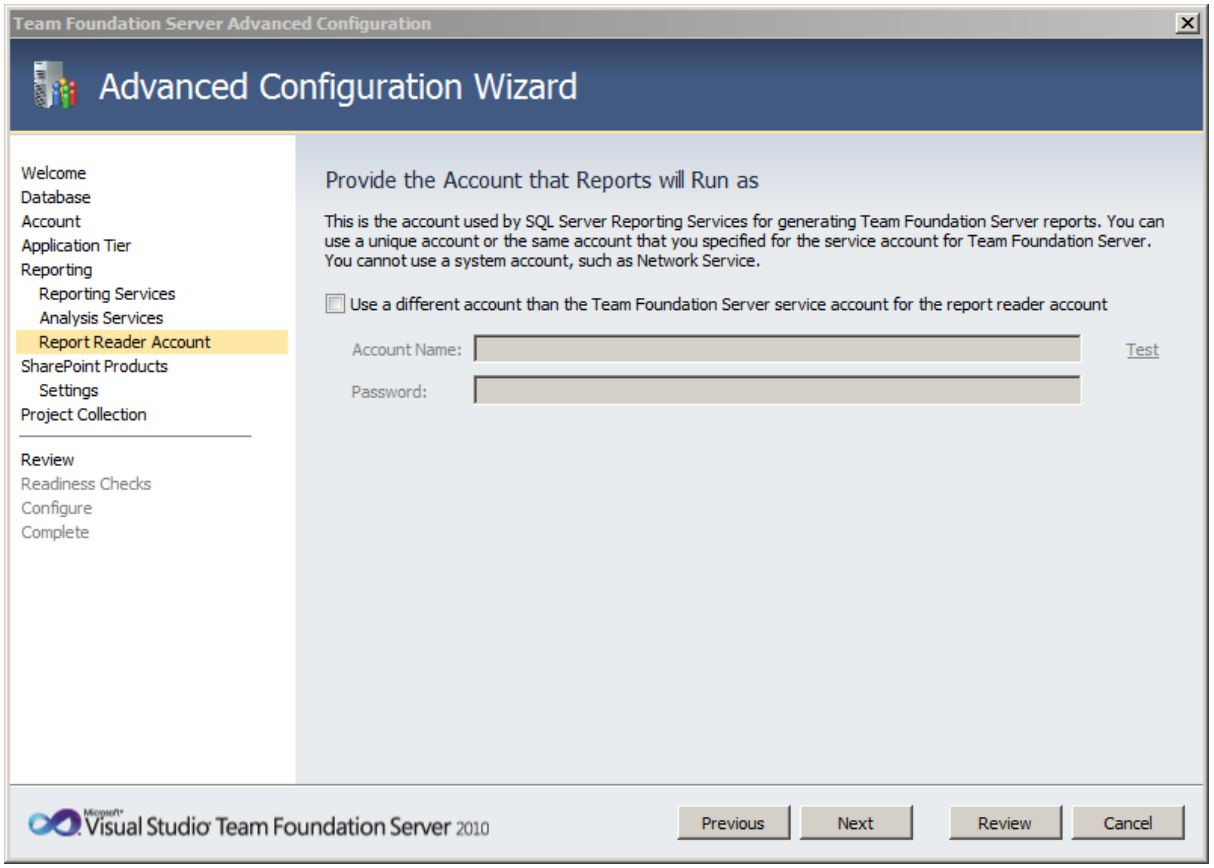
شکل 17-7: تنظیمات سرویس گزارش گیری

بخش بعدی Analysis Service می باشد که صرفاً Instance مربوط به SQL SERVER Analysis Service را طلب می کند. این فیلد نیز در صورت نصب Analysis Service به صورت پیش فرض مقدار دهی می شود(شکل 18-7).



شکل 18-7: تنظیمات سرویس آنالیز در ویزارد پیکربندی TFS

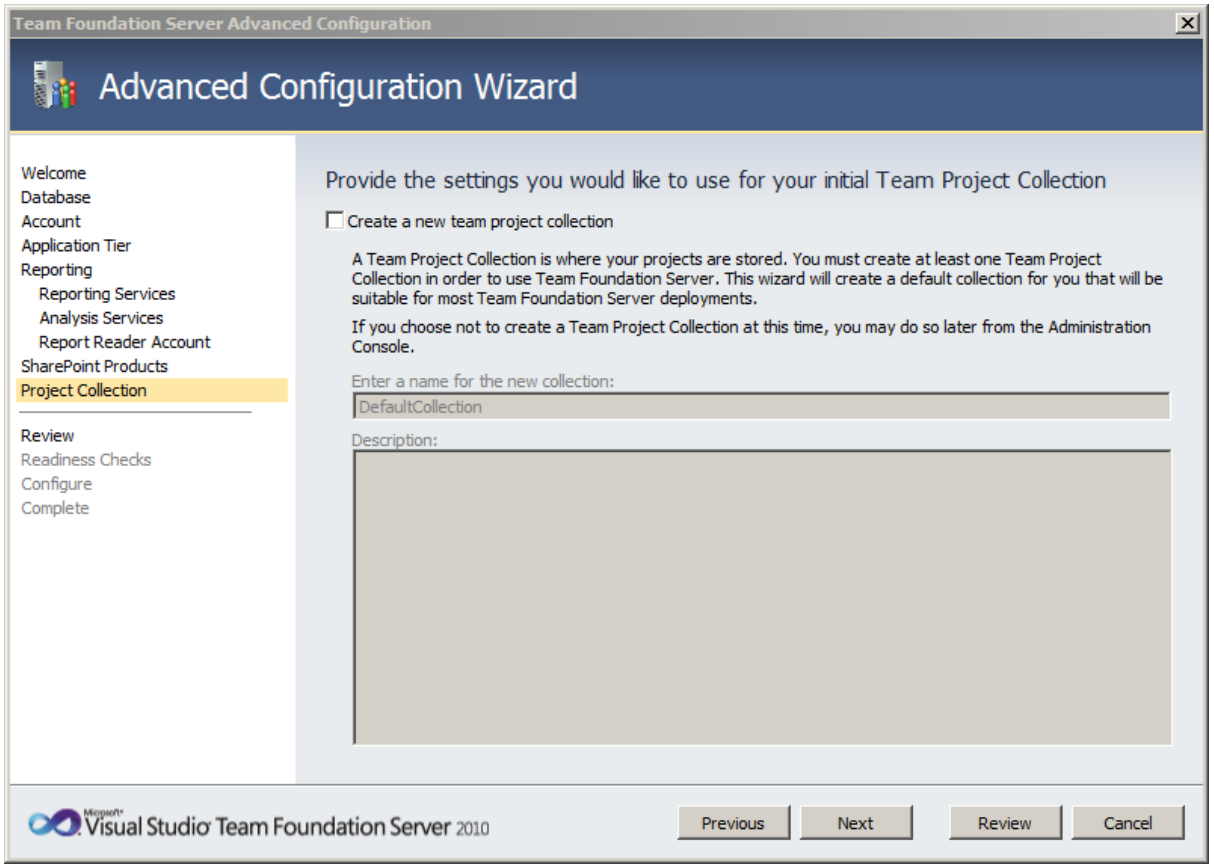
در بخش Report Reader Account حسابی باید معرفی شود که SQL SERVER از طریق آن بتواند به TFS دسترسی پیدا کند. این حساب حتماً باید کاربری باشد و نمی توان از حساب سیستمی برای این منظور استفاده کرد. اگر در بخش Account از یک حساب کاربری استفاده شده باشد نیازی به معرفی حسابی جدید برای این بخش نمی باشد و از همان حساب استفاده می شود. با علامت دار کردن Use a different account می توان حسابی متفاوت از حسابی که در بخش Account تعیین شد و معرفی شود(شکل 19-7).



شکل 19-7: تنظیمات اکانت خواننده گزارش در ویزارد پیکربندی TFS

بخش بعدی مربوط به فعال سازی قابلیت Share Point در TFS می باشد. اگر گزینه Configure SharePoint for use with Team Foundation Server علامت داشته باشد ضمن فعال سازی SharePoint مرحله ای برای تنظمات آن نیز اضافه می شود. لازم به ذکر است که برای تنظیم کردن این بخش نیاز است پیش از آن SharePoint نصب شده باشد. بخش تنظیمات(Setting) شامل دو قسمت است. قسمت اول هنگامی استفاده می شود که SharePoint نصب نباشد و برای نصب آن بخواهیم از Farm Mode استفاده نماییم. این گزینه نیاز به معرفی یک حساب کاربری دارد. اگر در بخش Account یک حساب کاربری معرفی شده باشد نیازی به معرفی مجدد آن در این بخش نمی باشد. قسمت دوم برای وقتی استفاده می شود که سرویس SharePoint از قبل نصب شده باشد. در این قسمت نیاز است تا SiTeam Explorer URL و Administrator URL سرور مورد نظر معرفی شوند. گفتی است که از آن جایی که در ادامه از سرویس SharePoint استفاده نخواهیم کرد و گزینه پیکربندی آن را بدون علامت رها می کنیم.

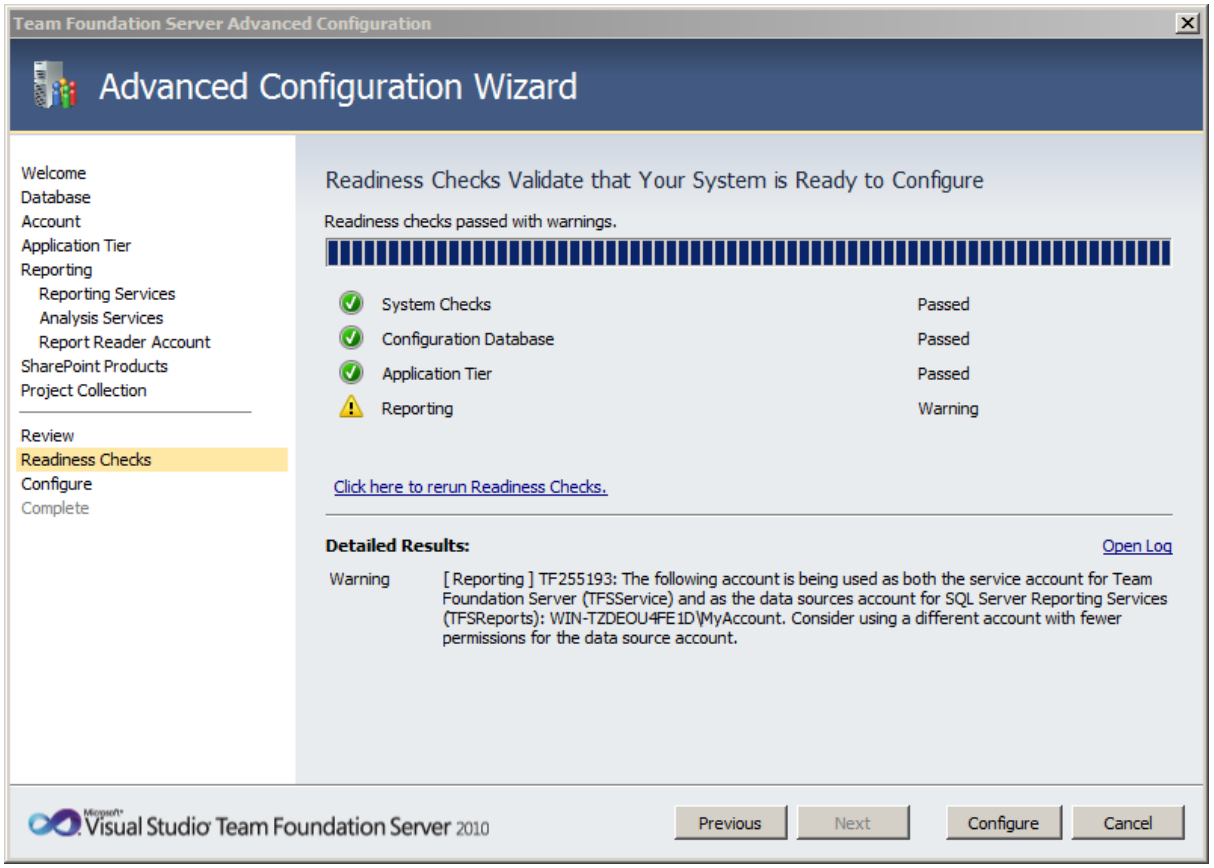
آخرین بخش پیکربندی مربوط به Project Collection می باشد. در این بخش برای پروژه یک Collection معرفی می شود. Collection مکانی است که پروژه در آن ذخیره می شود و برای دسته بندی تیم ها مورد استفاده قرار می گیرد پس نیاز است که حداقل یک Collection وجود داشته باشد. این Collection نیاز به نامی یکتا دارد. در صورت عدم تمایل به ساختن یک Collection پیش فرض کافی است علامت Create new Team Exploreram project collection برداشته شود. پس از پیکربندی نیز می توان Collection هایی به پروژه اضافه نمود(شکل 20-7).



شکل 20-7: تنظیمات Collection در ویزارد پیکربندی TFS

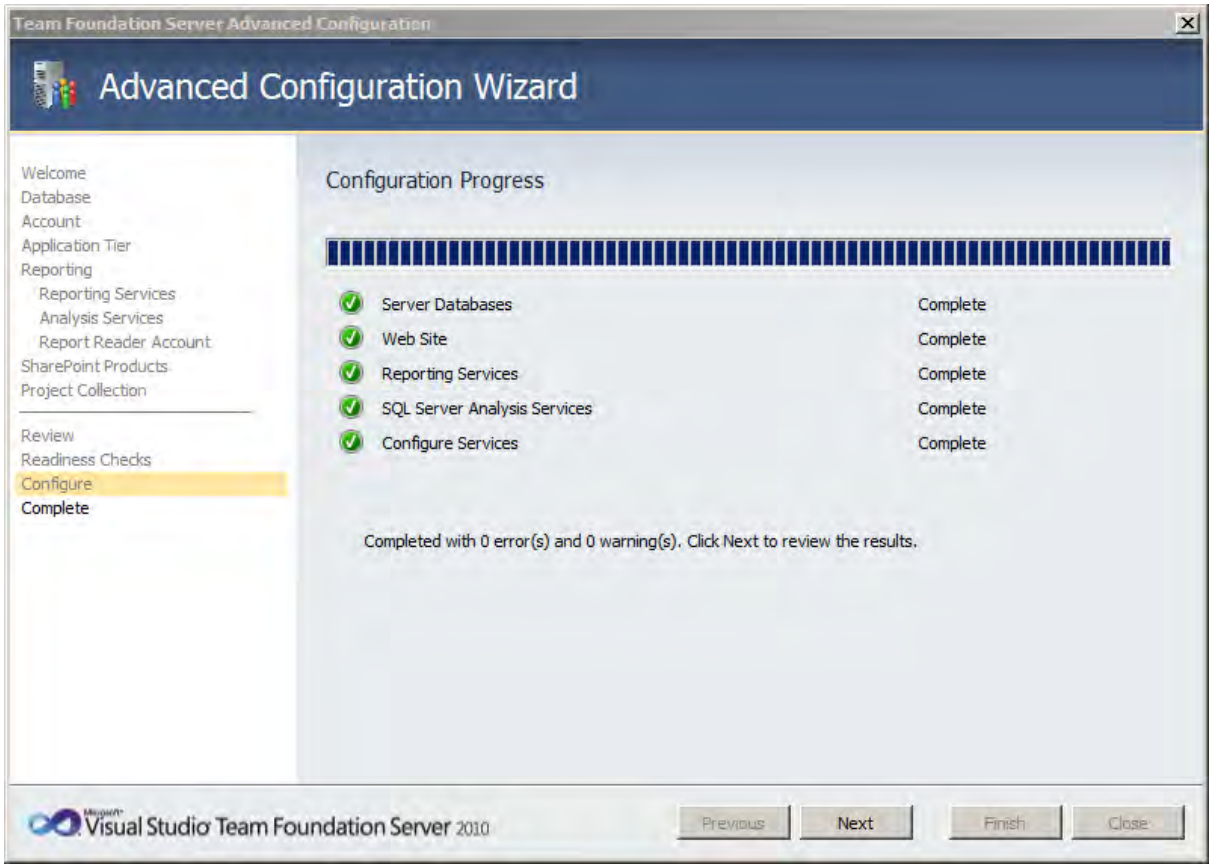
با کلیک روی Next به بخش Review خواهیم رفت که خلاصه از تنظیمات پیکربندی را ارائه می کند. بخش بعدی Readness Checks می باشد که با کلیک روی Verify نمایش داده می شود. این قسمت به بررسی تنظیمات پیکربندی اعمال شده در بخش های مختلف می پردازد. در صورت داشتن مشکل از طریق گزارشی مشکل مربوطه را به اطلاع کاربر می رساند.

در صورت عدم مشکل در پیکربندی کلید Configure فعال می شود(شکل 21-7).

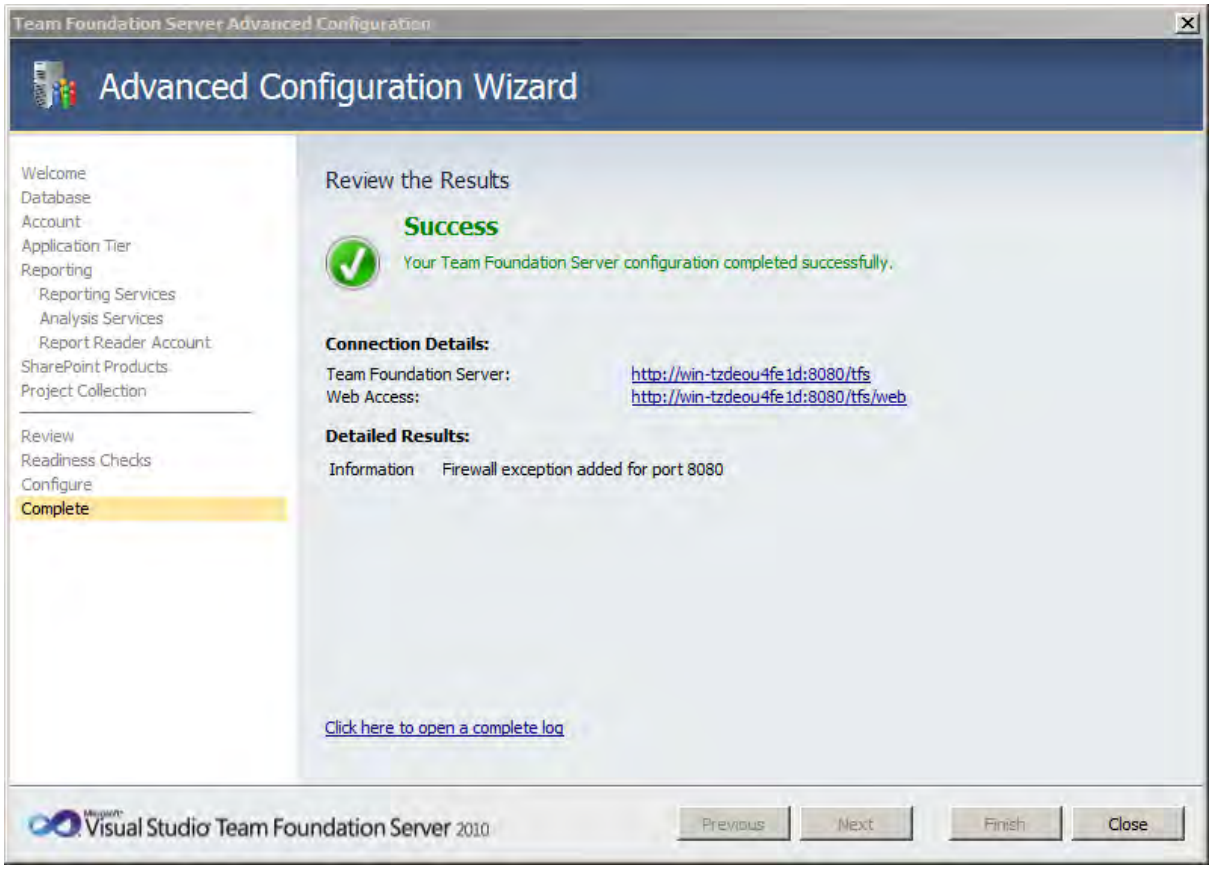


شکل 21-7: چک کردن سیستم برای اعمال پیکربندی TFS

با کلیک روی این کلید عملیات پیکربندی انجام می شود. توجه شود که این عملیات ممکن است زمان بر باشد. پس اتمام پیکربندی در صورت موفقیت آمیز بودن عملیات پیغامی مبتنی بر موفقیت آمیز بودن پیکربندی در بخش آخر یعنی Complete نمایش داده خواهد شد(شکل 7-22)(شکل 7-23).

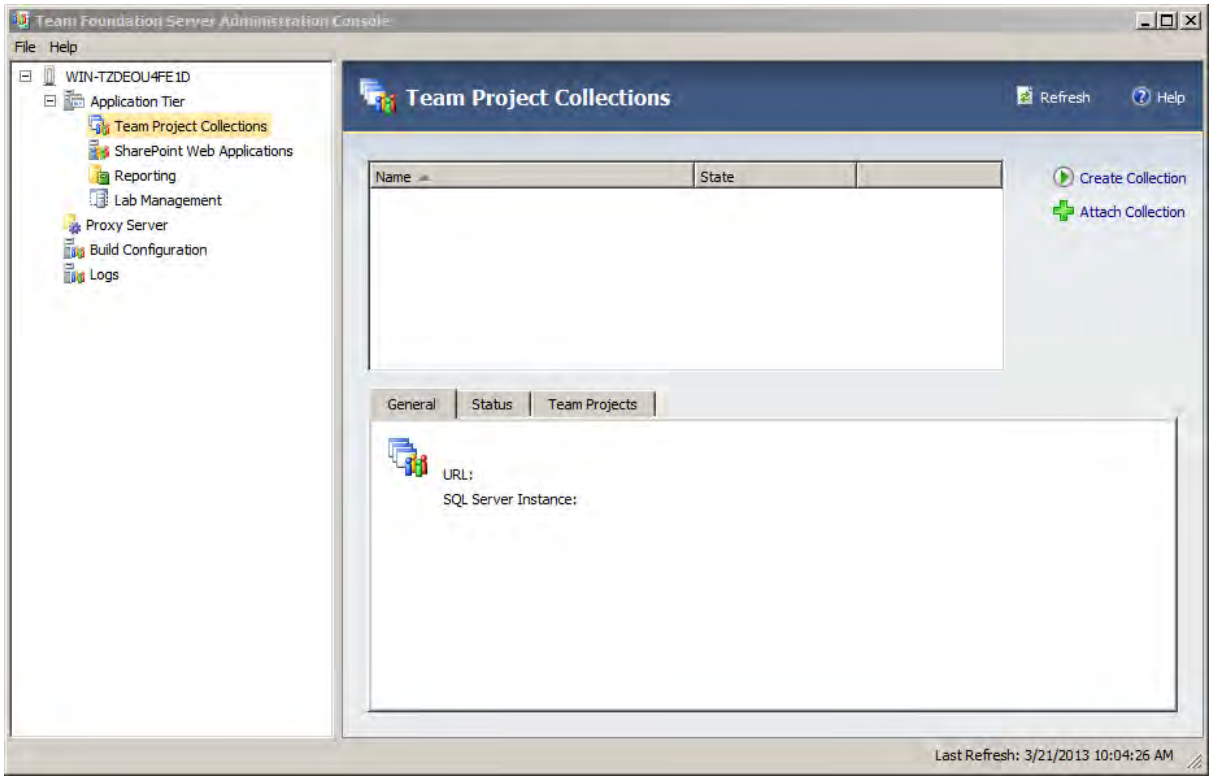


شکل 22: فرآیند پیکربندی TFS



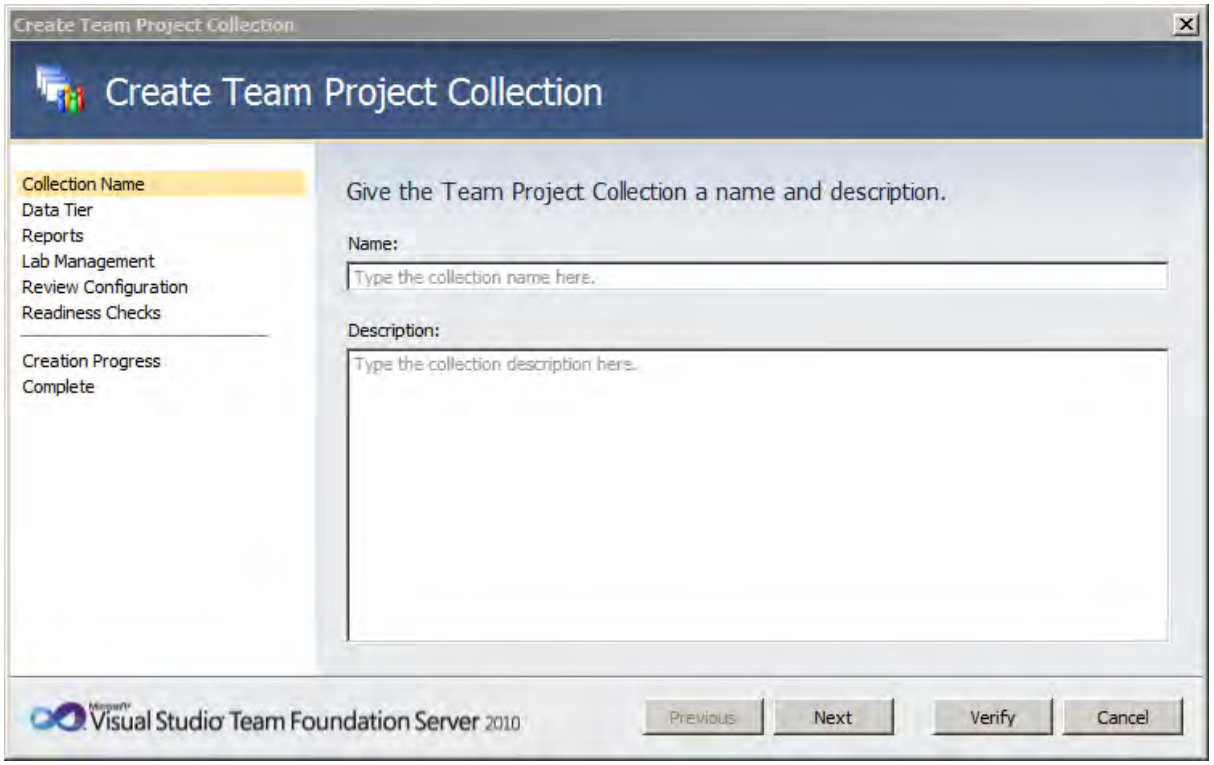
شکل 23-7: اتمام پیکربندی

همانطور که گفته شد Collection ها محل های ذخیره سازی پروژه می باشند. در سطح کلان هر کدام از تیم ها عضو یک Collection می باشند. اگر تیم های کمی در پروژه وجود داشته باشند صرفاً نیاز به یک Collection وجود دارد. Collection پیش فرض معمولاً هنگام پیکربندی ساخته می شود. در صورت عدم وجود Collection پیش فرض می توان از بخش Application Tier در پنجره Team Foundation Administration Consol گزینه Team Collection را انتخاب کرد. این عمل باعث ظاهر شدن پنل Team Project Collection می شود. در این پنل می توان به اضافه، حذف و مدیریت Collection ها اقدام کرد(شکل 24-7).



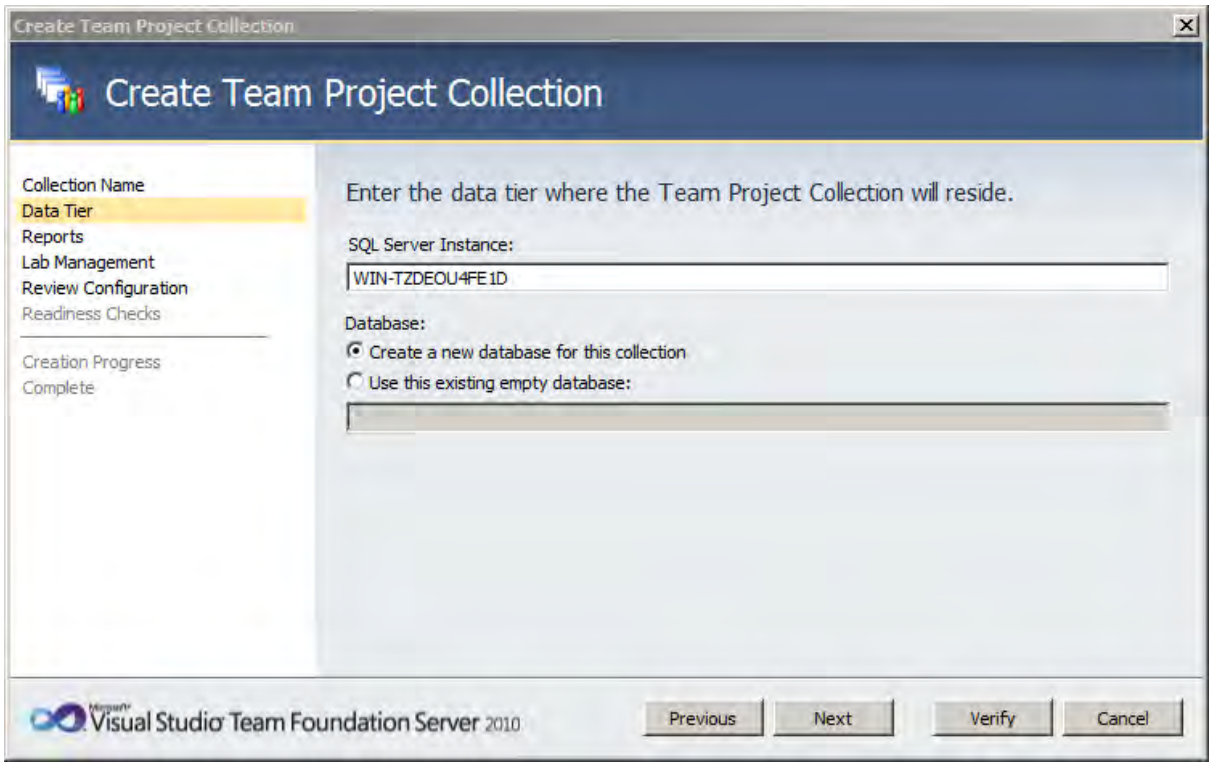
شکل 24-7: بخش مدیریت Collection در کنسول مدیریت TFS

برای اضافه کردن یک Collection باید روی Create Collection کلیک کرد تا ویزارد Create Team Project Collection اجرا شود(شکل 25-7).



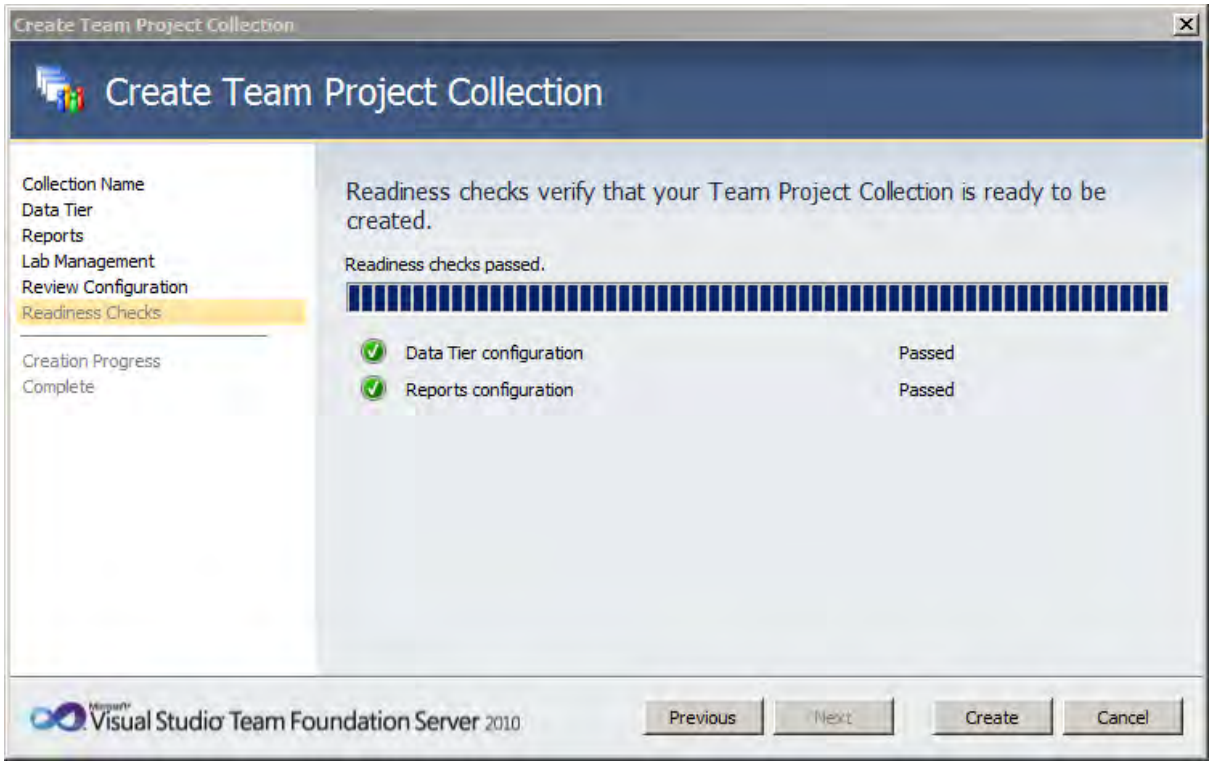
شکل 25-7: ورودی نام Collection در ویزارد ایجاد Collection

اولین گام این ویزارد نام و توضیحاتی درباره Collection اخذ می کند. گام دوم که با عنوان Data Tier مشخص شده است اطلاعات مرتبط با دیتابیس Collection را اخذ می کند. به ازای هر Collectionی که ساخته می شود یک دیتابیس ایجاد می گردد. از همین صفحه می توان با کلیک روی دکمه Verify اقدام به بررسی تنظیمات کرد(شکل 26-7).



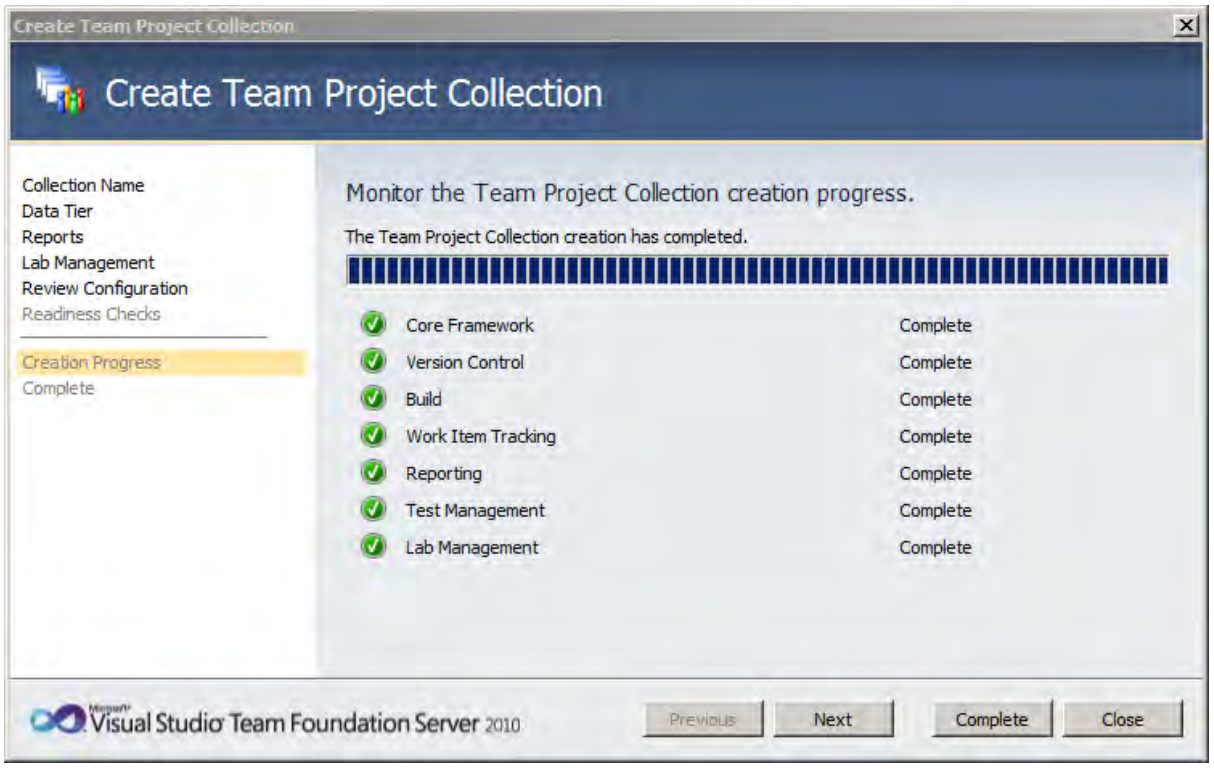
شکل 26-7: معرفی SQL Server Instance در ویزارد ایجاد Collection

در صورت عدم مشکل احتمالی دکمه Create فعال می شود(شکل 27-7).

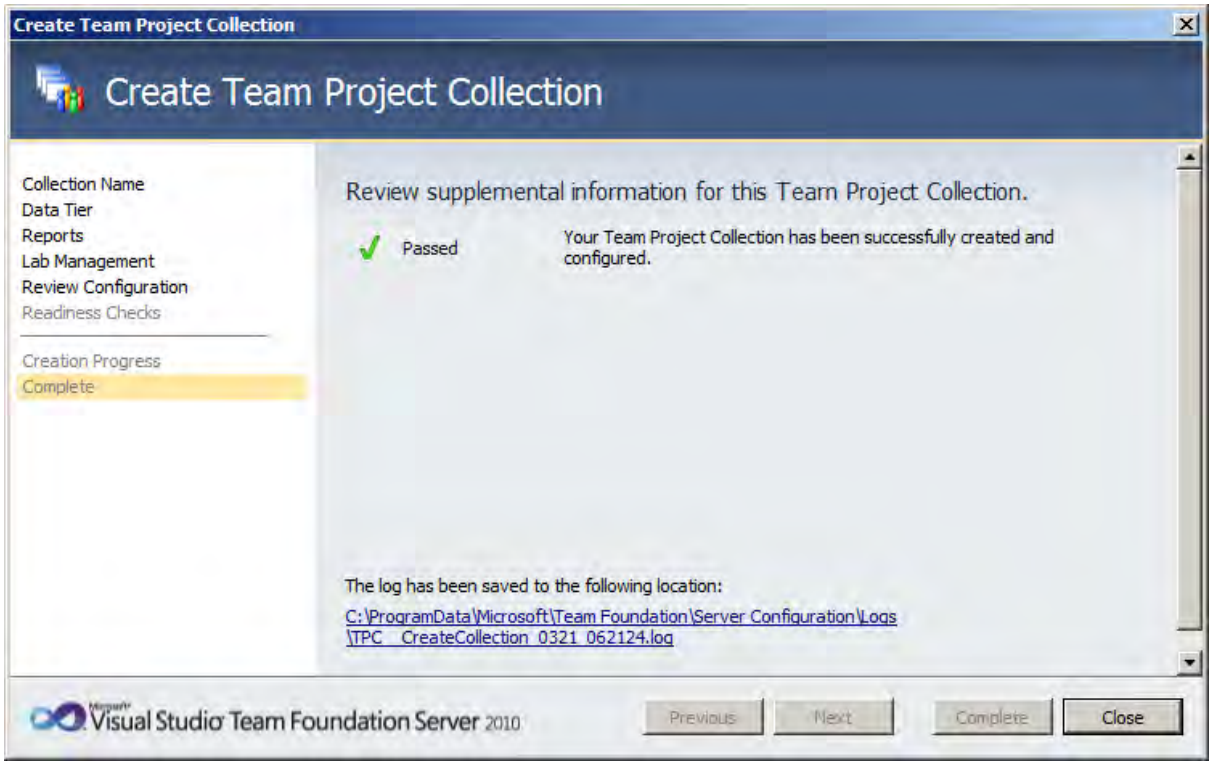


شکل 27-7: بررسی آمادگی ساخت Collection

با کلیک روی این دکمه فرآیند ایجاد Collection آغاز می شود. پس از گذشت چندی Collection با اعلان پیغامی مبتنی بر موفقیت آمیز بودن فرآیند، ایجاد می شود(شکل 28-7 و 29-7).

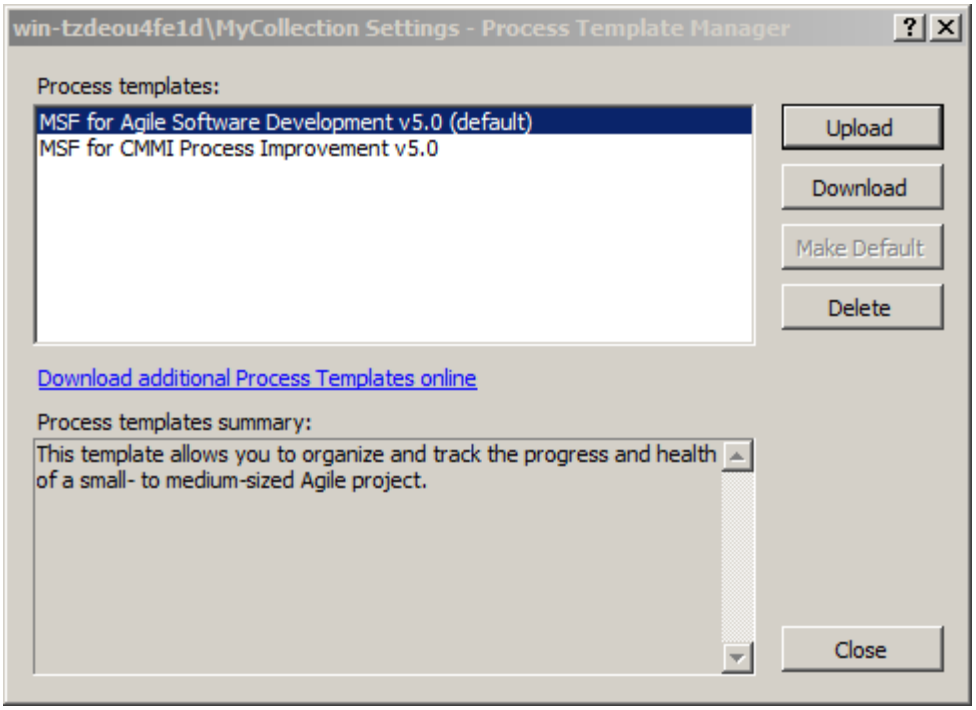


شکل 28-7: فرآیند ساخت Collection



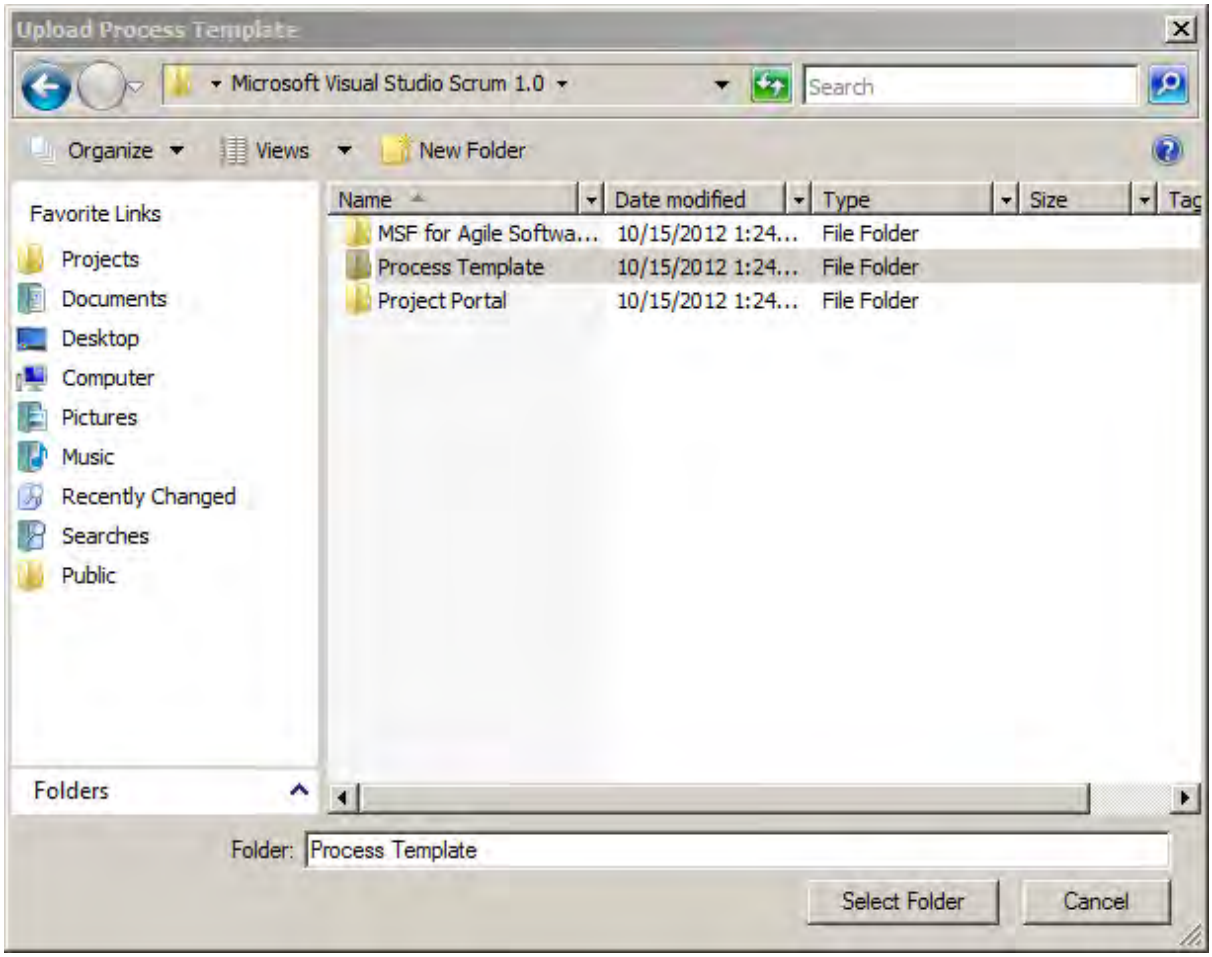
شکل 29-7: تکمیل ساخت Collection

برای ایجاد تیم ها در TFS و همچنین مدیریت تیم باید از محیط Team Explorer استفاده شود. اما پیش از ایجاد تیم ها باید قالب اسکرام در TFS بارگذاری شود. همانطور که قبلاً گفته شد TFS به جزء اسکرام از متدولوژی های دیگری چون MSF نیز پشتیبانی می کند. در TFS برای هر کدام از متدولوژی ها یک قالب تعریف شده است که شامل آیتم های کاری خاص همان متدولوژی می باشد. به صورت پیش فرض قالب اسکرام در TFS 2010 وجود ندارد. این قالب ابتدا باید از ... دانلود شده و سپس به TFS اضافه شود. فایلی که برای قالب اسکرام دانلود می شود یک فایل MSI می باشد و نیاز به نصب دارد. پس از دانلود و نصب این قالب باید از طریق Team Explorer آن را به TFS اضافه کرد(توجه شود که از این پس TFS به Team Explorer متصل شده است و سایر عملیات مرتبط با TFS از طریق این محیط انجام می شود). برای اضافه کردن قالب اسکرام به TFS باید از طریق مسیر Team->Team Project Collection Setting گزینه Process TemplaTeam Explorer Managment انتخاب شود. با کلیک بر روی گزینه فوق پنجره Process TemplaTeam Explorer Managment باز خواهد شد(شکل 30-7).



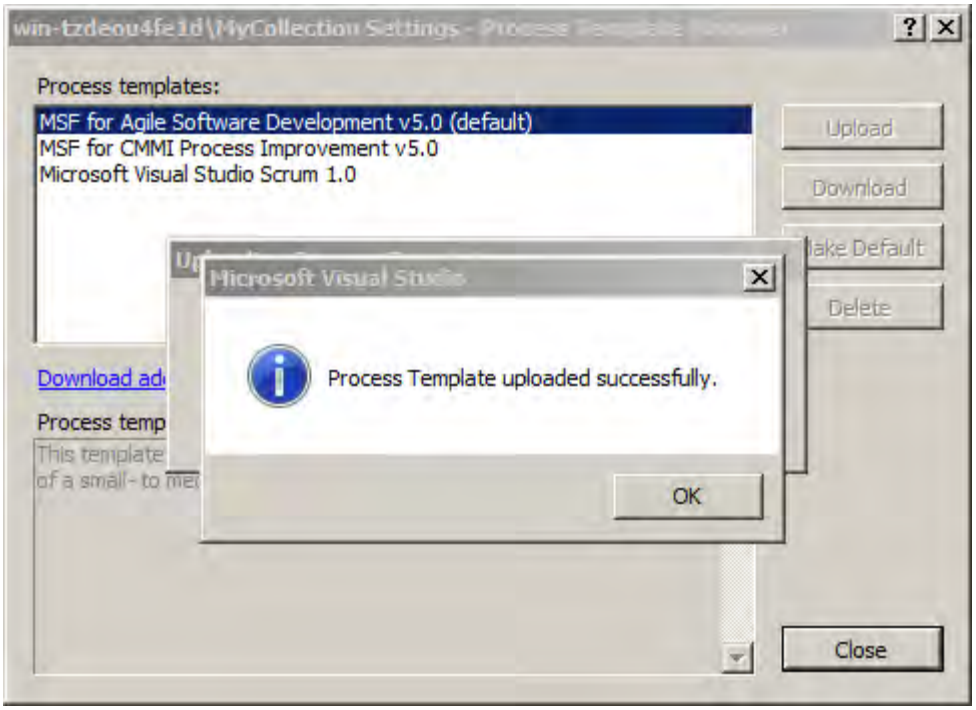
شکل 30-7: مدیریت قالب در TFS

کلیک روی دکمه Upload در این پنجره موجب باز شدن پنجره Upload Process TemplaTeam Explorer خواهد شد. این پنجره مسیر نصب قالب را درخواست می کند(توجه شود که این پنجره یک پوشه را انتخاب می کند و نه یک فایل را). با انتخاب پوشه Process TemplaTeam Explorer در پوشه اصلی نصب قالب و کلیک روی دکمه Select Folder، قالب اسکرام در TFS اصطلاحاً آپلود خواهد شد(شکل 31-7).



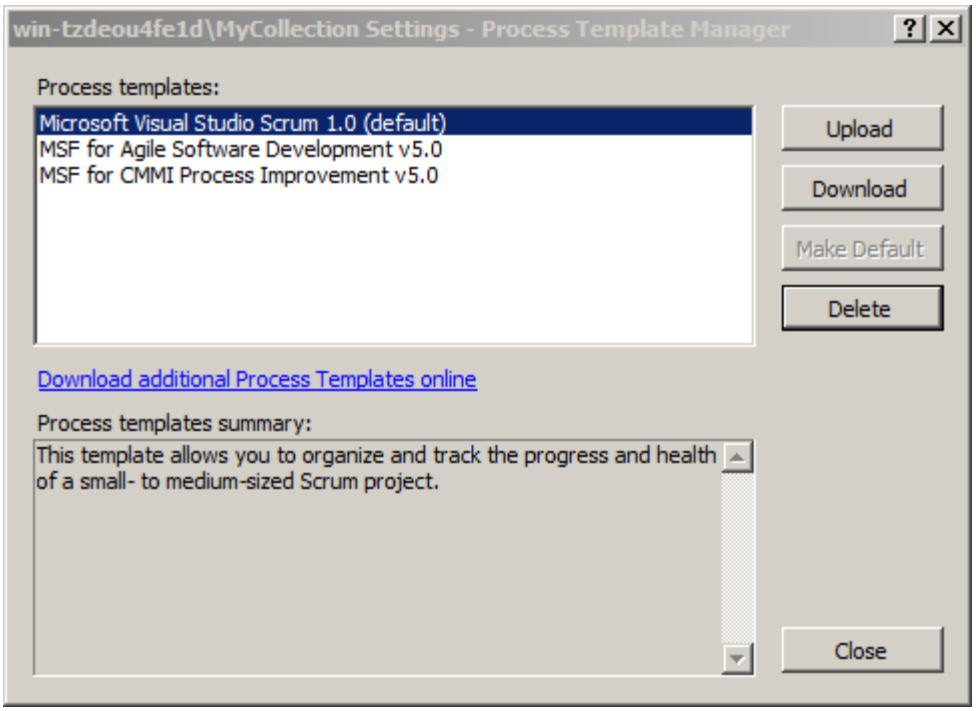
شکل 7-31: پنجره انتخاب قالب

اگر آپلود با موفقیت همراه باشد، Team Explorer از طریق پیغامی آن را به اطلاع کاربر می رساند(شکل 7-32).



شکل 7-32: اضافه شدن قالب اسکرام به TFS

پس از کامل شدن فرآیند آپلود، قالب اسکرام به عنوان Microsoft Visual Studio Scrum 1.0 به لیست Process TemplaTeam Explorer اضافه خواهد شد. می توان با انتخاب قالب اسکرام و کلیک روی گزینه Set Default این قالب را به عنوان قالب پیش فرض تعیین کرد(شکل 33-7).



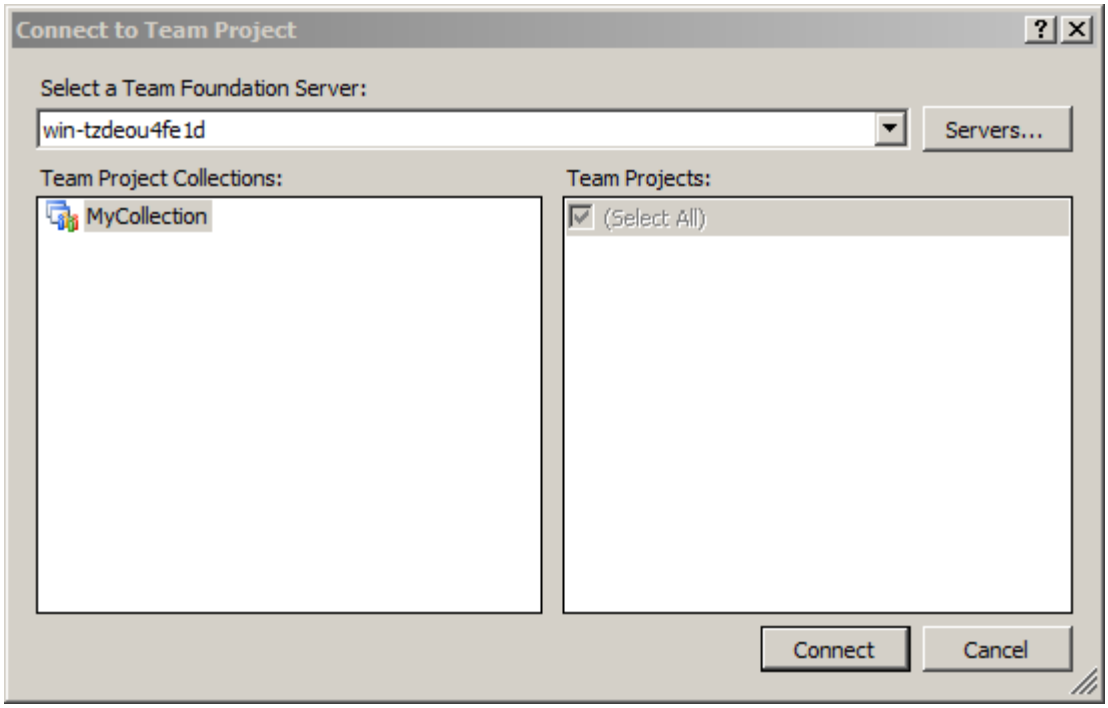
شکل 33-7: انتخاب قالب اسکرام به عنوان قالب پیش فرض

TFS از طریق پنجره Team Explorer قابل مدیریت می باشد. برای فعال سازی این پنجره می توان از طریق View->Team Explorer و یا کلید های ترکیبی Crtl+\(برای آشکار سازی) و Ctrl+M(برای پنهان سازی) اقدام کرد. این پنجره در سمت راست به صورت پیش فرض ظاهر می شود.

برای مشاهده تیم ها ابتدا باید Team Explorer به TFS متصل شود. سه روش برای این اتصال وجود دارد:

1. کلیک بر روی لینک Connect to Team Project در پنجره Start Page(این لینک در صورتی ظاهر خواهد شد که Start Page در حالت General Development قرار داشته باشد).
2. کلیک بر روی تک آیتم منوی Team یعنی Connect to Team Project
3. کلیک بر روی آیکن Connect to Team Project در پنجره Team Explorer

هر کدام از روش های بالا به گشوده شدن پنجره Connect to Team Project ختم خواهند شد(شکل 34-7).

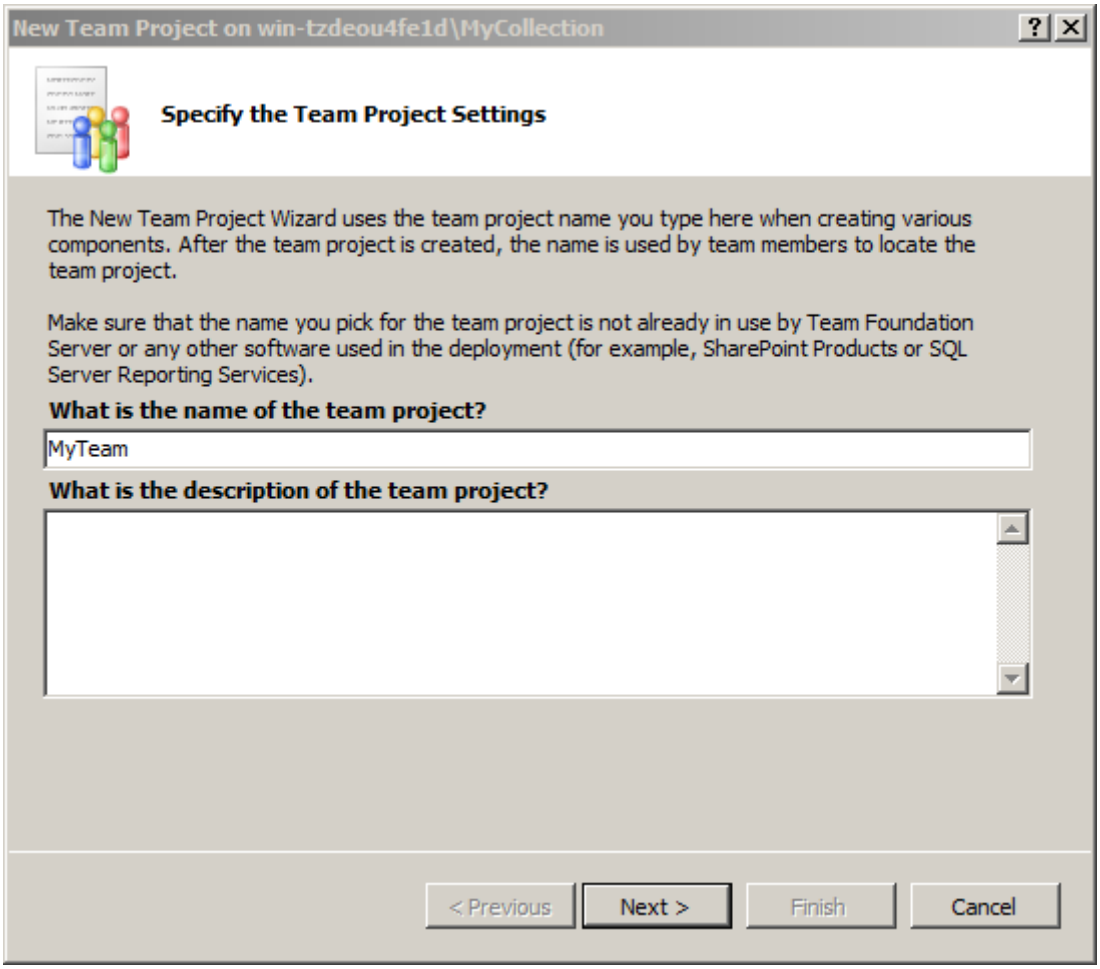


شکل 34-7: پنجره اتصال به TFS

می توان با استفاده از این پنجره به تیم های موجود در TFS وصل شد. این پنجره از سه بخش تشکیل شده انتخاب سرور، انتخاب Collection و انتخاب تیم. اگر سرور برای TFS تعریف شده باشد، در لیست انتخاب سرور وجود خواهد داشت. انتخاب سرور مذکور لیست Collection های آن در بخش پایین(سمت چپ) نمایش داده می شود. و به همین ترتیب با انتخاب یکی از Collectionها لیست تیم های موجود در آن Collection نیز آشکار می شود. اما از آن جایی که تا به حال تیمی به هیچ Collection اضافه نشده است لیست سمت راست خالی می باشد. با کلیک روی دکمه Connect تنها سرور و Collection مورد انتخاب در Team Explorer به نمایش در خواهد آمد.

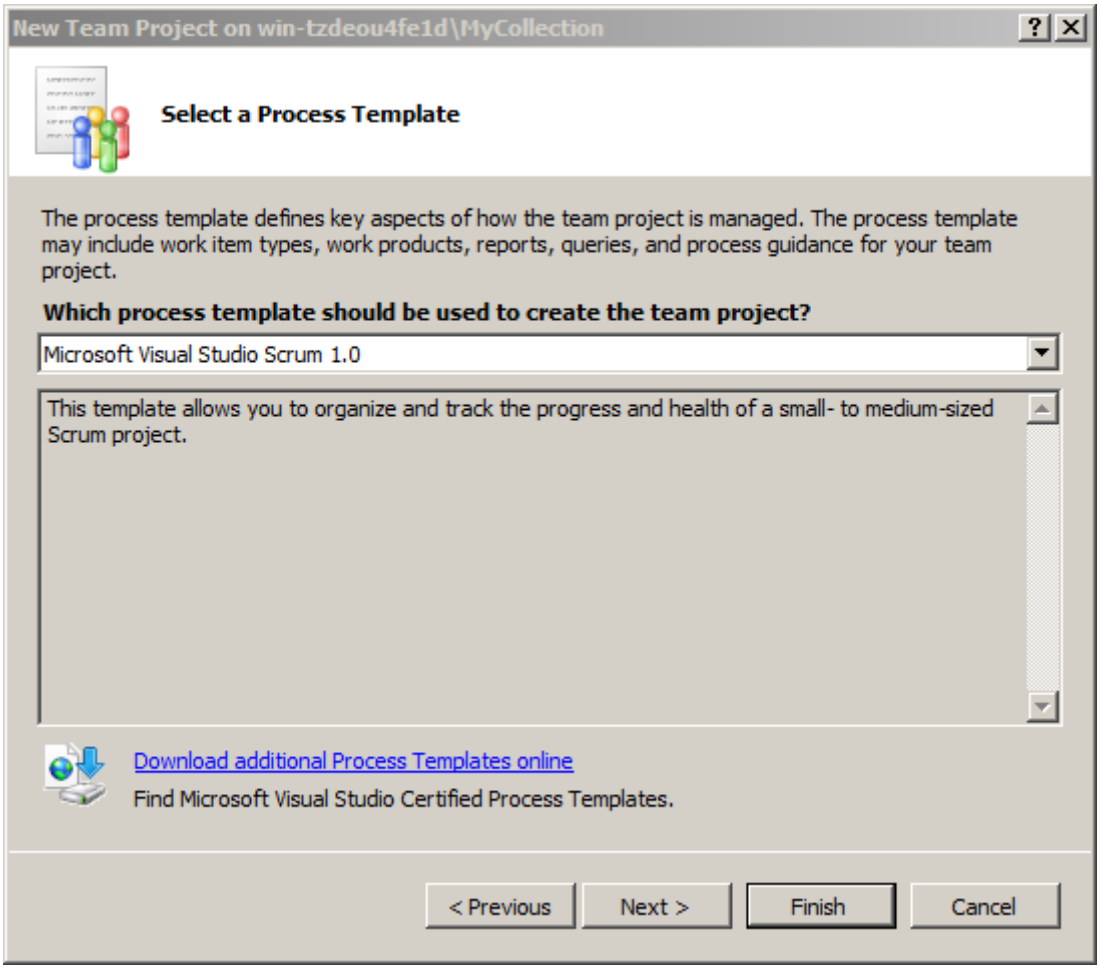
اضافه کردن یک تیم به TFS با استفاده از Team Explorer(Visual Studio) قابل انجام است. کافی است از مسیر File->New گزینه Team Project انتخاب شود. با انتخاب آن در صورت عدم اتصال سرور Collection به Team Explorer پنجره Connect to Team Project مجدداً نمایش داده می شود.

از این پنجره باید سرور و Collection مورد نظر(که تیم قرار است در آن ساخته شود) انتخاب شود. با کلیک روی دکمه Connect اتصال سرور/Collection به Team Explorer برقرار می شود. اما با این تفاوت که در این حالت ویزاردی برای افزودن تیم به TFS ظاهر خواهد شد(شکل 35-7).



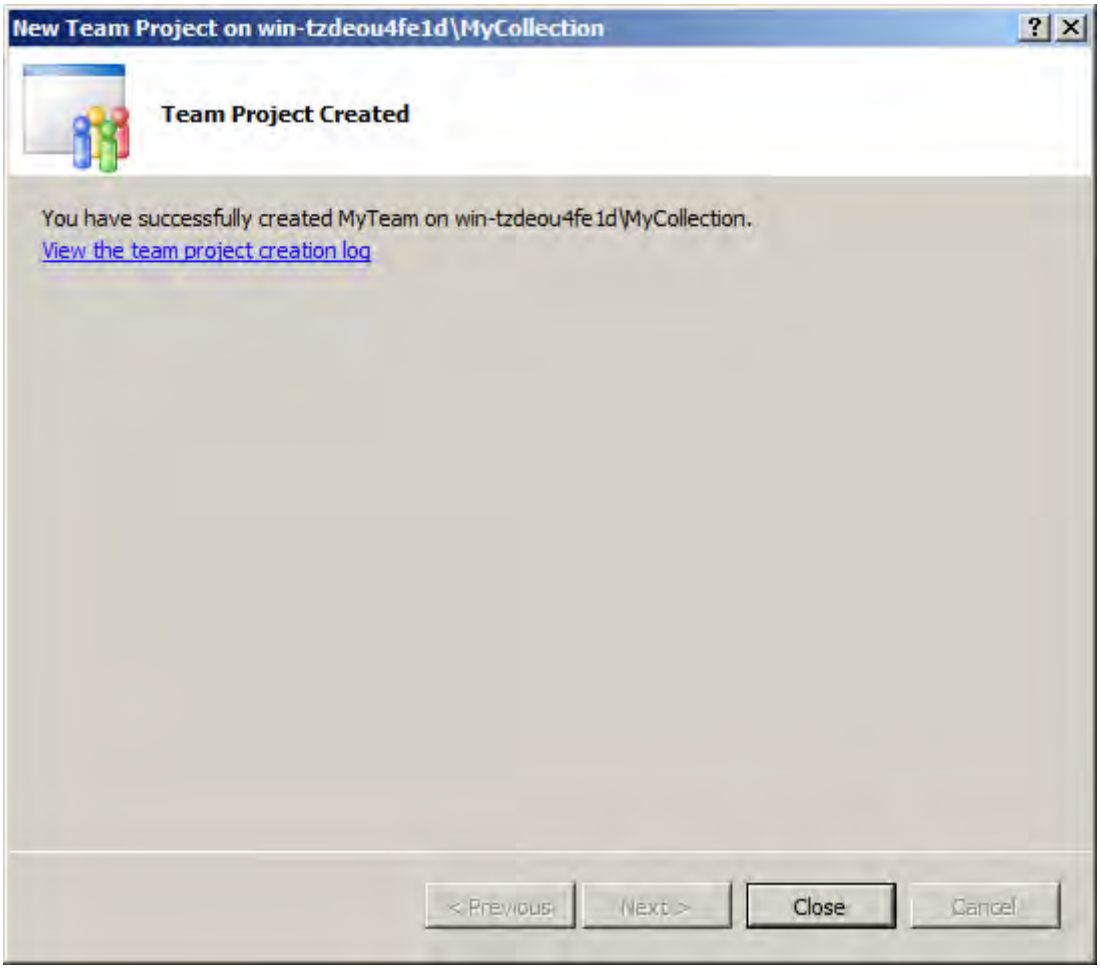
شکل 35-7: ورودی نام و توضیحات و ویزارد ساخت تیم

در گام اول ویزارد نام و توضیحات تیم از کاربر دریافت می شود. نام تیم باید در Collection یکتا انتخاب شود. با کلیک روی دکمه Next بخش انتخاب قالب نمایش داده خواهد شد. در این بخش لیستی از قالب های موجود در TFS وجود دارد. در صورتی که پیش از این قالب اسکرام در TFS آپلود شده باشد، این قالب در این لیست قابل مشاهده خواهد بود(شکل 36-7).



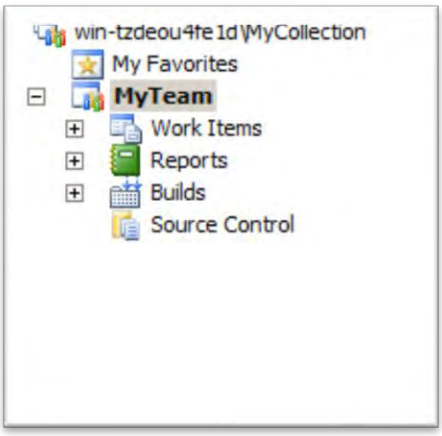
شکل 36-7: انتخاب قالب در ویزارد ساخت تیم

با انتخاب قالب اسکرام و کلیک بر روی Finish فرآیند ایجاد تیم آغاز می شود. با اتمام این فرآیند که ممکن است چند دقیقه به طول بیانجامد، پنجره نهایی با پیغامی مبنی بر موفقیت آمیز بودن ساخت تیم به نمایش در خواهد آمد(شکل 37-7).



شکل 37-7: تکمیل فرآیند ساخت تیم

در صورتی که تمام مراحل فوق به درستی انجام شده باشند، تیم جدید به TFS اضفه می شود و در پنجره Team Explorer نمایش می یابد(شکل 38-7).



شکل 38-7: نمایی از پنجره Team Explorer در Visual Studio

آیتم تیم، خود از بخش های مختلفی تشکیل می شود. آنچه در جهت نیل به هدفمان در این کتاب نیاز داریم، در بخش های WorkItem و Report قرار دارد. در طول فصل های آتی به طور مفصل این دو بخش تشریح خواهند شد.

در پنجره Team Explorer می توان بیش از یک تیم را همزمان مشاهده کرد. با راست کلیک کردن روی سرور جاری و انتخاب گزینه Connect to Team Project یا انتخاب همین گزینه از جعبه ابزار Tean Explorer می توان بر اساس مراحلی که سابق بر این گفته شد تیم جدید را متصل ساخت. توجه شود که این گزینه برای اتصال تیم هایی است که قبلاً ایجاد شده اند. برای ایجاد تیم جدید می توان از منویی که به وسیله راست کلیک باز خواهد شد، گزینه New Team Project را انتخاب کرد.

در صورت نیاز به خارج کردن تیمی از Team Explorer، در منوی راست کلیک گزینه ای به نام Disconnect وجود دارد که این وظیفه را انجام می دهد. با کلیک روی این گزینه پیغامی مبنی بر بسته شدن آیتم های کاری و قطع ارتباط سرور TFS ظاهر می شود و از کاربر برای این عمل تأییدیه گرفته می شود. گزینه Disconnect باعث حذف یک سرور از Team Explorer خواهد شد. اما در صورتی که نیاز باشد که یک تیم تنها از Team Explorer حذف شود(سرور آن حفظ شود)، باید روی تیم مورد نظر راست کلیک شود و سپس گزینه Remove انتخاب شود. با این عمل مانند حالت Disconnect از کاربر برای حذف آیتم تأییدیه گرفته می شود. مجدداً تأکید می شودکه گزینه Remove صرفاً موجب حذف یک تیم از Team Explorer خواهد شد.

یکی از امکانات مفید Team Explorer بخش My Favorities آن می باشد. از طریق این بخش می توان آیتم های مورد علاقه و مورد استفاده را از دسترس پذیرتر نمود. برای استفاده از آن کافی است آیتم مورد علاقه را Copy نموده و در بخش My Favorities الحاق کرد. برای دسته بندی آیتم های مورد علاقه نیز می توان از پوشه(Folder)ها استفاده نمود. لازم به ذکر است که ایجاد پوشه ها به صورت سلسله مراتبی نیز مقدور خواهد بود.

فصل هشتم: مدیریت اعضا

TFS توسط افراد مختلفی مدیریت می شود. به عبارت دیگر هر فردی که در توسعه نقشی داشته باشد می تواند به TFS دسترسی داشته باشد. و بر اساس اختیاراتش در آن در تغییرات ایجاد کند. توسعه دهندگان در TFS به عنوان Member یا عضو شناخته می شوند. هر کدام از اعضا باید در ماشین سرور یک اکانت داشته باشند. این اکانت باعث دسترسی Remote اعضا به ماشین سرور و ابزار TFS می شود. سرور در اینجا کامپیوتری است که نقش مرکز داده و نرم افزاری TFS را بر عهده دارد و سایر کامپیوترها به عنوان اعضا و یا کلاینت به آن دسترسی خواهند داشت.

پس از نصب بخش نرم افزار TFS اعضا باید در سرور تعریف شوند. متخصصان آی تی بر اساس نوع سیستم عامل سرور به تعریف این اعضا خواهند پرداخت(در صورتی که از ویندوز سرور استفاده می شود، معمولاً برای تعریف اعضا از امکاناتی چون Active Directory استفاده می شود). پس از تعریف همه اعضا هر کدام از توسعه دهندگان می توانند با حالت Remote به سرور متصل شوند و بر اساس مجوزشان به مشاهده و تغییر آیتم های کاری بپردازند.

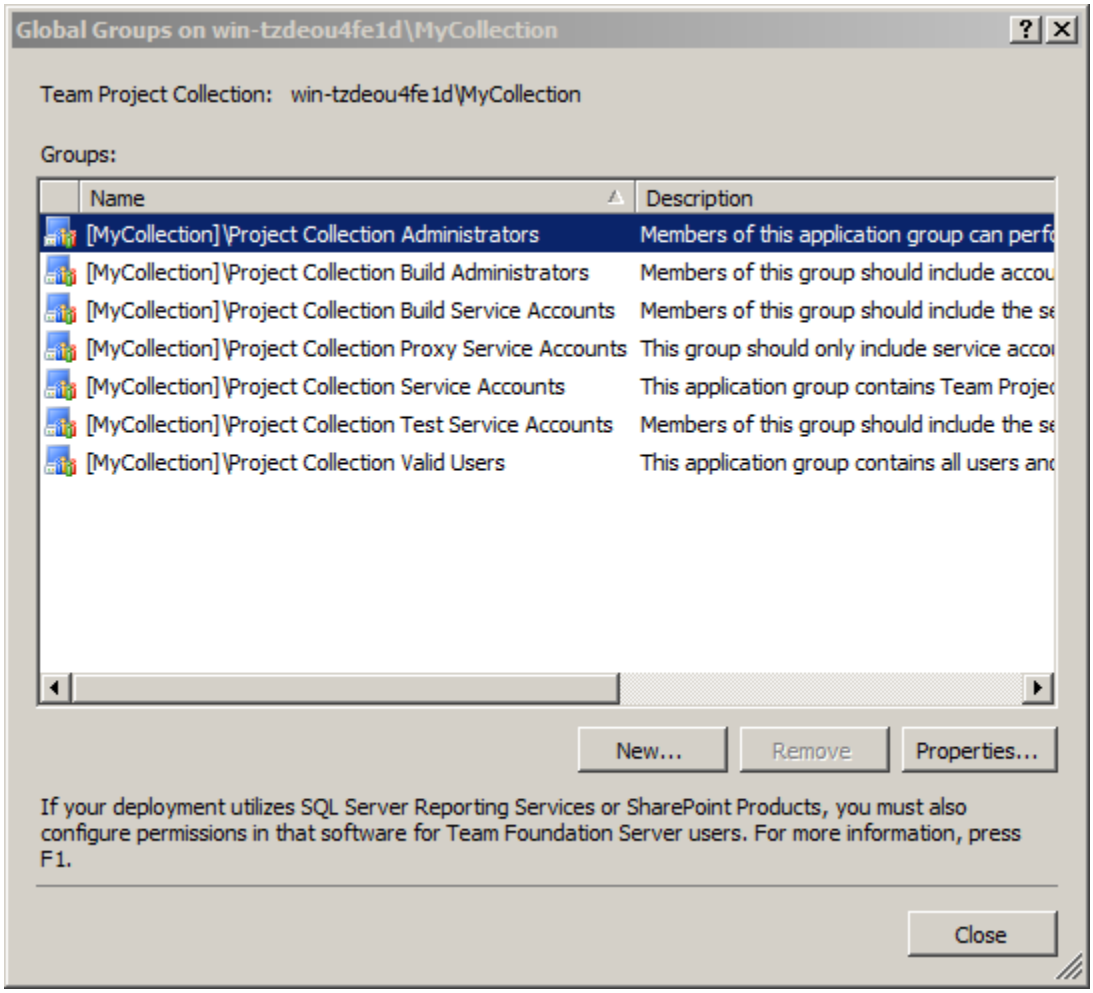
هر عضوی که در سرور تعریف می شود تا زمانی که عضو یکی از گروه های TFS قرار نگرفته باشد قادر به دسترسی به TFS نخواهد بود. TFS دارای گروه های از پیش تعریف شده ای است که می توان از آنها برای عضو گیری استفاده نمود.

همانطور که در فصل های پیش ذکر شد هر کدام از نقش های موجود در TFS دسترسی خاصی به آیتم های کاری دارند. به عنوان مثال توسعه دهندگان نمی توانند اولویت بک لاگ محصول را تغییر دهند. به این اختیارات نقش های مجوز گفته می شود. هر کدام از گروه ها دارای مجوز هایی هستند که سطح دسترسی به TFS را تعریف می کنند. این مجوز ها از مجوزهایی که در سطح شبکه برای دسترسی به ماشین(کامپیوتر) ها داده می شود متفاوت هستند(اگر چه هر دو آنها مورد استفاده اعضای تعریف شده در Active Directory می باشند). مجوزهای سطح شبکه برای دسترسی به ماشین و مجوزهای TFS برای دسترسی به آیتم های کاری اسکرام استفاده می شوند. مزیت گروه ها این است که می توان برخی مجوزها را به صورت دسته ای به اعضا اعطا نمود.

گروه بندی اعضا در سطح Collection

تعریف گروه ها و گروه بندی اعضا در دو سطح قابل انجام است. گروه های سطح Collection و گروه های سطح تیم. اعضای گروه های سطح Collection اغلب مدیرانی هستند که به اعمال مدیرتی در سطح Collection می پردازند. اما در طرف دیگر اعضای گروه های سطح تیم معمولاً از توسعه دهندگان تشکیل می شوند.

برای تعریف گروه های سطح Collection ابتدا باید بر روی Collection مورد نظر راست گلیک شود و از بخش Team Project Collection Setting گزینه Group Membership انتخاب شود. این عمل باعث گشوده شدن پنجره Global Groups on Server/Collection خواهد شد(شکل 1-8).



شکل 1-8: پنجره مدیریت گروه در سطح Collection

انتهای نام پنجره ترکیب سرور/Collection قرار خواهد گرفت. به صورت پیش فرض در لیست Group گروه های از پیش تعریف شده ای موجود است. این گروه ها عبارت اند از:

Project Collection Administrator: اعضای این گروه مسئول همه عملیات اعطای مجوز روز کل Collection می باشند.

Project Collection Build Adminidtrator: اعضای این گروه شامل افرادی هستند که در پروژه Buildها را مدیریت می کنند.

Project Collection Service Account: اعضای این گروه شامل حساب های سرویسی هستند که به وسیله Build Serverهای نصب شده در این Collection استفاده می شوند.

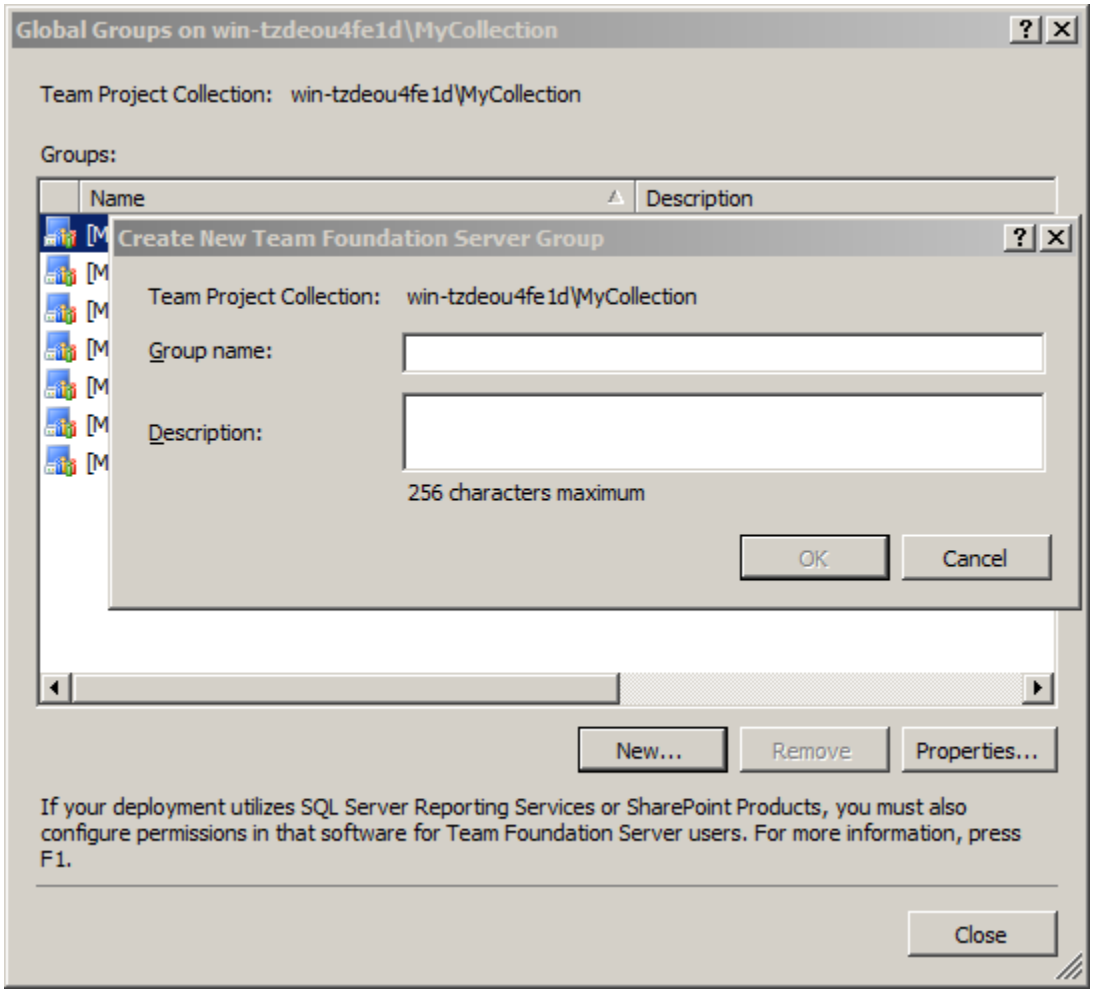
Project Collection Proxy Service: این گروه تنها شامل آن دسته حساب های سرویسی است که به وسیله پراکسی نصب شده در این Collection مورد استفاده قرار می گیرند.

Project Collection Service Account: این گروه شامل حساب های سرویس Collection می باشد.

Project Collection Test Service Accounts: اعضای این گروه شامل حساب های سرویس استفاده شده به وسیله کنترل کننده های تست نصب شده در Collection جاری می باشند.

Project Collection Valid Users: این گروه شامل همه کاربران و گروه هایی است که به Collection دسترسی دارند.

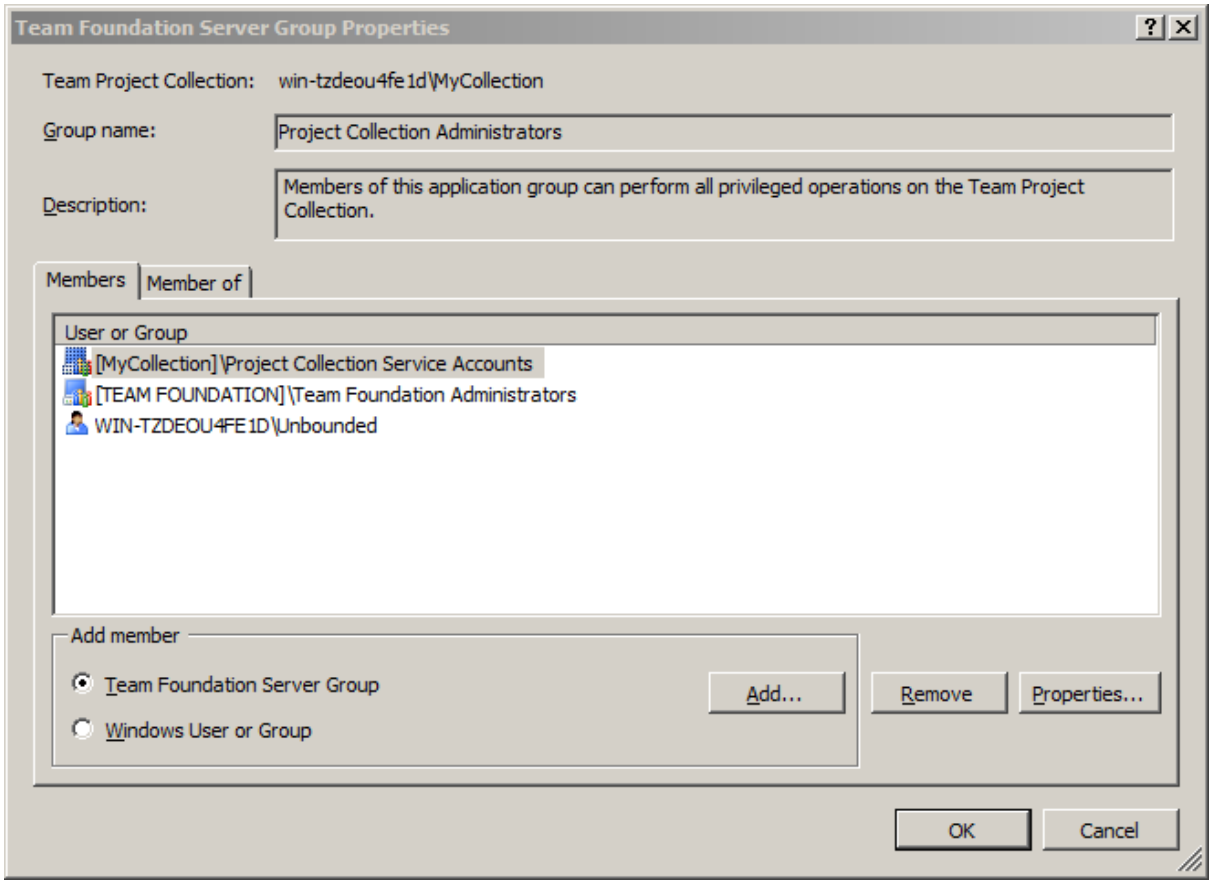
می توان گروه های دیگری نیز علاوه بر این گروه ها به Collection اضافه نمود. برای این کار کافی است در پایین کادر Group روی گزینه New کلیک شود. این کار باعث باز شدن پنجره Create New Team Foundation Server Group خواهد شد(شکل 2-8).



شکل 2-8: پنجره ساخت گروه جدید

این کادر دارای دو فیلد نام و توضیحات می باشد. نام گروه باید یک واژه حداکثر 256 کاراکتری یکتا باشد. پس از ورود اطلاعات خواسته شده با کلیک بر روی دکمه OK گروه جدید به Collection افزوده خواهد شد. به دلیل این که گروه جدید در سطح Collection ایجاد می شود پیش از نام آن، نام Collection قرار خواهد گرفت.

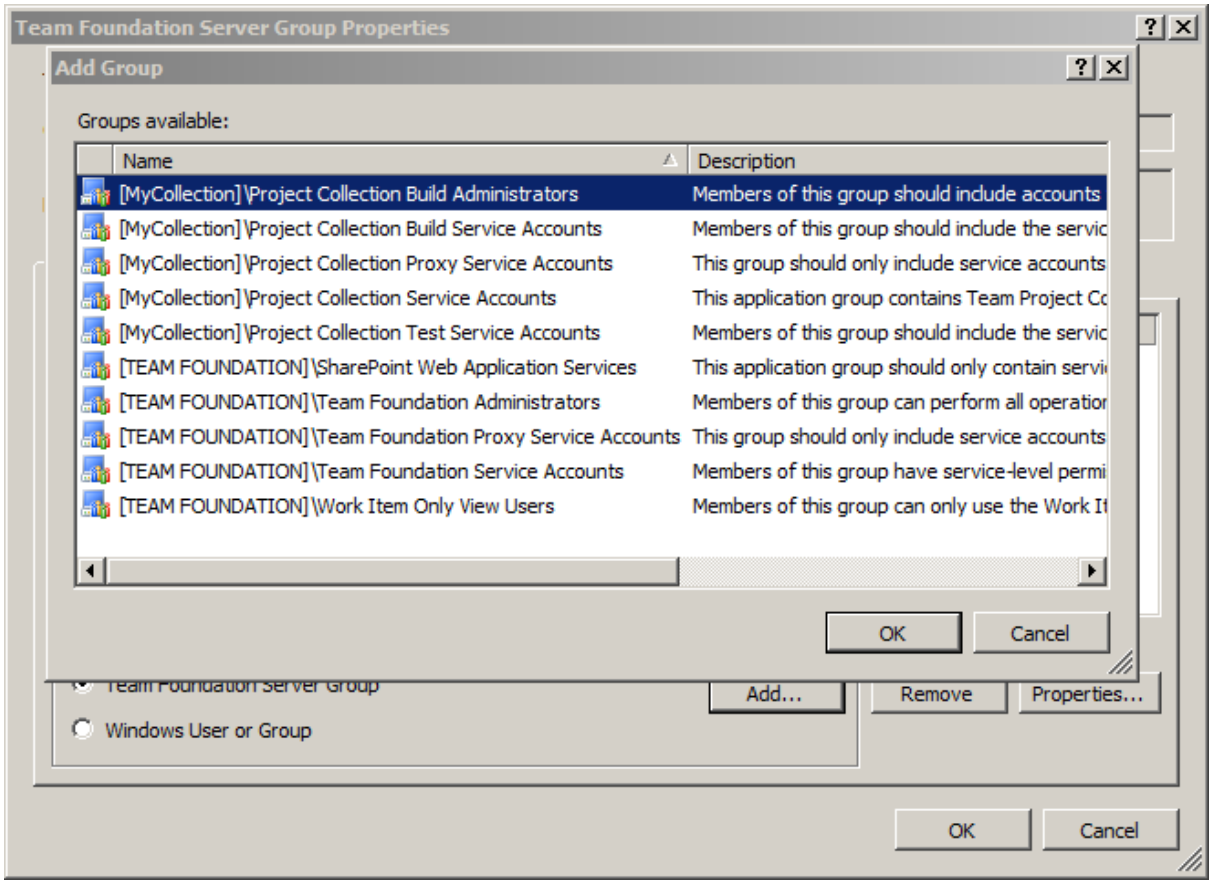
گروه هایی که در Collection قرار دارند در حالت پیش فرض خالی از اعضا می باشند. پس از تعریف گروه ها باید اعضای آن ها نیز تعیین شوند. همام طور که قبلاً ذکر شد اعضا در ابزاری همچون Active Directory ثبت و مدیریت خواهند شد. اعضای ثبت شده باید به عضویت گروه های TFS درآیند. برای افزودن یک عضو به گروه ابتدا باید در صفحه Global Groups on Server/Collection روی یک گروه در لیست Group کلیک شود. با فشردن دکمه Properties پنجره Team Foundation Server Group Properties برای گروه منتخب باز خواهد شد(شکل 3-8).



شکل 3-8: پنجره ویژگی های گروه های TFS

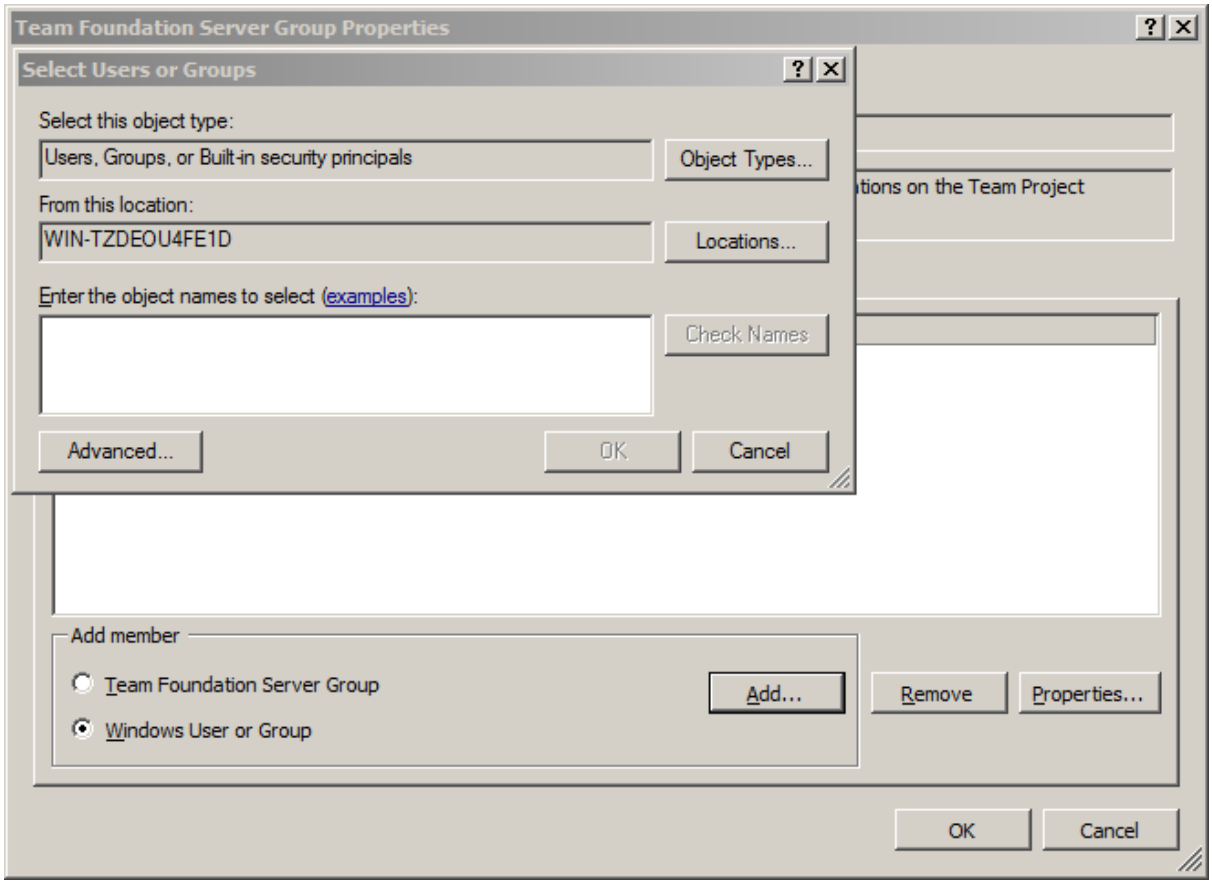
در میان صفحه دو Member و Member of وجود دارد. تب Member دارای لیستی است که کاربران و گروه های عضو آنها را نشان می دهد. هر گروه علاوه بر کاربران می تواند شامل گروه های دیگر نیز باشد. بدیهی است که اعضای گروه درونی به صورت سلسله مراتبی از مجوزهای گروه های بیرونی استفاده می کنند.

برای اضافه کردن گروه یا کاربر باید از بخش ...Add استفاده شود. دو حالت برای اضافه کردن موجود است. اولین حالت اضافه کردن یک گروه از TFS می باشد. با انتخاب این گزینه و فشردن کلید ...Add پنجره Add Group گشوده خواهد شد(شکل 4-8).



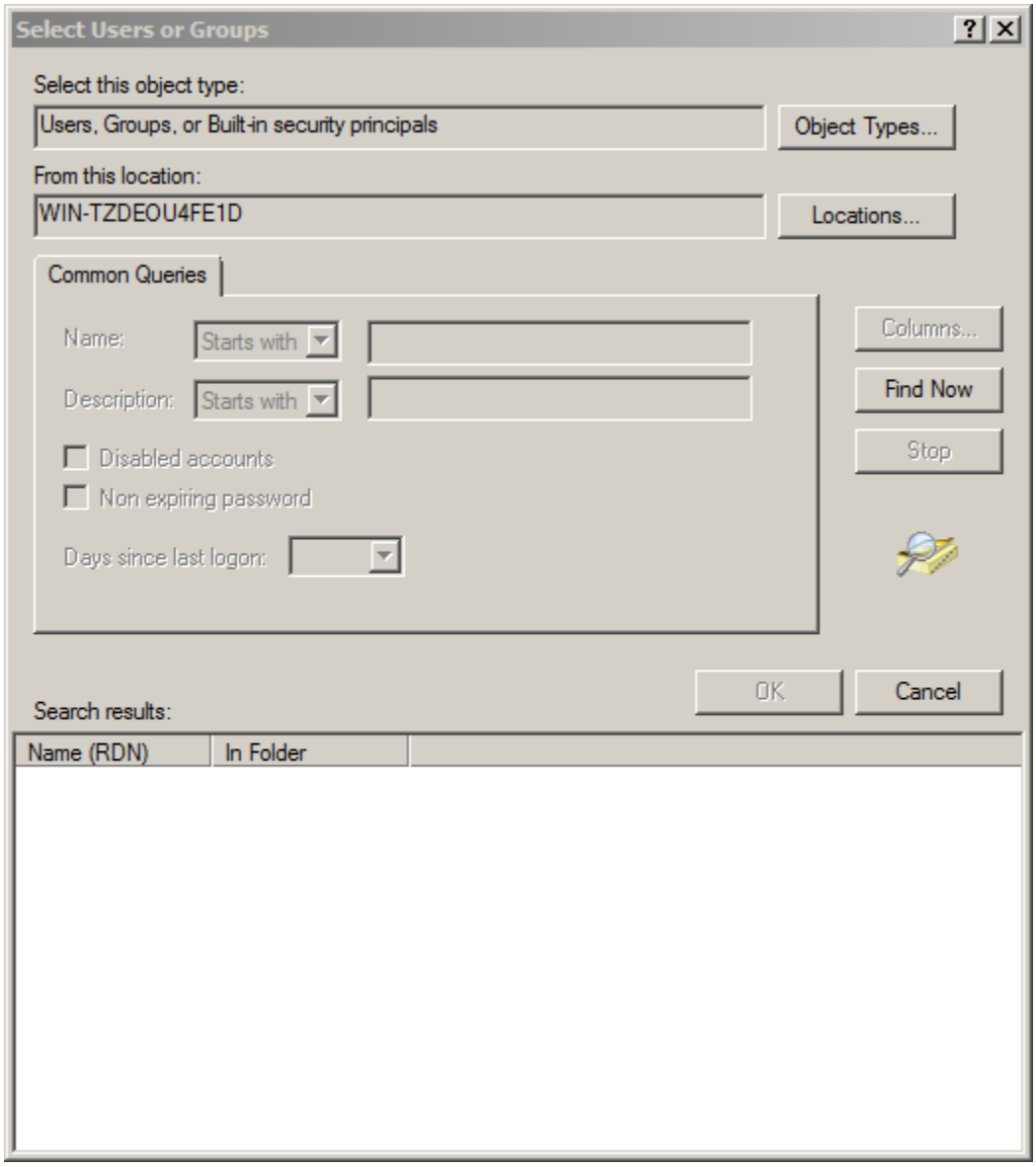
شکل 4-8: پنجره افزودن گروه

در این پنجره همه گروه های Collection نمایش داده خواهند شد. با انتخاب هر کدام از آنها و زدن کلید OK آن گروه به لیست اعضا اضافه خواهد شد. حالت دوم یعنی Wondows User or Group اضافه یک کاربر یا گروه از ویندوز می باشد. در این حالت گروه های تعریف شده در ویندوز که قابل افزودن هستند. با انتخاب این گزینه و فشردن کلید Add پنجره معروف Select User and Group ویندوز نمایان خواهد شد(شکل5-8).



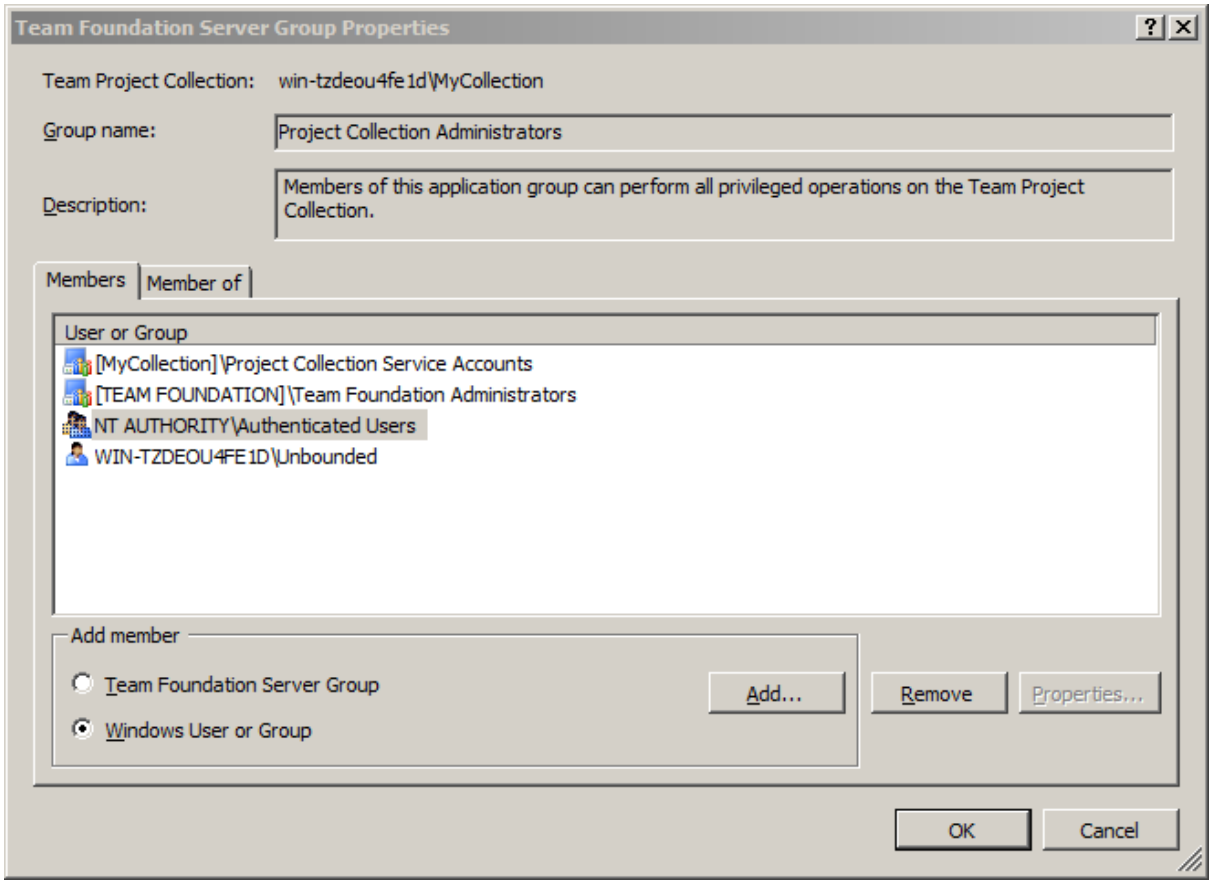
شکل 5-8: پنجره انتخاب کاربران و گروه ها

برای افزودن یک کاربر یا یک گروه کافی است نام آن در بخش Enter the object names to select تایپ شود. روش دیگر برای یافتن گروه ها و کاربران استفاده از حالت Advanced می باشد. با فشردن این دکمه که در پایین کادر قرار دارد، پنجره جدیدی باز می شود که از طریق آن می توان به جستجوی کاربران و گروه ها پرداخت(شکل6-8).



شکل 6-8: : پنجره انتخاب کاربران و گروه ها(حالت Advanced)

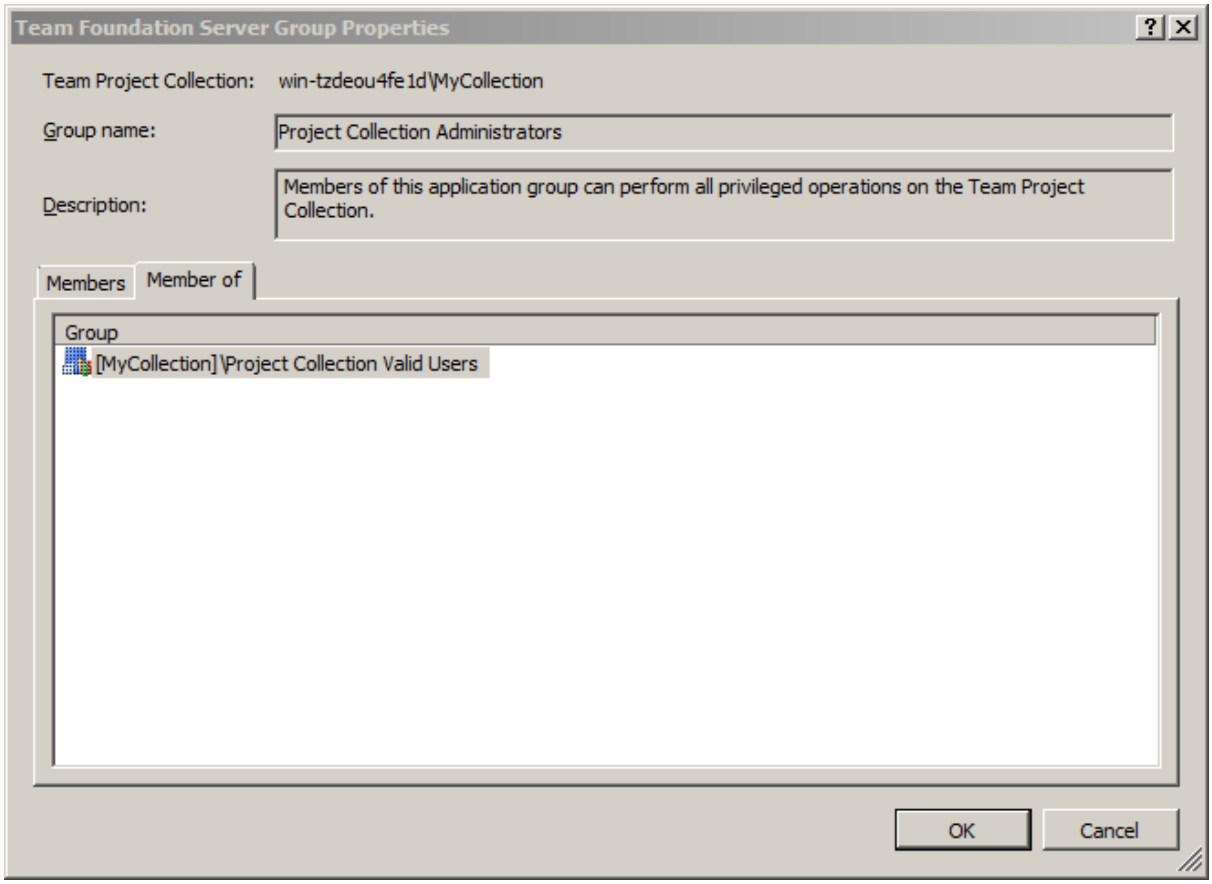
راحت ترین عمل، استفاده از کلید Find Now در وسط صفحه می باشد. کلیک روی این کلید باعث نمایان همه کاربران و گروه ها در لیست Search results خواهد شد. پس از آنکه کاربر و یا گروه مورد نظر انتخاب شد باید روی کلید OK کلیک شود. با بسته شدن این صفحه نام کاربر یا گروه منتخب درکادر Enter the object names to select صفحه قبل ظاهر خواهد شد. با کلیک بر روی کلید دکمه OK می توان کابر یا گروه انتخاب شده را به لیست اعضا اضافه کرد(به خاطر داشته باشید که تنها کابر Administrator قادر به حذف و اضافه کاربران و اعطای مجوز به آنها می باشد) (شکل 7-8).



شکل 7-8: گروه اضافه شده به TFS

برای لغو عضویت یک کاربر از لیست اعضا کافی است کاربر مورد نظر انتخاب شود و سپس روی دکمه Remove کلیک شود. جهت مشاهده جزئیات هر کدام از اعضا باید پس انتخاب عضو مورد نظر از لیست اعضا روی Properties کلیک شود.

همانطور که ذکر شد هر گروه خود می تواند عضوی از گروه های دیگر باشد. برای مشاهده این که یک گروه عضو چه گروه هایی می باشد می توان از Member of از صفحه Properties مرتبط با هر گروه استفاده کرد. در این تب لیستی از گروه هایی که این گروه عضو آنها می باشد نمایش داده می شود(شکل 8-8).

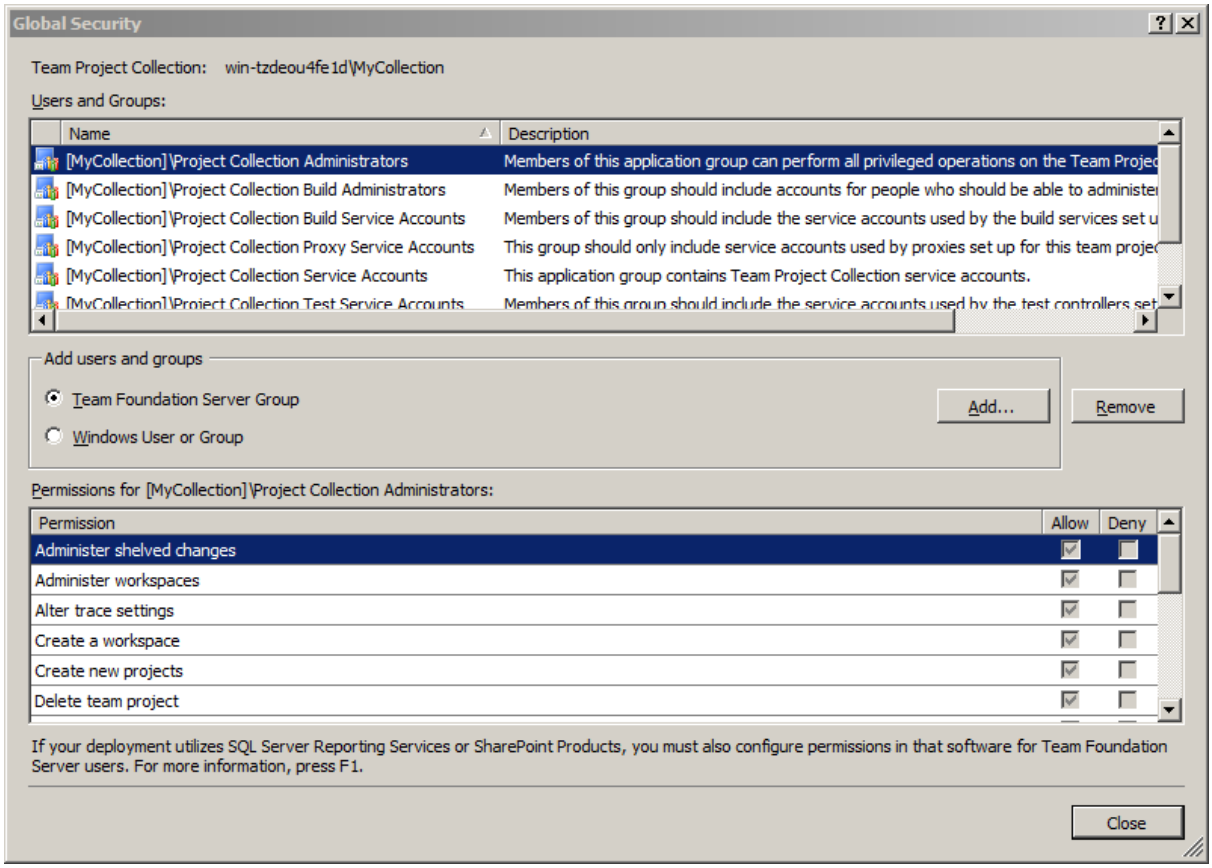


شکل 8-8: تب Member of برای یک گروه

لازم به ذکر است که در صفحه Global Groups on Server/Collection کلیدی به نام Remove وجود دارد که به وسیله آن می توان گروه ها را حذف نمود. این دکمه صرفاً برای حذف گروه هایی است که کاربران اضافه می کنند. گروه هایی پیش فرض TFS قابل حذف شدن نمی باشند.

مجوزها در سطح Collection

با اضافه شدن کاربران به تیم ها، از این پس هر کاربر مطابق با مجوزهایش به TFS دسترسی دارد. هر کدام از گروه های از پیش تعریف شده دارای مجوزهای خاصی می باشند. برای تغییر این مجوزها باید روی Collection راست کلیک کرده و از منوی باز شده Setting و سپس Security انتخاب شود. با این عمل پنجره Global Security گشوده خواهد شد(شکل 9-8).



شکل 9-8: تنظیمات امنیتی گروه ها در TFS

این پنجره از دو بخش تشکیل می شود. بخش فوقانی لیستی از گروه های سطح Collection l ارائه می دهد. این لیست متشکل از گروه های از پیش تعریف شده TFS و گروه هایی تعریف شده کاربر می باشد. با انتخاب هر کدام از گروه ها می توان به تغییر مجوزهای آنها اقدام کرد. در بخش پایین تعدادی مجوز وجود دارد که هر کدام از مجوزها در مقابل خود دو انتخاب را برای کاربر مهیا می کنند: Allow به معنی اجازه دادن و Deny به معنی سلب اجازه می باشد. هر مجوزی هر قرار است به هر گروهی اعطا گردد باید در این بخش تعریف شود. از میان همه گروه ها، مجوز های گروه Administriator را نمی توان تغییر داد. این گروه مجوز کامل همه نوع عملیاتی را در تیم داراست. مجوزهایی که در این بخش ارائه می شوند عبارت اند از:

AdminisTeam Explorerr shalved changed: مدیریت لغو و تغییرات

AdminisTeam Explorerr workspace: مدیریت فضای کاری

AlTeam Explorerr trace setting: تغییر تنظیمات ردگیری

Create a wokespace: ساختن فضای کاری

Create new project: ایجاد پروژه های جدید

Delete Team Exploreram project: حذف تیم

Edit collection-level information: ویرایش اطلاعات سطح Collection

Make requests on behalf of other: ایجاد درخواست از طرف دیگری

Manage build resources: مدیریت Buildها

Manage process Team ExplorermplaTeam Explorer: مدیریت قالب فرآیندها

Manage Team Explorerst controlles: مدیریت کنترل کنندگان تست

Manage work iTeam Explorerm link type: مدیریت نوع لینک های آیتم های کاری

Trigger events: رویدادها تریگر

Use build resources: استفاده از منابع Buildها

View build resources: نمایش منابع Buildها

View collection-level information: نمایش اطلاعات سطح Collection

View sysTeam Explorerm synchronization information: نمایش اطلاعات همگام سازی سیستم

گروه بندی در سطح تیم

آنچه تاکنون راجع به گروه ها و مجوزها گفته شد در سطح Collection جای داشت. کاربران این گروه ها اغلب مدیران سطح بالای Collectionها می باشند. اما گروه بندی دیگری نیز موجود است که به گروه بندی سطح تیم موسوم است. اکثر افراد پروژه کاربرانی از این نوع گروه ها می باشند. روش ها حذف و اضافه کردن گروه های سطح تیم و اعطای مجوز به آنها عیناً مشابه گروه های سطح Collection می باشد اما تفاوت هایی در نوع گروه های پیش فرض و نوع مجوزها آنها وجود دارد که به شرح زیر است:

علاوه بر گروه های معرفی شده در بخش Collection، گروه های جدیدی در سطح تیم وجود دارد که عبارت اند از:

Build: اعضای این گروه اضافه، حذف و تغییر Biuldها را در اختیار دارند. همچنین آنها می توانند Buildهای کامل شده یا صف بندی شده را مدیریت کنند.

Contribtors: اعضای این گروه توانایی اضافه، حذف و تغییر آیتم های درون تیم را دارا می باشند.

Project Administrator: اعضای این گروه می توانند انواع عملیاتی را در سطح تیم انجام دهند.

Reader: اعضای این گروه تنها می توانند به پروژه تیم دسترسی کند اما قدرت تغییر را ندارند.

مجوزها در سطح گروه

مجوزهای این سطح از گروه ها مطابق با عملیات دورن تیم ها است. این مجوزها عبارت اند از:

Create Team Explorerst runs: مدیریت اجرای تست ها در تیم

Delete Team Exploreram project: حذف پروژه تیم

Delete Team Explorerst run: حذف تست ها

Edit project-level information: ویرایش اطلاعات سطح پروژه

Manage Team Explorerst configurations: مدیریت پیکربندی تست ها

Manage Team Explorerst environments: مدیریت محیط تست در تیم

View project-level information: نمایش اطلاعات سطح پروژه

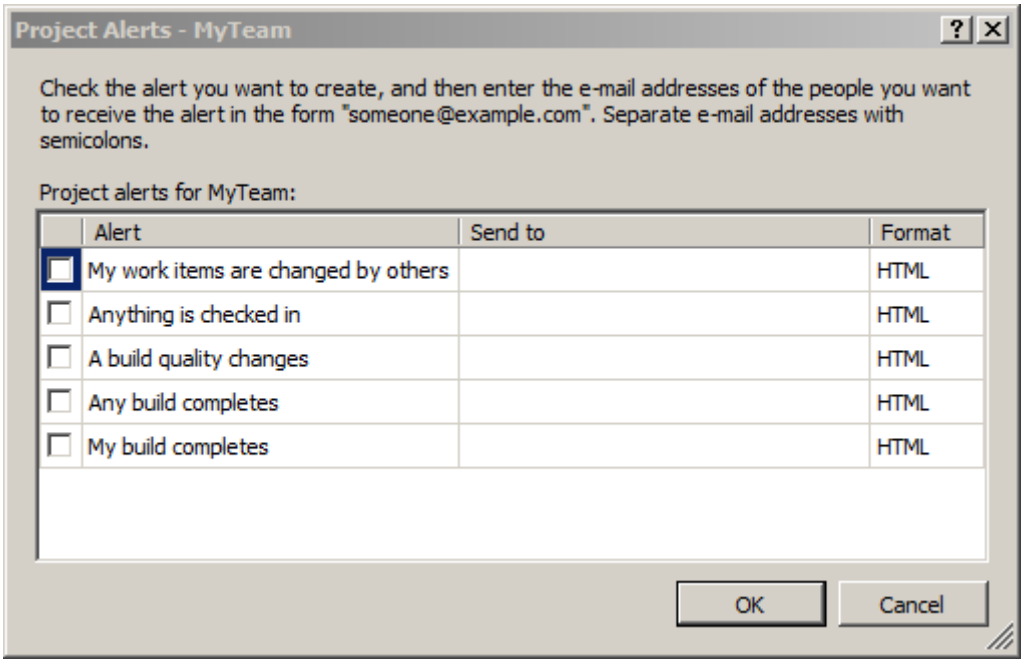
View Team Explorerst runs: نمایش تست ها در تیم

مجوز همه گروه ها به جزء گروه Project Administrators قابل تغییر است. مجوز این گروه استثناً قابل تغییر نمی باشد.

قابلیت Alert

یکی از امکانات جالبی که TFS در سطح تیم در اختیار کاربران قرار می دهد Project Alerts می باشد. این امکان یک سیستم اطلاع رسانی است که در صورت پیشآمد رویدادی از طریق ایمیل افرادی را مورد اطلاع قرار می دهد. به عنوان مثال اگر آیتم کاری شخصی توسط دیگری تغییر یابد(البته در صورتی که آن شخص مجوز اعمال تغییر را داشته باشد)، فرد اصلی از طریق ایمیل آگاهی پیدا می کند.

برای دسترسی و تنظیم این سرویس باید پس از راست کلیک کردن روی تیم مورد نظر در TFS گزینه Project Aletrs انتخاب شود. چنین عملی باعث باز شدن صفحه Project Aletrs خواهد شد(شکل 10-8).

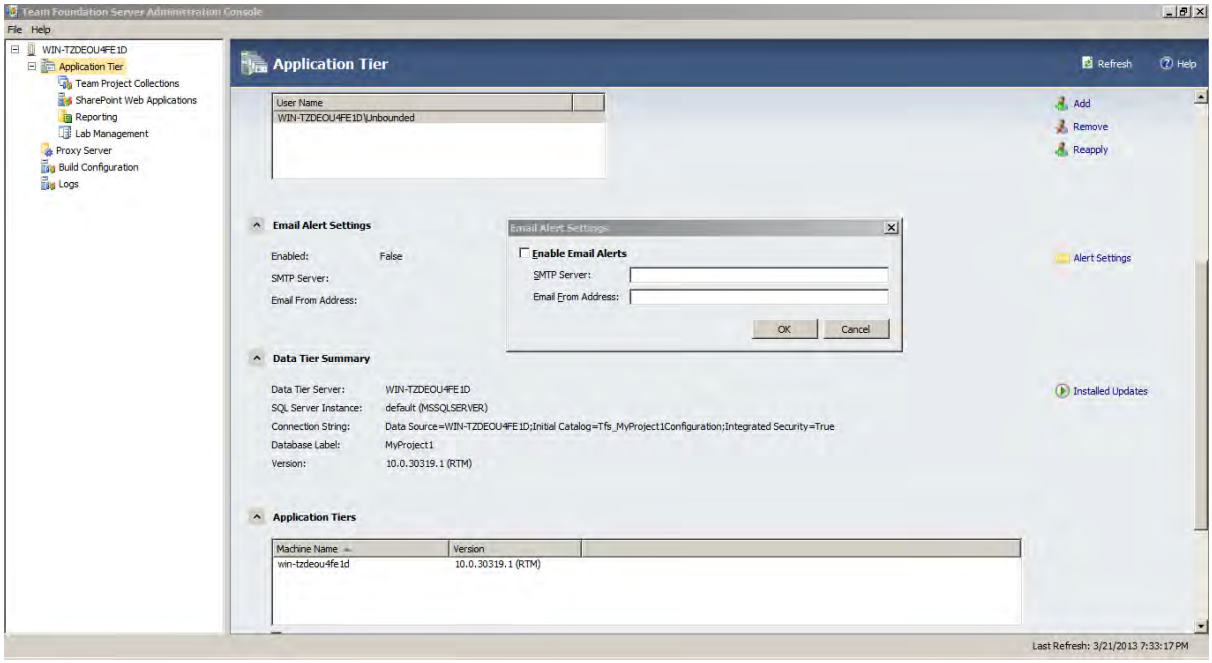


شکل 10-8: پنجره تنظیم آلارم در پروژه

تنظیم این سرویس بسیار ساده می باشد. پنج نوع آلارم(رویداد) در این سرویس وجود دارد. برای فعال شدن هر کدام باید آن آلارم علامت زده شود. در ستون دوم (بخش Send to) باید ایمیل اعضایی که قرار است توسط این آلارم مطلع شوند وارد شود. این ایمیل ها باید با علامت ";" از یکدیگر جدا شوند. در ستون آخر نوع ایمیل ارسالی تعیین می شود. نوع ایمیل می تواند متن خالی یا HTML باشد. پس از ثبت تنظیمات برای اِعمال آنها کافی است روی دکمه OK کلیک شود.

تنظیمات Alert

برای استفاده از قابلیت Alert ابتدای باید آن را فعال نماییم و سرور آن را برای TFS تعریف نماییم. برای اِعمال این تنظیمات باید از Team Foundation Server Administration Console استفاده شود. در این پنجره باید در بخش Application Tier روی گزینه Alert Setting کلیک شود. این عمل باعث باز شدن پنجره Email Alert Setting خواهد شد(شکل 11-8).



شکل 11-8: تنظیمات ایمیل برای آلارم در کنسول مدیریت TFS

در این پنجره با علامت زدن گزینه Enable Email Alerts قابلیت Alert فعال خواهد شد. از دو فیلد SMTP Server و Email From Address به ترتیب برای ثبت آدرس سرور ایمیل و ثبت آدرس ایمیلی که از آن به سایر اعضا ایمیل فرستاده می شود، استفاده می شود. با اِعمال این تنظیمات قابلیت Alert آماده استفاده می باشد.

فصل نهم: آیتم های کاری

در فصل چهارم آیتم های کاری موجود در اسکرام به تفصیل شرح داده شدند. در این فصل به تشریح این آیتم ها در TFS و چگونگی مدیریت آنها خواهیم پرداخت. از نظر TFS هر آنچه در اسکرام وجود دارد اعم از اسپرینت، آیتم های بک لاگ و حتی Splike یک آیتم کاری محسوب می شود. برای کار با هر کدام، باید ابتدا نمونه ای از آنها ایجاد نمود. برای ایجاد آیتم های کاری از دو شیوه می توان استفاده نمود:

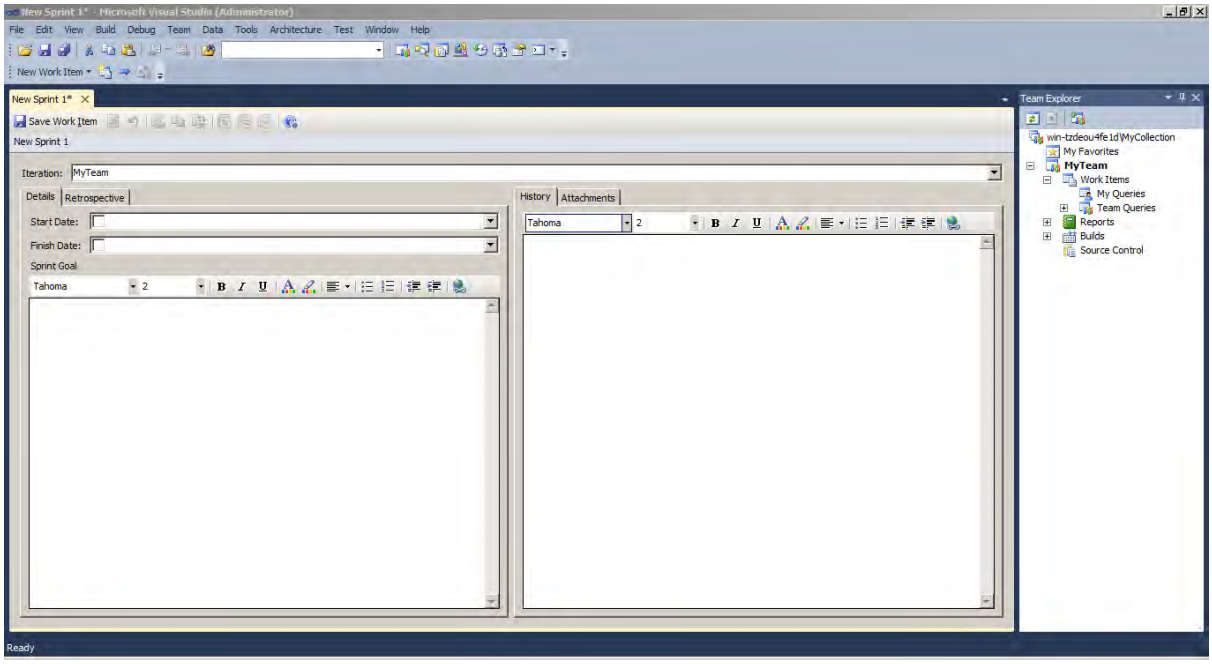
1. استفاده از New Work Item در منوی Team
2. راست کلیک بر روی بخش Work Item و انتخاب گزینه New Work Item

هر دو روش منجر به باز شدن زیر منویی خواهند شد که هفت آیتم کاری را ارائه می دهند. با کلیک روی هر کدام از آنها پنجره خاصی گشوده خواهد شد. این هفت آیتم را در ادامه فصل به تفصیل معرفی خواهند شد.

اسپرینت

اسپرینت ها در TFS به عنوان یک آیتم کاری شناخته می شوند. برای هر اسپرینتی که برنامه ریزی می شود باید یک آیتم کاری وجود داشته باشد. ایجاد آیتم اسپرینت به وسیله کلیک روی گزینه Sprint در زیر منوی New Work Item صورت خواهد پذیرفت. با این عمل پنجره ای نمایان می شود که به شکل تب(زبانه) می باشد. هر آیتمی در TFS دارای عنوان یا Title است. اگر آیتمی جدید باشد و عنوانی برای آن انتخاب نشده باشد با کلمه New در ابتدا، نام آیتم در میانه و یک شماره ترتیبی در انتها نمایش داده خواهد شد. متفاوت با آنچه تاکنون در TFS دیده ایم تب ها دارای کلید ذخیره می باشند. هر کدام از تب ها مرتبط با یک آیتم کاری هستند و اعمال تغییرات در آنها مستلزم ذخیره سازی آنها است. با آیتم های کاری در TFS مانند سندی رفتار می شود که فیلدهای از پیش تعریف شده ای دارند. اگر تغییری در یک تب(آیتم کاری) ایجاد شود کنار عنوان آن علامت * مشاهده می شود. این علامت به این معنی است که تغییرات، هنوز ذخیره نشده اند.

اگر به یک آیتم جدید در اسپرینت نگاهی بیاندازیم در می یابیم که در تب این آیتم اثری از فیلد عنوان وجود ندارد. در حقیقت اسپرینت ها را نمی توان نام گذاری کرد. پس از ذخیره آنها با یک شماره ترتیبی (ID) قابل دسترس خواهند بود(شکل 1-9).



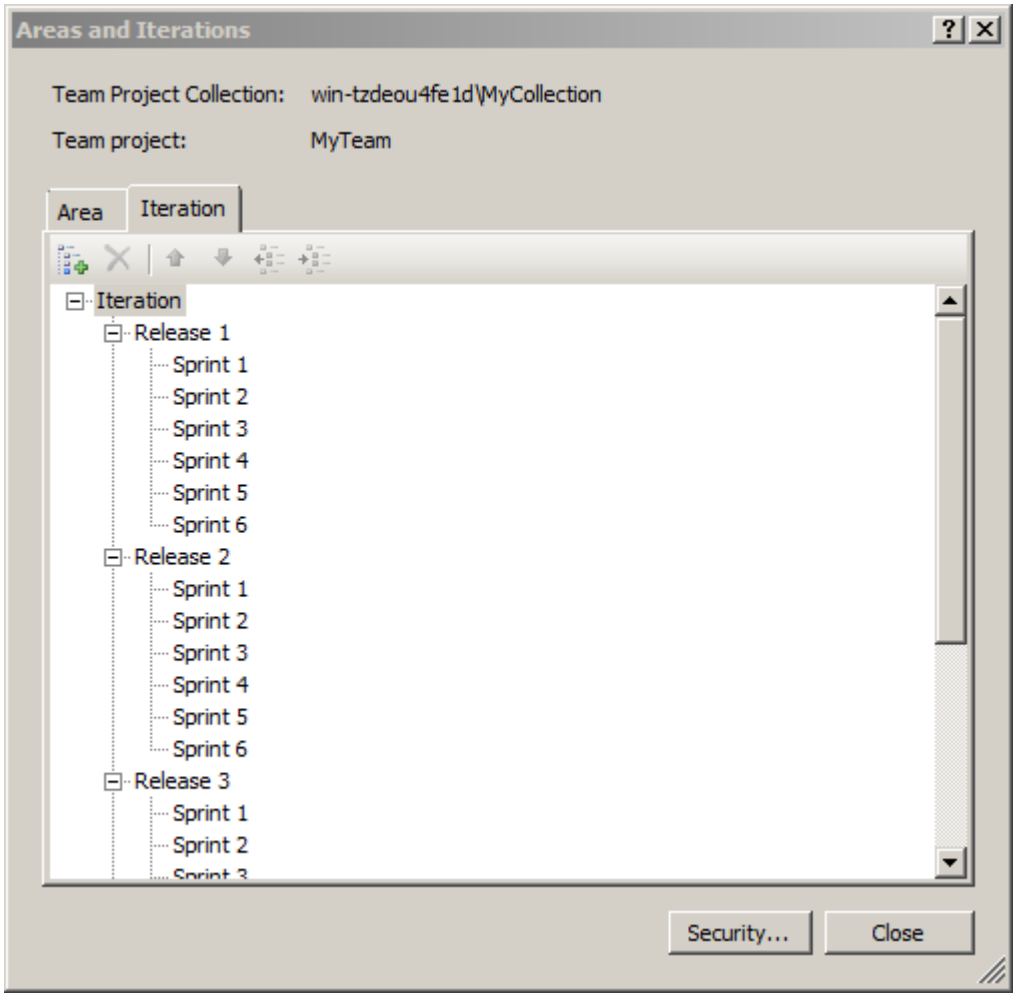
شکل 1-9: تب اسپرینت جدید

اولین و مهمترین فیلدی که در تب اسپرینت قرار دارد Iteration می باشد. Iteration ها در واقع همان ترکیب Release/Sprint در اسکرام می باشند. به صورت پیش فرض TFS دارای تعدادی Iteration از پیش تعریف شده است. TFS از همان ابتدا چهار Release که هر کدام از آنها دارای شانزده اسپرینت هستند را ارائه می دهد.

Iterationها صرفاً سلسله مراتبی از نام ها هستند که مسیر تکرارها در توسعه را ارائه خواهند داد. آنها هیچ موجودیت مستقلی به عنوان یک آیتم کاری از خود ندارند. و در حقیقت آنها به عنوان قالب هایی برای آیتم های کایر واقعی استفاده می شوند. هر کدام از اسپرینت هایی هایی که در Iterationها تعریف می شوند باید به یک آیتم کاری اسپرینت متصل شوند.

Iterationها

علاوه بر آنچه به صورت پیش فرض در TFS برای Iterationها تعریف شده، می توان تکرارهای جدیدی نیز تعریف نمود. برای این کار باید با رفتن به مسیر Team->Team Project Setting گزینه ...Area and Iterations انتخاب شود. انتخاب این گزینه موجب باز شدن پنجره Area and Iterations خواهد شد. این پنجره شامل دو تب می باشد:Area و Iteration(شکل 2-9).

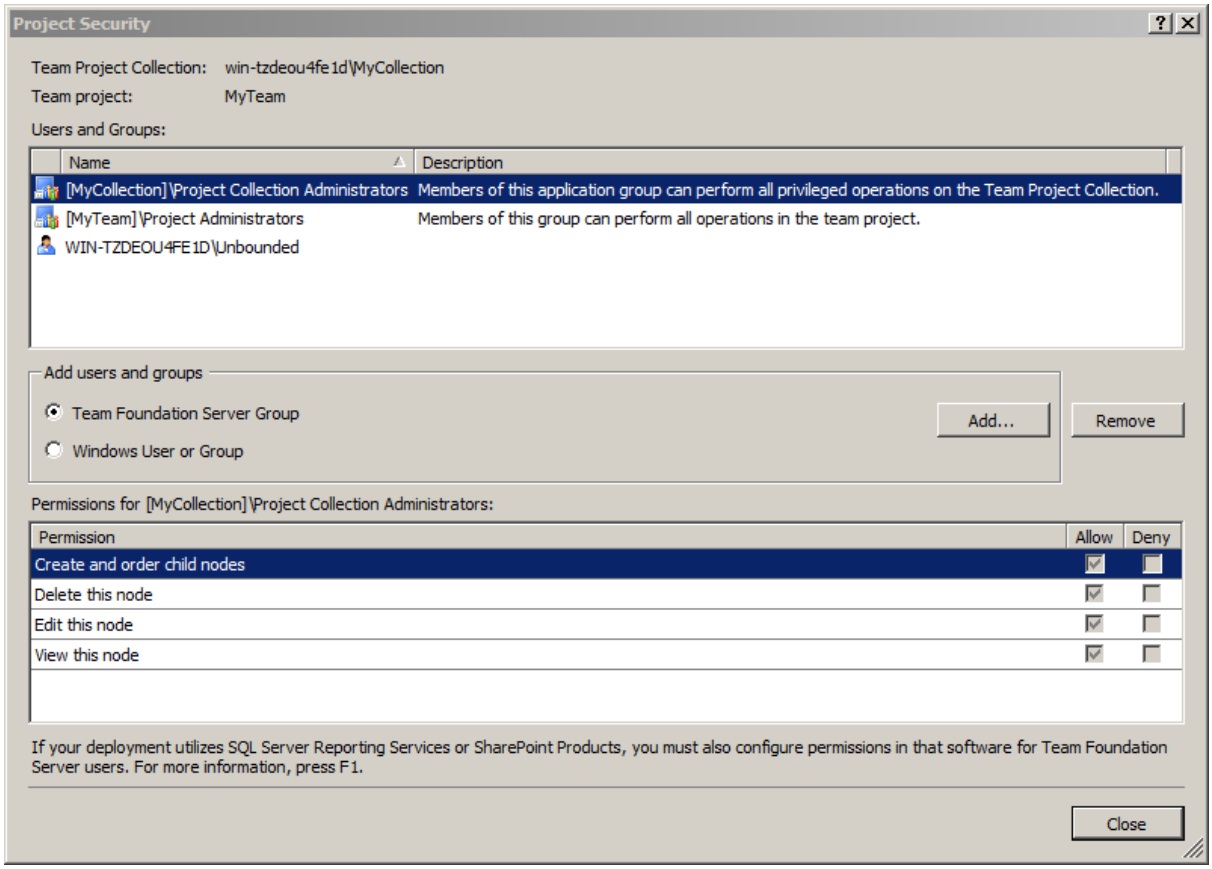


شکل 2-9: پنجره تکرارها و سطح ها

با استفاده از تب Iteration قادر خواهیم بود که در سلسله مراتب تکرارها تغییر ایجاد کنیم. واژه Iteration در بالاترین سطح سلسله مراتب به معنی ریشه می باشد. در سلسله مراتب پس از آن چهار Release قرار دارد. برای Releaseها تنها به داشتن نامی در این سلسله مراتب بسنده شده است و آنها دارای آیتم های کاری مجزا نمی باشند. در این سلسله مراتب برخی از آیتم ها والد و برخی دیگر فرزند می باشند. Releaseها به عنوان فرزندان ریشه (Iteration) و اسپرینت ها به عنوان فرزندان Releaseها محسوب می شوند. برای اضافه کردن یک فرزند کافی است والد آن را انتخاب نموده و سپس بر روی کلید Add a child node در نوار ابزار کلیک نمود. بدین وسیله یک آیتم جدید ایجاد خواهد شد. نام پیش فرض آن واژه Iteration به علاوه یک شماره ترتیبی است. در بخش نوار ابزار دکمه هایی جهت حذف، جابجایی و تنظیم تورفتگی آیتم ها وجود دارد. هر تغییری در این صفحه بلافاصله اعمال می شود.

تنظیمات امنیتی Iterationها

در پایین صفحه دکمه ای وجود دارد به نام Secure که فشردن آن باعث باز شدن صفحه Project Security می شود(شکل 3-9).



شکل 3-9: پنجره تنظیمات امنیتی پروژه

این پنجره مجوزهای گروه ها را برای حذف و اضافه کردن تکرارها تنظیم خواهد کرد. به صورت پیش فرض تنها گروه هایی که مجوز همه عملیات مرتبط با تکرارها را دارند، گروه Project Collection Administrators و گروه ه Project Administrators هستند. البته مانند حالت پیشین از طریق کلید Add نیز می توان به اضافه کردن گروه های دیگر موجود در TFS، گروه های تعریف شده در ویندوز و کاربران پرداخت چهار مجوزی که در این بخش وجود دارد عبارت اند از:

Create: ساختن و مرتب سازی فرزندان

Delete: حذف گره جاری

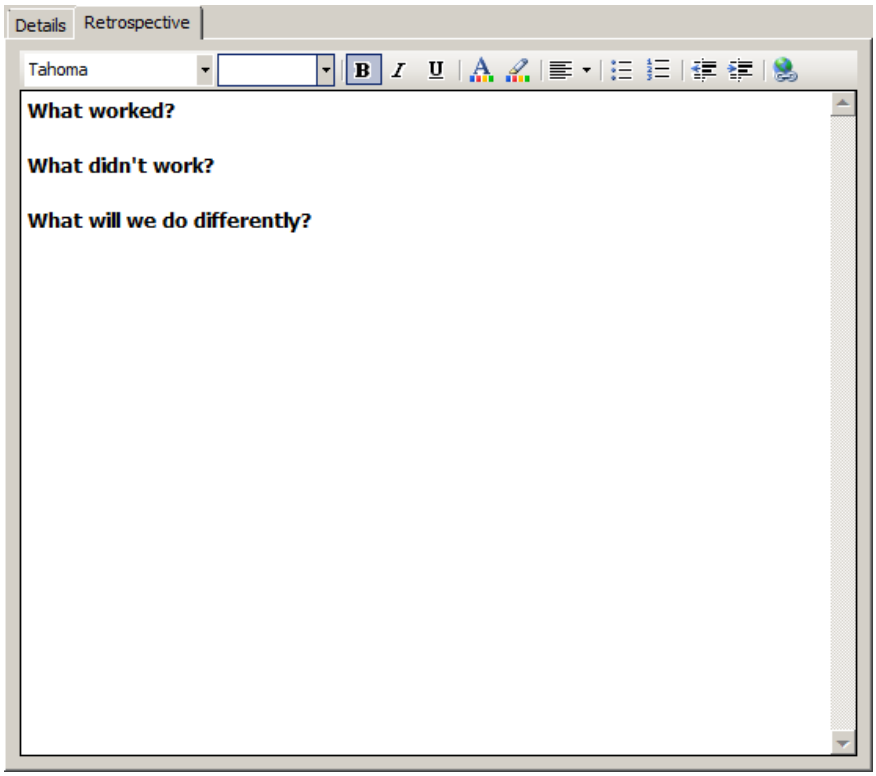
Edit: ویرایش گره جاری

View: مشاهده گره جاری

حال پس از تعیین سلسله مراتب اسپرینت ها، باید به هر کدام یک آیتم واقعی نسبت داده شود. این ارتباط از طریق فیلد Iteration در تب آیتم اسپرینت انجام می پذیرد. برای این عمل کافی است از طریق لیست Iteration، تکرار مورد نظر برای آیتم اسپرینت انتخاب شود. با این روش می توان همه اسپرینت های موجود در سلسله مراتب را صاحب یک آیتم واقعی کرد. توجه شود که اسپرینت هایی که از قبل در TFS تعریف شده اند همگی به طور پیش فرض دارای آیتم اسپرینت می باشند. اما برای سایر اسپرینت های جدید باید یک آیتم اسپرینت ساخته شود.

تب آیتم اسپرینت دارای دو پنل می باشد. هر کدام از این پنل ها خود دارای دو قسمت مجزا هستند. تب جزئیات در پنل سمت چپ، تنظیمات بیشتری را برای آیتم اسپرینت ارائه می دهد. ابتدای این تب دو فیلد Start Date و Finish Date وجود دارد که تاریخ ابتدا و انتهای اسپرینت را مشخص می کنند. در پایین این دو فیلد ابزار ورودی متنی قرار دارد که می توان از طریق آن جزئیات و توضیحاتی راجع به اسپرینت ثبت کرد. این جعبه ابزار، ظاهری شبیه به Wordpad ویندوز دارد و امکانات نوشتاری زیادی را مهیا می سازد.

تب دوم از پنل سمت چپ Retrospective نام دارد. این تب فقط دارای یک ابزار ورودی متن می باشد. این ورودی متن پس از پایان یافتن اسپرینت تکمیل می شود. در این بخش تیم، خروجی جلسه بازبینی را درج می کند. به صورت پیش فرض در ابزار سه جمله به صورت ضخیم (Bold) نوشته شده است(شکل 4-9):



شکل 4-9: تب Restrodpectivr در آیتم اسپرینت

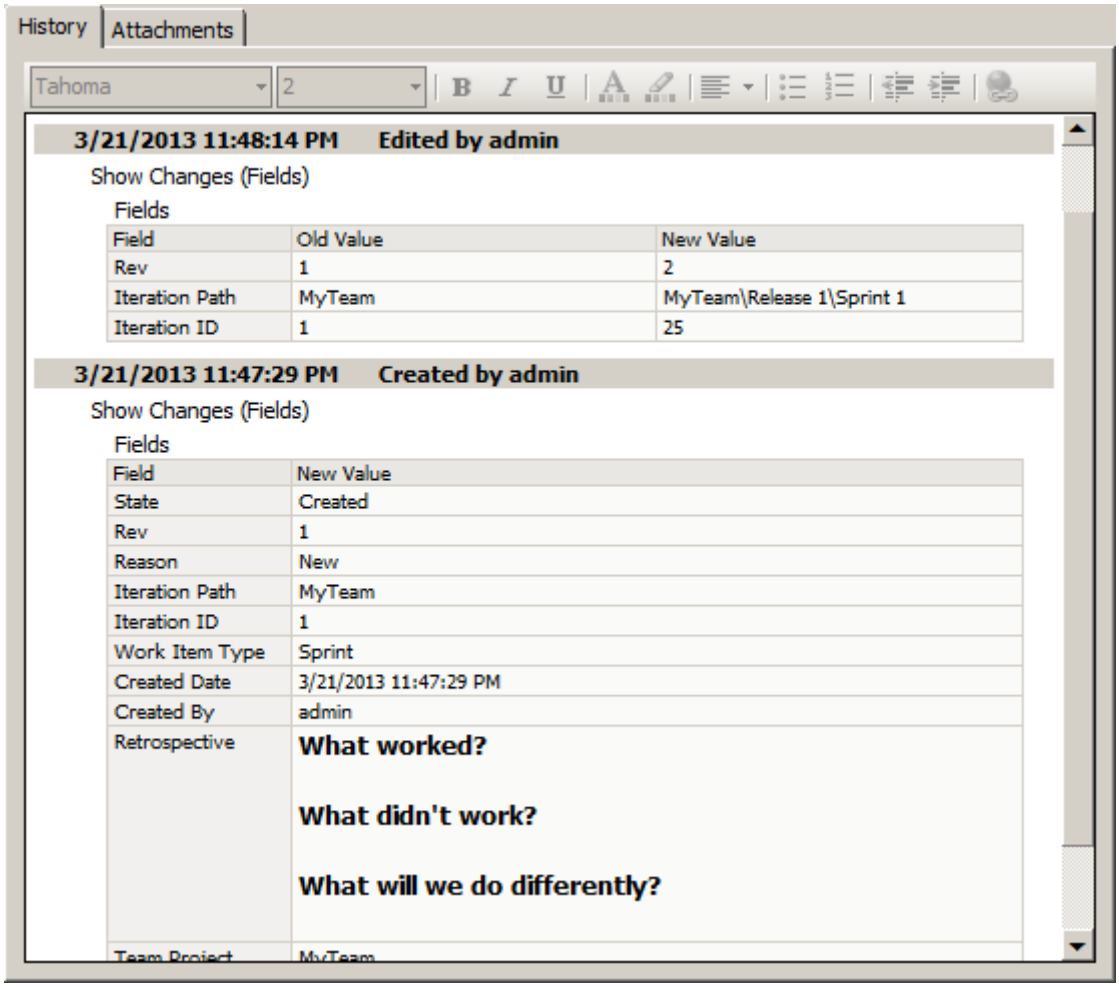
What worked? : چه کارهایی انجام شده

What didn't work? : چه کارهایی انجام نشده

What will we do differently? : چه کارهایی باید متفاوت انجام شود

این سه جمله یادآور خروجی سه بخشی جلسه بازبینی اسکرام است. از آنجایی که خروجی جلسات جهت ردگیری و بایگانی باید ثبت شوند، این بخش بهترین راه برای ثبت خروجی جلسه بازبینی است.

در پنل سمت راست نیز دو تب وجود دارد. تب اول History نام دارد. این تب صرفاً برای نمایش اطلاعات استفاده می شود و کاربر نمی تواند در آن ورودی خاصی ثبت کند. بخش History بخش مهمی برای مدیریت پروژه به حساب می آید، چرا که موجب تسهیل مستند سازی و ردگیری کل پروژه می شود. هر تغییری که در پروژه و آیتم های آن ایجاد شود به تفصیل در بخش History ثبت و نگهداری خواهد شد. این بخش برای همه آیتم های اسکرام به صورت مجزا وجود دارد. هر آیتم در طول حیات خود ممکن است دست خوش تغییرات زیادی شود. این تغییرات جزء به جزء برای آن آیتم در بخش History مربوط به خود ثبت و نگهداری می شود. این بهترین راه مستند سازی کل پروژه آن هم با جزئیات فراوان است(شکل 5-9).



شکل 5-9: تب History در آیتم اسپرینت

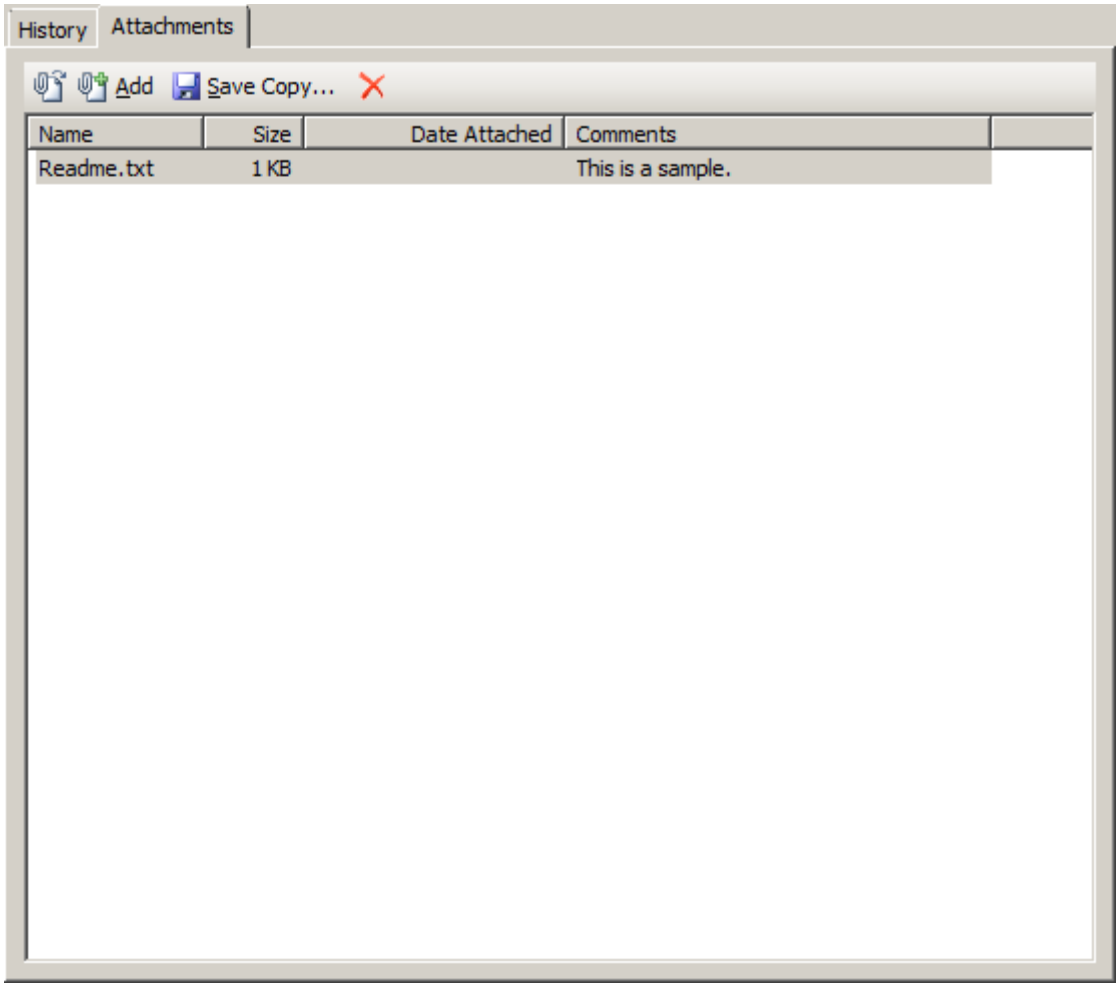
به طور کلی دو نوع تاریخچه برای یک آیتم وجود دارد. تاریخچه ثبت و تاریخچه ویرایش. تاریخچه ثبت اولین تاریخچه ای است که در این لیست ثبت می شود. این تاریخچه با ایجاد آیتم کاری به صورت پیش فرض ایجاد می شود. ثبت شدن این نوع تاریخچه منوط به ذخیره آیتم کاری می باشد. پس از اولین ذخیره سازی بخش Create ایجاد می شود. در عنوان این نوع تاریخچه تاریخ و زمان رویداد، کلمه Create و کاربری که آن را ایجاد کرده است ثبت می شود. هر تاریخچه علاوه بر عنوان دارای آیتم هایی نیز می باشد که به صورت سلسله مراتبی مرتب شده اند. آنچه در تاریخچه Create همواره ایجاد می شود آیتم فیلدها (Fields) می باشد. همه آیتم های فیلد به عنوان Show حساب می شوند. با باز کردن این مجموعه یک جدول دو ستونی ظاهر خواهد شد. ستون اول نام فیلدها و ستون دوم مقداری است که اولین بار به فیلدها نسبت داده می شود. همه فیلدهایی که در ایجاد یک آیتم به صورت پیش فرض مقار دارند در این جا به همراه مقدارشان ظاهر می شوند. فیلدهایی که مقدار پیش فرضشان Null است در این جدول ظاهر نخواهند شد. همانطور که گفته شد مقادیر فیلدها فقط خواندنی می باشد و امکان تغییر در آنها وجود ندارد.

نوع دیگری از تاریخچه ها Edit (ویرایش) نام دارد. این نوع از تاریخچه ها نیز به وسیله ذخیره کردن آیتم ثبت می شوند. تفاوت آنها با نوع Create در این است که با هر اِعمال تغییری یک نمونه از این تاریخچه ثبت خواهد شد. عنوان این تاریخچه ویرایش نیز مانند Create می باشد با این تفاوت که به جای کلمه Create واژه Edit درج می شود. تاریخچه ویرایش دارای آیتم های بیشتری نسبت به Create می باشد. آیتم Fields در این نوع تاریخچه، تغییرات اِعمال شده در فیلدها را آشکار می کند. آیتم Fields دارای یک جدول سه ستونی است. ستون اول (سمت چپ) نام فیلد، ستون دوم مقدار پیشین و ستون سوم مقدار جدید می باشد. تغییر در هر فیلدی ازاین طریق قابل پیگیری است. علاوه بر آیتم Fields آیتم های دیگری چون Attachment نیز در تاریخچه ویرایش قابل ثبت است. Attachment قابلیتی برای ضمیمه کردن فایلی به یک آیتم می باشد. با هر ضمیمه سازی، یک تاریخچه ویرایش برای آن به ثبت خواهد رسید. اما این بار این ثبت تاریخچه متفاوت تر از قبل با جدولی چهار

ستونی همراه است. عنوان این آیتم در این حالت Attachment قرار خواهد گرفت. ستون های این جدول عبارت اند از: نام فایل(به همراه لینکی برای اجرای مستقیم آن)، اندازه فایل، توضیحات و حالت تغییر. حالت تغییر یک عبارت یک کلمه است که حالت تغییر را ارائه می دهد.

علاوه بر آنچه به صورت خودکار ثبت می شود کاربران خود می توانند توضیحات را درج کنند. در بالای بخش تاریخچه عبارتی به نام .Type your comment here وجود دارد که با کلیک بر روی آن ورودی متنی ارائه می شود که کاربر قادر به درج اطلاعات می کند.

بخش بعدیِ پنل سمت راست Attachment نام دارد(شکل 6-9).



شکل 6-9: تب Attachments در آیتم اسپرینت

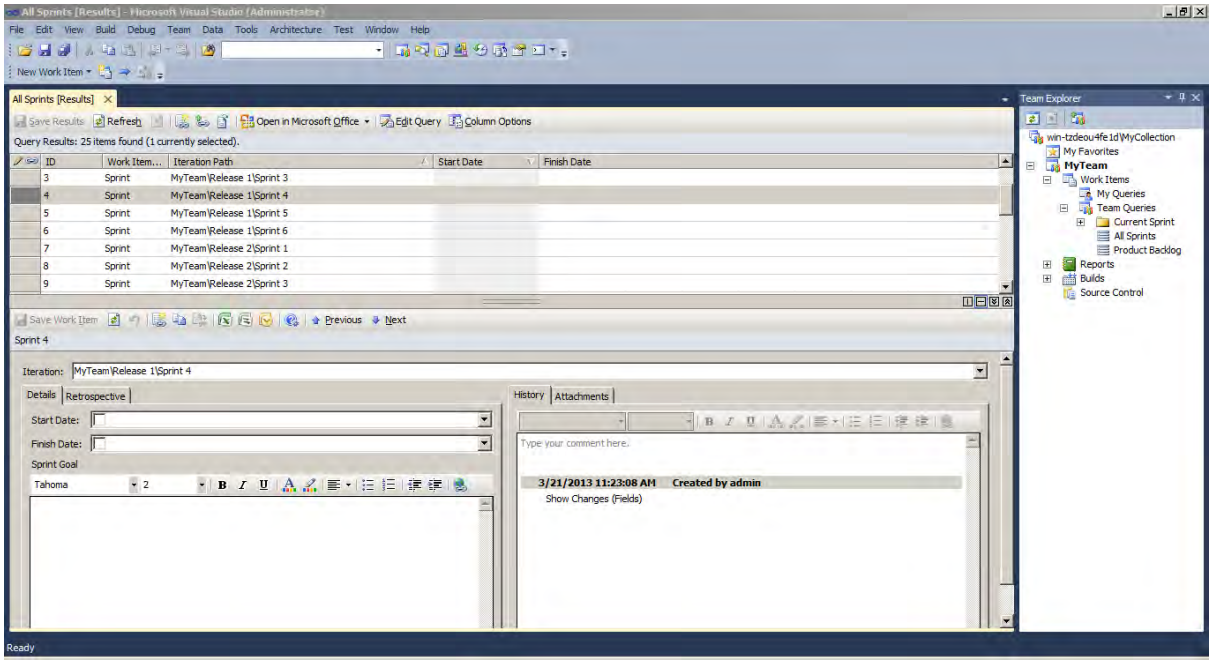
همانطور که پیش از این گفته شد این امکان به کاربر اجازه می دهد که فایل هایی را به عنوان ضمیمه به اسپرنیت متصل نماید. هر فایلی با هر فرمتی را می توان در این لیست به اسپرینت پیوند داد. هر کدام از فایل ها سطری را به لیست اضافه می کنند. در این سطر اطلاعات از قبیل نام فایل ، اندازه فایل، تاریخ ضمیمه و توضیحات وجود دارد. در قسمت فوقانی لیست جعبه ابزاری وجود دارد که دارای چهار کلید می باشد. اولین کلید (از سمت چپ) باعث اجرای فایل منتخب از لیست Attachment خواهد شد. دومین کلید برای اضافه کردن یک فایل جدید مورد استفاده قرار می گیرد. با کلیک روی این کلید پنجره Attachment جهت انتخاب فایل گشوده خواهد شد. این پنجره دارای دو فیلد Attachment و Comment می باشد. کلیک روی دکمه ...Browse از بخش Attachment باعث باز شدن پنجره دیالوگی خواهد شد که از طریق آن می توان فایل مورد نظر را انتخاب نمود. پس از انتخاب فایل و فشردن کلید Open پنجره دیالوگ بسته خواهد شد و مسیر فایل در فیلد Attachment ظاهر خواهد شد. در فیلد Comment نیز می توان توضیحات مرتبط با فایل افزود. با کلیک روی OK پیوندی از فایل انتخابی ساخته شد و در لیست Attachment قرار خواهد گرفت. با استفاده از کلید ...Save نیز می توان یک

نسخه از فایل انتخابی را در مسیر مورد نظر کپی کرد. کلید ...Delete نیز بدیهاً باعث حذف یک ضمیمه از لیست و حذف پیوند آن با اسپرینت خواهد شد.

با درج یک آیتم اسپرینت به بخش سلسله مراتبی تکرارها باعث می شود که هر آیتمی که به یک تکرار اشاره می کند به صورت خودکار به آیتم اسپرینت اضافه شده متصل شود. لازم به ذکر است با حذف یک تکرار آیتم تکرار حذف نخواهد شد و فقط لینک(پیوند) آن برداشته می شود. یک آیتم اسپرینت را علاوه بر بخش تکرارها از طریق ID آن می توان مورد دسترسی قرار داد.

نکته: آیتم های کاری را پس از ایجاد به هیچ طریقی نمی توان حذف نمود. در نگاه اول ممکن است این موضوع یک محدودیت فنی عمدی از جانب TFS تلقی می شود. از نظر مستند سازی هر آیتمی نقشی در توسعه ایفا می کند باید بایگانی شود. از آنجایی که در TFS خود آیتم ها به عنوان مستندات به کار می روند پس هیچ گاه نباید آنها را از بین برد و تنها باید به قطع ارتباطشان با دیگر آیتم ها اکتفا نمود. بدهی است آیتم هایی که با دیگران پیوند ندارند، دیده نخواهند شد. برای مشاهده این نوع آیتم ها باید از امکانات کوئری (درخواست) استفاده کرد. کوئری ها با اخذ پارامترهایی می توانند لیستی از آیتم های مرتبط را برگردانند. در فصل آینده مفصلاً به تشریح کوئری ها خواهیم پرداخت.

برای مشاهده لیست اسپرینت ها می توان از بخش All Sprints واقع در Work Items استفاده کنیم. این کار باعث باز شدن تبی دوگانه می شود(شکل 7-9).

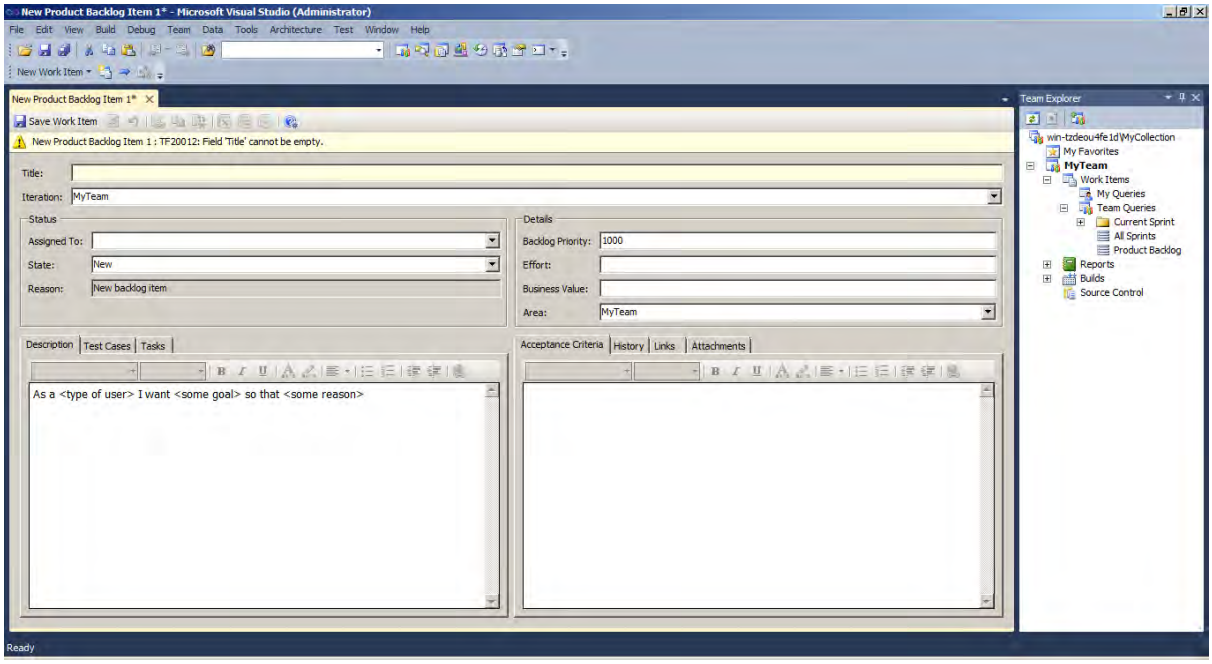


شکل 7-9: اجرای کوئری ذخیره شده All Sprint

بخش فوقانی تب اسپرینت ها را به صورت جدولی (مانند بانک اطلاعات) نشان می دهد. هر سطر بیانگر یک اسپرینت می باشد. این جدول شامل ستون هایی است که نماینده فیلدهای آیتم اسپرینت می باشند. در بخش پایین تب پنجره آیتم (اسپرینت وجود دارد (مشابه آنچه در بخش ایجاد آیتم اسپرینت مشاهده شد). هر کدام از سطر ها(اسپرنیت ها) که انتخاب شود، پنجره آیتم اسپرینت مخصوص به آن برای ویرایش احتمالی در بخش پایینی تب ظاهر می شود. در صورتی که روی یکی از این سطرها دو بار کلیک شود پنجره آیتم اسپرینت برای آن تب به صورت جداگانه باز خواهد شد.

PBI

آیتم بک لاگ مهمترین آیتم TFS است و دارای فیلدهای زیادی می باشد. برای ایجاد یک PBI جدید مانند آیتم اسپرینت باید به زیر منوی Team->New Work Item رجوع کرد و گزینه Product Backlog Item را انتخاب نمود. پس از انجام این عمل تب جدیدی باز می شود که در عنوان آن New Product Backlog به علاوه یک شماره ترتیبی نوشته شده است(شکل 8-9).

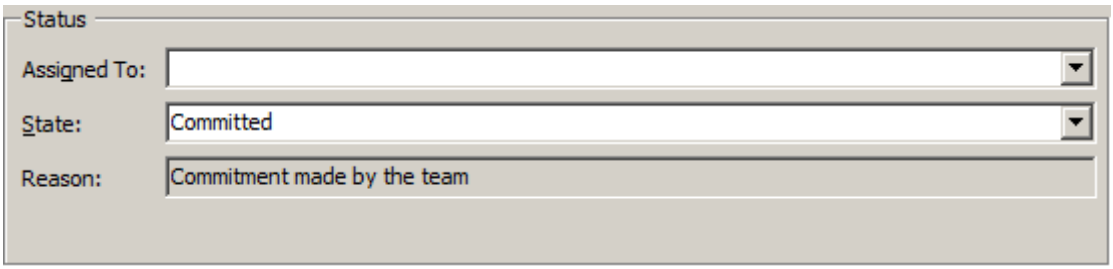


شکل 8-9: تب جدید PBI

اولین موردی که جلب توجه می نماید پیغام اخطاری است که بلافاصله پس از باز شدن تب در بخش فوقانی آن ظاهر می شود. TFS با استفاده از این پیغام، هشدار می دهد که عنوان آیتم جاری نمی تواند خالی باشد. برخلاف آیتم اسپرینت، آیتم بک لاگ حصول علاوه بر ID دارای یک عنوان منحصر به فرد نیز می باشد.

فیلد دوم این آیتم کاری Iteration می باشد. این فیلد در اینجا تفسیری متفاوت از آنچه در آیتم اسپرینت مشاهده شد دارد. Iteration برای هر آیتم به عنوان هویت آن آیتم تلقی می شود. مقدار Iteration همه آیتم ها به صورت پیش فرض نام تیم می باشد. این امر برای PBI ها باعث می شود که مستقیماً درون بک لاگ محصول قرار گیرند. برای انتقال یک PBI از بک لاگ محصول به بک لاگ اسپرینت کافی است مقدار Iteration آن آیتم برابر اسپرینت مورد نظر قرار گیرد. (دلیل این تغییر مکان در فصل کوئری ها شرح داده می شود). از آنجایی که اولین مکان هر PBI پس از ایجاد، بک لاگ محصول است بنابراین این فیلد نباید تغییر کند این فیلد معمولاً هنگامی تغییر خواهد کرد که آیتم به یک بک لاگ اسپرینت نسبت داده شود.

سایر فیلدهای آیتم بک لاگ در دو پنل وضعیت و جزئیات قرار می گیرند. اولین آیتم پنل وضعیت Assignment می باشد(شکل 9-9).



شکل 9-9: بخش Status در PBI

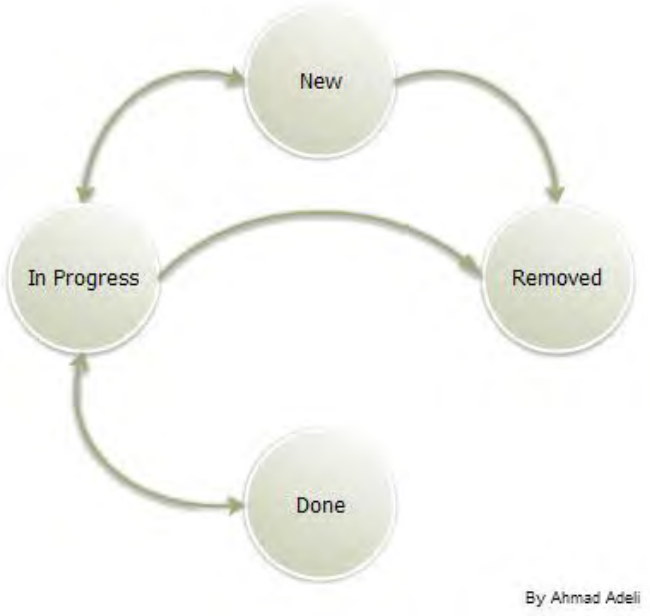
مقدار این فیلد نام یکی از اعضای موجود در TFS است. این فیلد مشخص می کند که کدام یک از اعضا مسئول انجام این آیتم می باشند. با کلیک روی فیلد Assignment لیستی از اعضا TFS را نشان داده خواهد شد. محدودیت Assignment این است که اگر یک آیتم به بیش از یک نفر اختصاص یابد فقط می توان یک نفر را به عنوان مقدار این فیلد تعیین نمود.

توسط فیلد State می توان تعیین کرد یک آیتم در چه مرحله ای از توسعه قرار دارد. در واقع حالت آیتم در توسعه توسط مقادیر این فیلد شرح داده می شود. اولین حالتی که یک آیتم در آن قرار می گیرد حالت New می باشد.

این حالت برای زمانی صدق می کند که آیتم در دست طراحی است و هنوز از سوی مالک محصول برای ورود به بک لاگ تأیید نشده است. در لیست فیلد State حالت های متفاوتی وجود دارد اما در وهله اول و پیش از ذخیره سازی فقط یک حالت New در آن مشاهده می شود. اگر برای بار اول PBI ذخیره شود به لیست فیلد State دو گزینه اضافه می شود. گزینه Approved هنگامی استفاده می شود که آیتم بک لاگ توسط مالک محصول تأیید شود. حالت Approved به معنی آماده بودن آیتم برای ورود به یک اسپرینت می باشد. بنابراین آیتمی می تواند در جلسه برنامه ریزی اسپرینت انتخاب شود که حالت Approved را داشته باشد. حالت دیگری وجود دارد به نام Remove، این حالت در ظاهر باعث حذف یک عنصر می شود. اما از آنجایی که آیتم های ایجاد شده در TFS قابل حذف نمی باشند، حالت Remove عملاً باعث می شود که در آیتم مذکور در بک لاگ، صرفاً مشاهده نشود. این آیتم از طریق کوئری قابل دستیابی است و مجدداً در صورت نیاز می تواند به حالت های دیگری تغییر یابد.

حالت Remove اغلب برای مواردی آیتم هایی استفاده می شود که مدت زیادی قابل استفاده نباشند. حرکت بین حالت ها با نظم خاصی همراه است. به عنوان مثال با انتخاب حالت Approved و ذخیره آیتم حالت های لیست State به New و Removed تغییر می کند. حتی اگر یک آیتم آماده نیز باید نمی تواند از حالت Removed به حالت Approved تغییر یابد. بلکه ابتدا باید حالت را به New تغییر داد سپس آن را به Approved تبدیل کرد. با انتخاب گزینه Approved نیز حالات دیگری به وجود می آید. با انتخاب این حالت لیست State به Approved، Commited و Removed تغییر می یابد. حالت Commited برای زمانی استفاده می شود که آیتم توسط تیم برای یک اسپرینت گزینش شده باشد. آیتم هایی که را که تیم روی آنها کار می کند دارای حالت Commited می باشند. اگر Commited به عنوان حالت فعلی انتخاب شود.

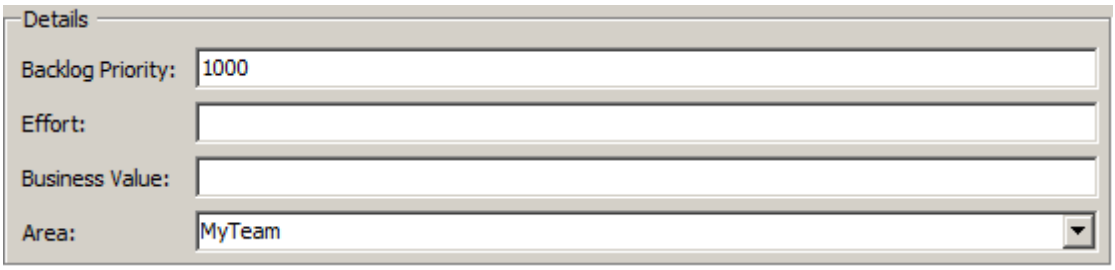
علاوه بر Approved و Commited گزینه دیگری اضافه می شود به نام Done. از حالت Done هنگامی استفاده می شود که کار روی آیتم تمام شده باشد. به عبارتی دیگر هر گاه تعریف Done(تکمیل شده) برای آیتمی صدق کند آن آیتم باید به حالت Done تبدیل شود. اگر نیاز باشد که در میان اسپرینت آیتم ها به بک لاگ محصول برگردانده شودند بدیهی است که ان آیتم ها از حالت Commited به حالت Approved تغییر خواهند کرد. ممکن است پس از انتقال حالت یک آیتم به Done در جلسه تحویل قابلیت آیتم های Done شده رد شوند. در این صورت آیتم باید مجدداً در بک لاگ محصول قرار گیرد. از این رو هنگام قرار گرفتن آیتم در حالت Done حالت Commited هم در لیست State وجود خواهد داشت تا در صورت لزوم مجدداً در بک لاگ اسپرینت وارد شود(شکل PBI States).



شکل PBI States: حالت های مختلف یک PBI

بعد از فیلد State فیلدی فقط خواندنی وجود دارد به نام Reason. این فیلد توضیحاتی را در خصوص هر حالت ایجاد شده نمایش می دهد. توضیحات، از پیش تعریف شده هستند و قابل تغییر نمی باشند.

پنل سمت راست پنل Details نام دارد که گزینه های بیشتری را برای آیتم بک لاگ محصول ارائه می دهد(شکل 10-9).



شکل 10-9: بخش Details در PBI

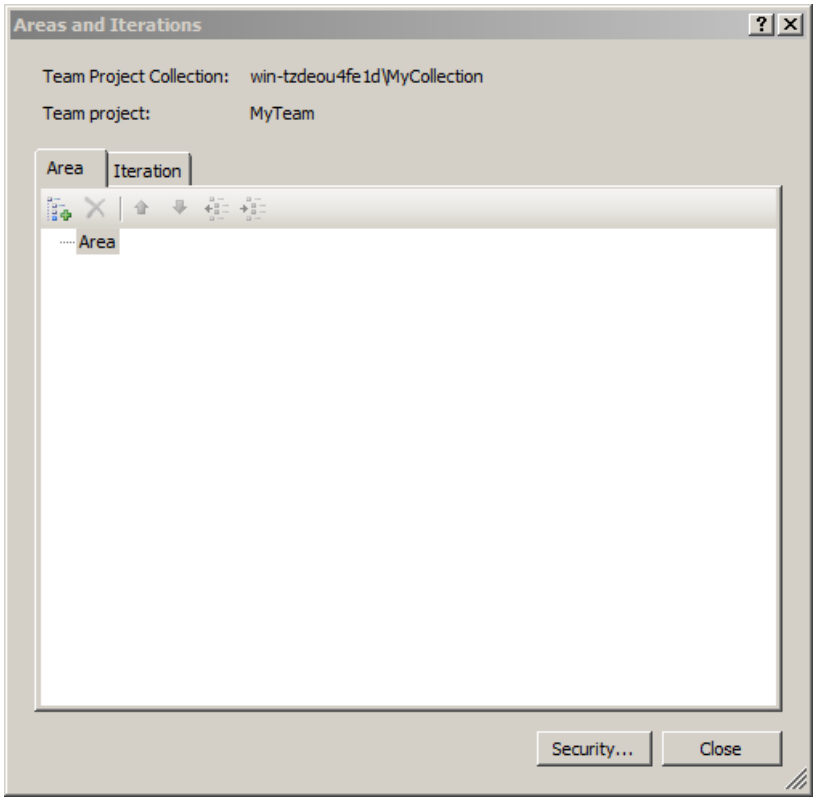
اولین فیلد آن Periority می باشد. از این فیلد برای تعیین اولویت آیتم در بک لاگ استفاده می شود. مقدار این فیلد عددی است و به طور پیش فرض 1000 می باشد.

فیلد Effort را می توان مهمترین فیلد در این بخش نامید. این فیلد همان برآورد آیتم ها بک لاگ محصول می باشد که در جلسه برنامه ریزی اسپرینت توسط تیم توسعه صورت خواهد پذیرفت. این فیلد نیز عددی است و واحد آن معمولاً Story Point می باشد. فیلد Effort و فیلد Business Value هنگامی که حالت آیتم به Done تغییر کند غیر فعال می شوند؛ به این دلیل که دیگر نیازی به تعیین برآورد آن آیتم وجود ندارد.

فیلد Business Value یکی از گزینه های اختیاری در اسکرام می باشد. این فیلد معیاری عددی را برای ارزش تجاری یک آیتم ارائه می دهد. بازه این عدد در اختیار کاربر است اما معمولاً از صفر تا 100 تعیین می شود. این عدد ارزش یک آیتم را از لحاظ تجاری برای توسعه دهندگان یادآور می شود. تعیین این عدد معمولاً وظیفه مالک محصول است. او مقدار این فیلد را از میزان اهمیت و حساسیت مشتریان نسبت به یک قابلیت استخراج می کند. فیلد Business Value یکی از فیلدهایی است که پیش از جلسه برنامه ریزی اسپرینت باید تعیین گردد. لازم به ذکر است که این فیلد معمولاً دارای رابطه ای مستقیم با فیلد Periority است.

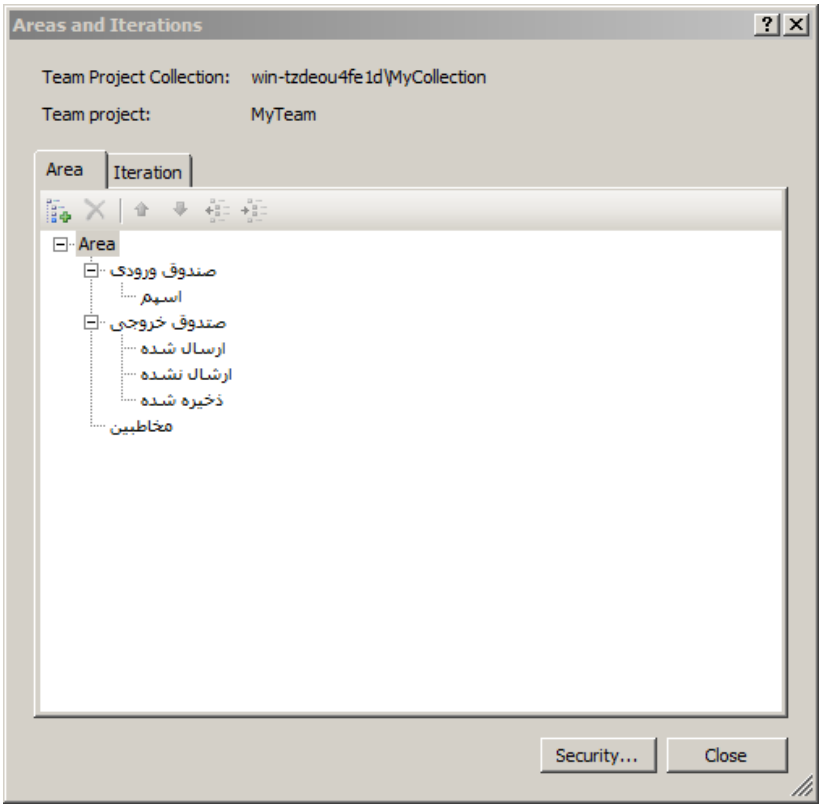
در بخش Details فیلد مفیدی وجود دارد به نام Area. از این فیلد تا به حال در این کتاب سخنی به میان نیامده است. توسعه دهندگان عموماً پروژه تحت توسعه خود را به سطح های مختلفی تقسیم بندی می کنند. این سطوح را می توان همان زیرسیستم ها و یا کامپوننت ها نام نهاد. به عبارت دیگر هر سطح نوعی دسته بندی بخش ها مختلف نرم افزار بر اساس موضوع آنها می باشد. این سطوح سلسله مراتبی هستند و ریشه آن کل نرم افزار است. در هر سطح نرم افزار به بخش های کوچکتری تبدیل می شود، این تفکیک سطح به توسعه دهندگان کمک می کنند تا بدانند که هر آیتم چه جایگاهی در نرم افزار دارد. گاهی نیز برخی تیم ها بر اساس سطوح توسعه دهندگان خود را دسته بندی می کنند. در این گونه تیم ها هر دسته به طور ویژه در یک سطح کار می کند.

برای استفاده از سطوح، ابتدا باید آنها را تعریف نمود. با رفتن به مسیر Team->Team Project Settings->Area and Iterations پنجره Area and Iterations باز خواهد شد (شکل 11-9).



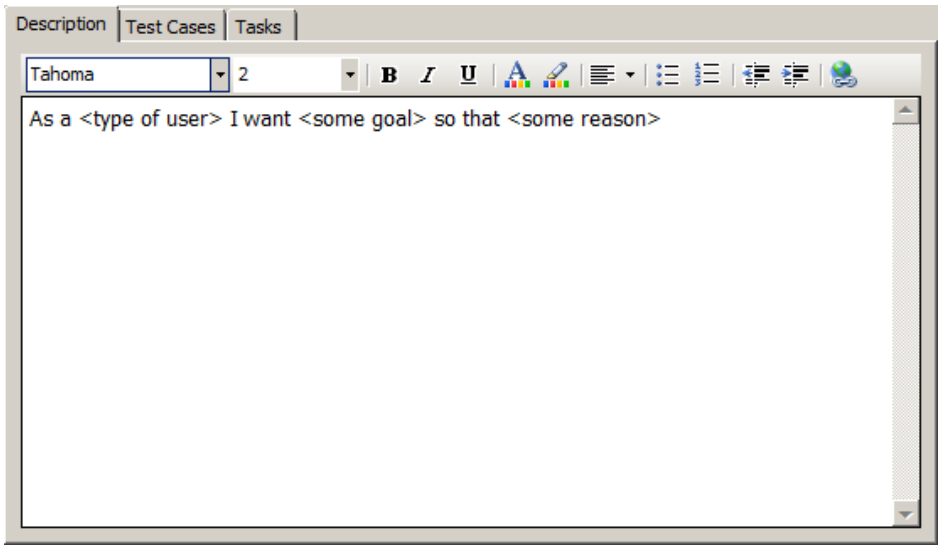
شکل 11-9: پنجره تکرارها و سطح ها

از این پنجره سابق بر این برای تعریف تکرار (Iteration) استفاده کردیم. همان طور که قبلاً گفته شد این پنجره شامل دو تب می باشد. تب Itreative (که قبلاً شرح داده شد) برای تعریف تکرارها استفاده می شود و تب Area که ساختاری مشابه دارد، امکان تعریف Area ها را برای پروژه فراهم می سازد. در ریشه گره ها پیش فرض واژه Area قرار داده شده است. با کلیک روی ابزار Add a child node در نوار ابزار بالا، یک گره (سطح) اضافه خواهد شد. از طریق راست کلیک روی یک گره و یا انتخاب گره و استفاده از ابزارهای بالا می توان اعمالی از قبیل حذف گره، تغییر نام، جابجایی و تورفتگی را انجام داد. پس از تعریف سطح ها به راحتی می توان برای هر کدام از آیتم ها سطحی را تعیین نمود(شکل 12-9).



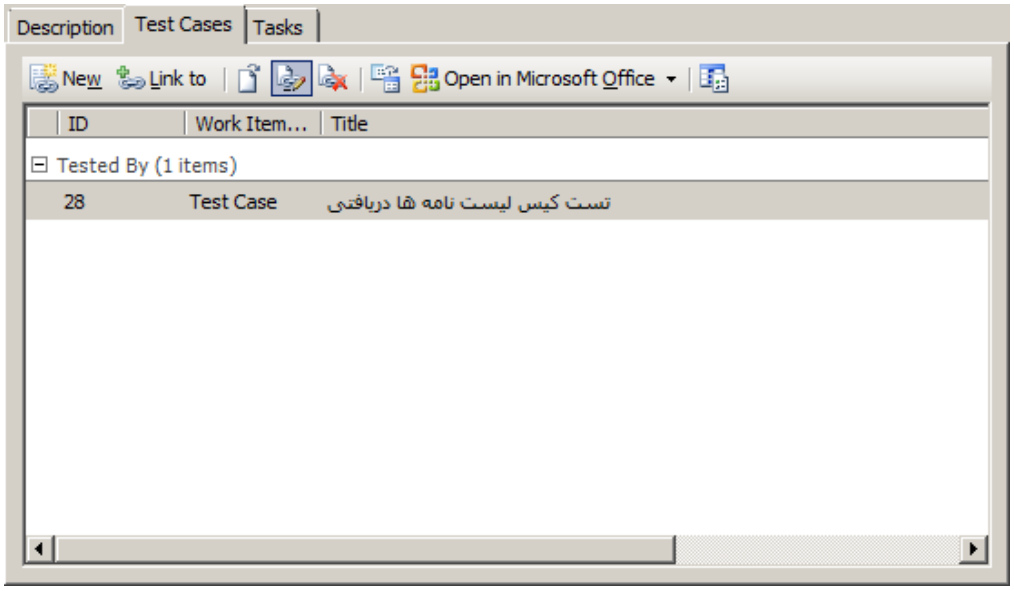
شکل 12-9: سطح های تعریف شده برای پروژه

در بخش پایین پنجره دو پنل وجود دارد که هر کدام دارای تب های متنوعی هستند. پنل سمت چپ دارای سه تب Description ، Test Case و Tasks می باشد. در تب Description یک ورودی متن قرار دارد که برای نوشتن متن توضیحات آیتم مورد استفاده قرار می گیرد(شکل 13-9).



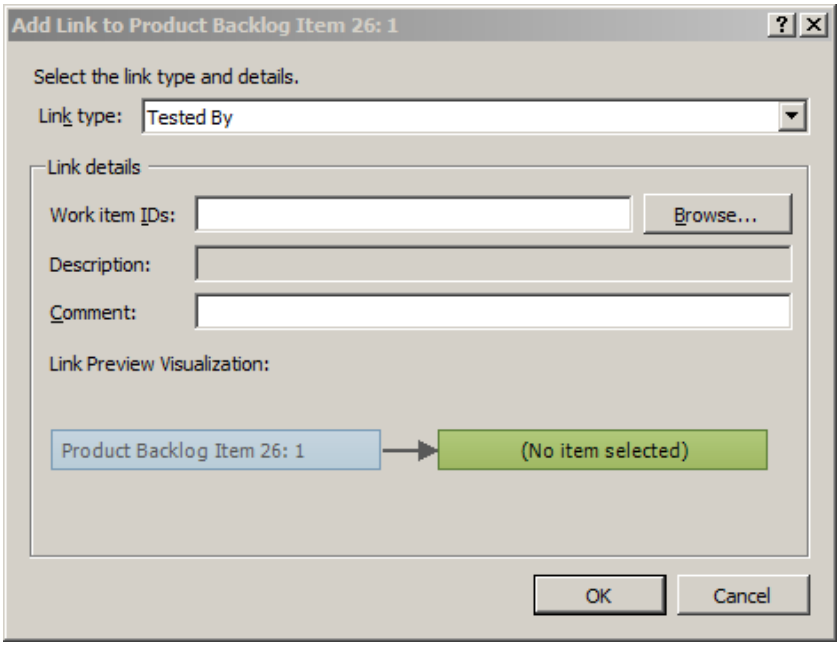
شکل 13-9: تب Decription در PBI

بهتر است برای هر آیتمی توضیحاتی درج شود؛ این عمل در جهت شفاف سازی هدف آیتم ضروری است. از تب Test Case پیوند دادن یک آیتم بک لاگ به آیتم های تست کیس آن استفاده می شود(شکل 14-9).



شکل 14-9: تب Test Cases در PBI

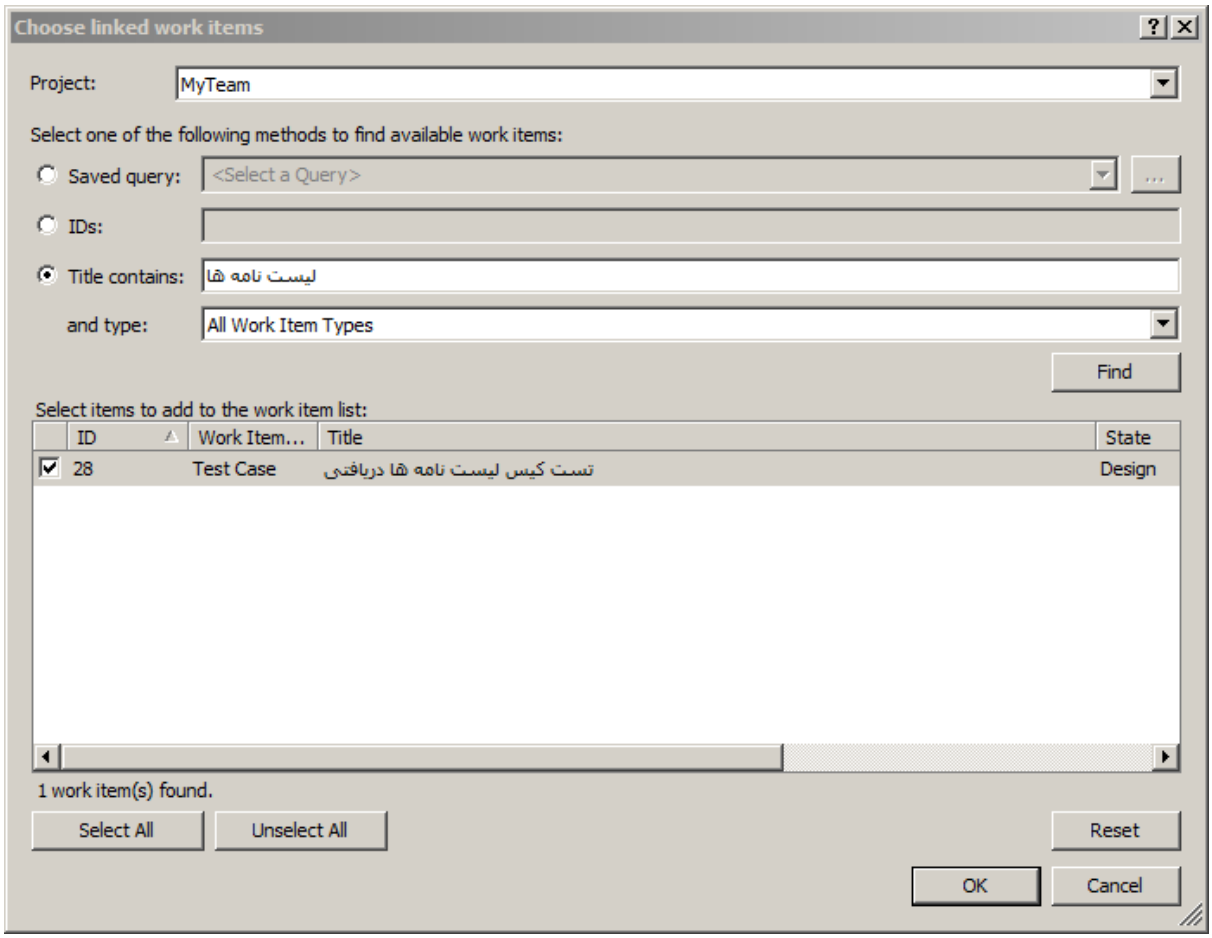
در این تب جدولی وجود دارد که لیست تست کیس های مرتبط با آیتم را ارائه می دهد. هر سطر یک تست کیس را نشان می دهد که دارای سه ستون ID، نوع آیتم کاری و عنوان می باشد. با فرض این که قبلاً برای این آیتم، تست کیسی ایجاد شده باشد می توان از نوار ابزار بالا به وسیله کلید link To... تست کیس مورد نظر را به آیتم پیوند داد. با کلیک این دکمه (link To...) پنجره Add Link to Product Backlog Item گشوده می شود(شکل 15-9).



شکل 15-9: پنجره افزودن لینک به یک آیتم

وظیفه این پنجره که در ادامه به کرات از آن استفاده خواهد شد، پیوند دو آیتم کاری می باشد. پنجره از دو بخش تشکیل می شود؛ بخش ابتدایی یک منوی پایین رونده است که نوع پیوند را مشخص می کند. پیوند زیادی بین دو آیتم وجود دارد اما در این بخش تنها پیوند موجود Test Case می باشد. بخش دوم پنجره Add Link to Product Backlog Item است. این بخش خود از چهار قسمت دیگر تشکیل می شود. اولین قسمت فیلدی است نام Work item IDs. این فیلد ID آیتم تست کیسی که قرار است به آیتم بک لاگ پیوند بخورد را دریافت می کند. در صورت داشتن ID تست کیس مذکور به راحتی می توان پیوند را بر قرار ساخت. اما اگر تست کیس از روی ID قابل شناسایی نباشد می توان از روش های دیگری برای جستجوی تست کیس مورد نظر استفاده

کرد. با کلیک روی دکمه Browse مقابل فیلد، پنجره Choose linked work items باز خواهد شد. این پنجره ابزار مفیدی برای جستجو آیتم های مختلف در TFS می باشد(شکل 16-9).



شکل 16-9: پنجره انتخاب آیتم لینک

پنجره Choose linked work items دارای گزینه مفیدی برای جستجو یک آیتم است که به ترتیب بررسی خواهد شد. اولین قسمتی که در این پنجره به چشم می خورد Project می باشد. از منوی پایین روند مقابل آن می توان پروژه ای که آیتم مورد نظر در آن قرار دارد را انتخاب نمود. این قابلیت وجود دارد که به طور همزمان جستجو در همه پروژه ها موجود در TFS صورت پذیرد. برای این کار کافی است از منوی پایین رونده Project گزینه Any Project انتخاب شود.

بخش های بعدی، فیلدهایی برای جستجو می باشند که در هر زمان فقط می توان از یکی از آنها استفاده کرد (جستجو ترکیبی امکان پذیر نمی باشد). با استفاده از فیلد Saved query می توان از کوئری های آماده و ذخیره سازی شده قبلی برای جستجو استفاده کرد. به عنوان مثال می توان کوئری ای که آیتم های تست کیس های در دست آماده را لیست می کند مورد استفاده قرار داد. برای انتخاب کوئری از دو امکان مشابه هم می توان استفاده نمود. منوی پایین رونده و دکمه "...". هر دو امکان کاملاً مشابه یکدیگر می باشند و منجر به انتخاب کوئری خواهند شد.

فیلد بعدی جستجو بر اساس ID می باشد. این فیلد دقیقاً مشابه همان فیلدی است که در صفحه Add Link to Product Backlog Item مشاهده شد. فیلد Title contains را می توان پر استفاده ترین شیوه جستجو نامید. این فیلد همان طور که از نامش پیداست عنوان یا قسمتی از عنوان یک آیتم را گرفته و بر اساس آن جستجو را انجام می دهد. این حالت جستجو استثناً دارای یک منوی پایین رونده است که نوع آیتم مورد نظر را تعیین می کند. تعیین نوع آیتم به دو علت صورت می گیرد؛ اول اینکه جستجو در میان تعداد زیادی آیتم زمان بر است و محدود کردن آیتم به نوع خاصی می تواند باعث کمینه شدن زمان جستجو گردد. همچنین ممکن است عنوان برخی آیتم ها مشابه یکدیگر باشد، با تعیین نوع می توان جستجو را برای نوع مورد نظر محدود کرد.

پس از تعیین پارامترهای جستجو باید روی کلید Find کلیک شود تا لیست آیتم هایی که با پارامترها قابل تطبیق هستند ظاهر شود. برای انتخاب آیتم جستجو شده باید کنار آنها علامت زده شود. از دو دکمه Select All و Unselect All می توان برای انتخاب و عدم انتخاب همگانی آیتم ها استفاده نمود. انتخاب چندگانه آیتم ها باعث می شود هر کدام یک به یک با آیتم بک لاگ پیوند ایجاد نمایند.

کلید Reset در پایین صفحه نیز باعث خالی شدن لیست و فیلدهای جستجو خواهد شد. از این کلید برای انجام یک جستجو جدید استفاده خواهند شد. پس از انتخاب آیتم های مورد نظر برای تأیید آنها باید روی دکمه OK کلیک شود.

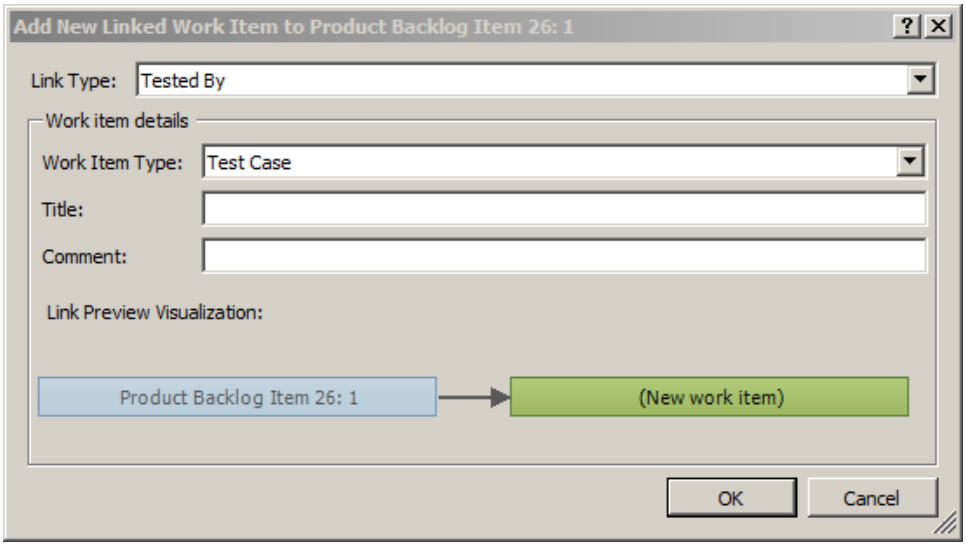
ID آیتم های انتخابی در فیلد Work item IDs از پنجره Add Link to Product Backlog Item نمایش داده می شوند. اگر چند آیتم انتخاب شده باشد ID آیتم ها با کاما از یکدیگر جدا می شوند. در پایین این بخش فیلد Description وجود دارد. این فیلد فقط خواندنی است و شرحی راجع به آیتم های انتخابی ارائه می کند در فیلد Comment نیز می توان توضیحات بیشتری برای این پیوند درج کرد.

پنجره Add Link to Product Backlog Item از قابلیت بسیار جالبی به نام link Preview Visualization بهره می برد. این قابلیت در حقیقت پیش نمایشی بصری از پیوند در حال ارتباط است. روابط پیوندی میان آیتم ها برخی اوقات ممکن است کمی پیچیده شود؛ پیش نمایش بصری باعث تسهیل در فهم بهتر این روابط خواهد شد. این پیش نمایش به صورت کلی از دو مستطیل که هر کدام نشان گر یک آیتم می باشند تشکیل می شود. این دو مستطیل با خطی به یکدیگر متصل می شوند. این خط جهت دار است و بسته به نوع ارتباط جهت های مختلفی خواهد داشت. برای آیتم های چندگانه معمولاً از نمای پشته مستطیل ها استفاده می شود.

با ایجاد پیوندها ، آیتم های پیوندی در تب Test Case لیست خواهند شد. لیست پیوندها بر اساس نوع پیوند آیتم ها را دسته بندی می کند. در تب Test Case از آنجایی که فقط یک نوع پیوند قابل تعریف است صرفاً دسته Test Case به نمایش در خواهد آمد. برای باز کردن هر آیتم پیوندی (تست کیس ها) کافی است در لیست روی آیتم مورد نظر دو بار کلیک شود و یا در جعبه ابزار آیکن Open Linked Item فشرده شود. این کار باعث باز شدن تست کیس در تبی جداگانه خواهد شد.

در جعبه ابزار دکمه ای وجود دارد به نام Edit Link. این کلید برای ویرایش یک پیوند مورد استفاده قرار می گیرد. کلیک روی کلید Edit Link باعث باز شدن مجدد پنجره پیوند قبلی، این بار با عنوانی متفاوت (Edit Link) خواهد شد. برای حذف پیوند میان دو آیتم باید پس انتخاب پیوند مورد نظر در لیست کلید Delete Link از جعبه ابزار فشرده شود. این عمل باعث حذف پیوند میان دو آیتم می شود. لازم به ذکر است که کلید Delete Link قادر به حذف آیتم اصلی نمی باشد و صرفاً می تواند پیوندها را حذف نماید.

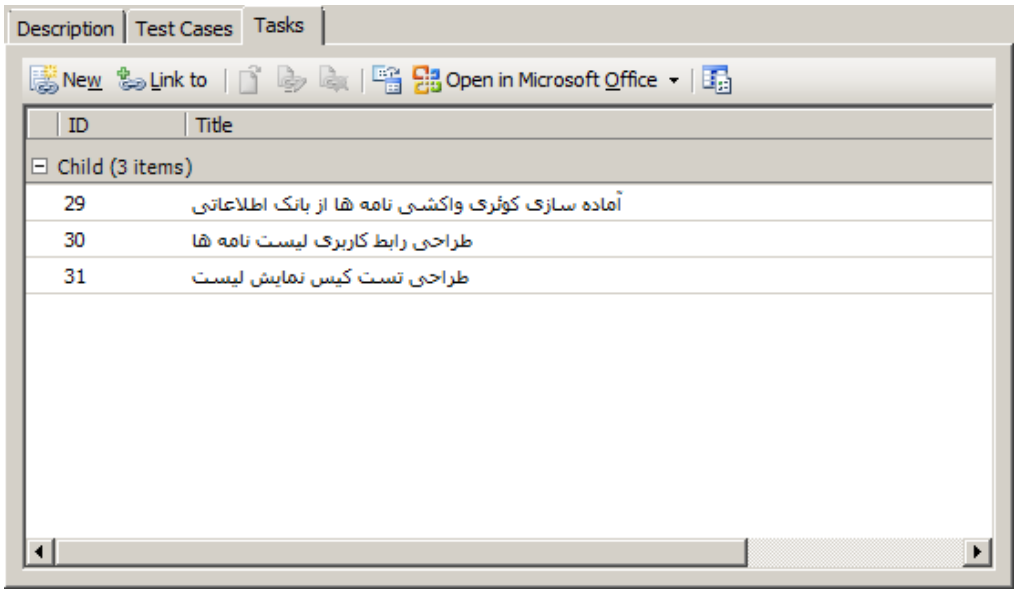
یکی از قابلیت های جالبی که در TFS برای پیوندها ارائه می شود پیوند به یک آیتم جدید است. اگر از پیش آیتمی برای پیوند وجود نداشته باشد، می توان از طریق کلید New در نوار ابزار آیتمی را ایجاد کرد و همزمان به آیتم جاری پیوند داد. با کلیک روی این گزینه روی صفحه پیوند با شکلی جدید ظاهر می شود. تفاوت این صفحه در بخش Work Work Item می باشد(شکل 17-9).



شکل 17-9: تست کیس اضافه شده به عنوان آیتم لینک

اولین فیلد این صفحه Work Work Item نام دارد. این فیلد نوع آیتمی که قرار است ایجاد شود را تعیین می کند. به دلیل این که این صفحه از طریق تب Test Cases باز شده است، صرفاً دارای آیتم Test Case می باشد. در فیلد بعدی به ترتیب عنوان و توضیحات آتیم را تعیین می کنند. با کلیک روی کلید OK آیتم جدیدی ایجاد شده و برای آن تب جداگانه ای گشوده خواهد شد. از طریق این تب می توان سایر فرآیند ایجاد آیتم را تکمیل کرد. با ذخیره سازی آیتم جدید پیوند آن در لیست تب Test آیتم بک لاگ (که این آیتم را ایجاد کرده است) نمایان خواهد شد.

تب آخر در پنل سمت چپ تب Tasks می باشد(شکل 18-9).

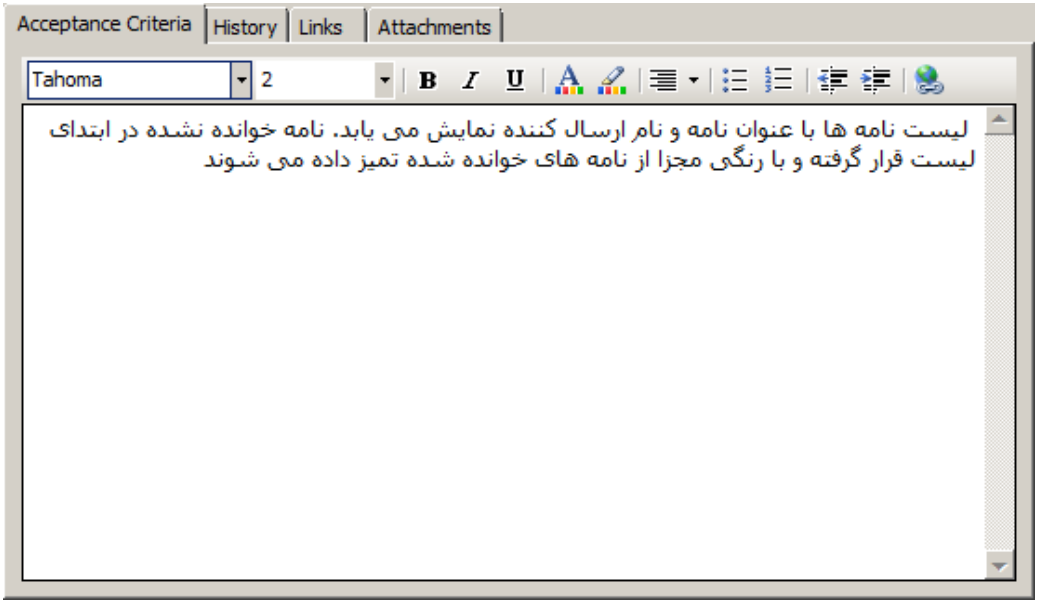


شکل 18-9: تب Tasks در PBI

این تب کاربردی مشابه به Test Case دارد با این تفاوت که برای Taskها(وظایف) به یک آیتم بک لاگ مورد استفاده قرار می گیرد. برای هر کدام از وظایفی که برای یک آیتم بک لاگ تعریف می شود باید آیتم های جداگانه Task ایجاد شود. از طریق تب Tasks می توان این پیوند ها را ایجاد کرد. کل ساختار این تب و چگونگی ایجاد پیوندها مشابه تب Test Case می باشد.

از آنجایی که Taskها عملاً زیر مجموعه PBI ها محسوب می شوند. نوع پیوند Child تعیین می شود. با نگاهی به پیش نمایش بصری درک بهتری از این نوع ارتباط حاصل می شود.

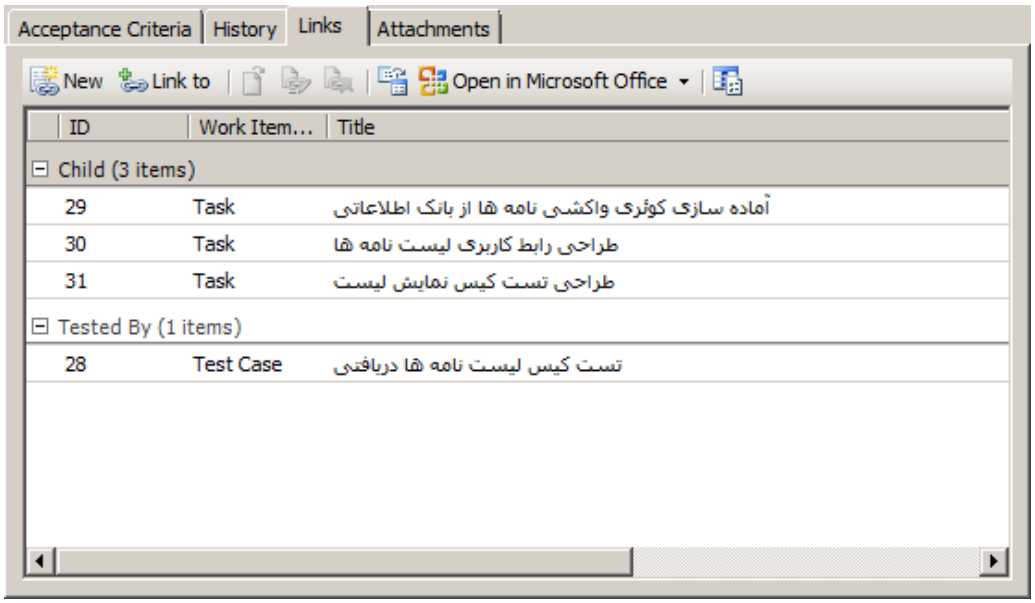
پنل سمت راست از چهار تب جدید تشکیل می شود. اولین و مهمترین تب Acceptance Criteria می باشد. این تب تنها از یک ورودی متن تشکیل شده است(شکل 19-9).



شکل 19-9: تب Acceptance Criteria در PBI

از این بخش برای درج شرایط تطبیق استفاده می شود. الگوی خاصی برای نوشتن متن شرایط تعیین است و دست کاربر در این مورد باز گذاشته شده است. شرایط تطبیق از بخش هایی است که پیش از جلسه برنامه ریزی اسپرینت باید برای آیتم های کاندید کامل شده باشد.

تب History در این بخش عیناً مشابه تب History در آیتم اسپرینت (که قبلاً توضیح داده شده) می باشد. تب Links مرکز مدیریت همه لینک(پیوند)های ارتباطی یک آیتم می باشد. اگر در بخش هایی چون Test Case و Tasks پیوندی درج شده باشد در این قسمت قابل مشاهده است(شکل 20-9).



شکل 20-9: تب Links در PBI

کاربرد دسته بندی پیوندها در بخش Link بیشتر احساس می شود. در صورت ازدیاد تعداد پیوندها، هر کدام از دسته بندی ها را می توان با کلیک روی آن عنوان آنها، کمینه ساخت.

از طریق این بخش می توان انواع مختلفی از پیوندها را برقرار کرد. با باز کردن صفحه پیوند مشاهده می شود که در بخش Link Type پیوندهای زیادی تعبیه شده است. با انتخاب هر کدام از پیوندها طرح بندی صفحه تغییر می کند. در ادامه تعدادی از پر استفاده ترین پیوند از شرح خواهد گذشت:

Child: از این لینک برای پیوند زدن آیتم جاری با یکی با یکی از فرزندانش (به عنوان مثال Task) استفاده می شود.

Hyper Link: از این پیوند برای ثبت یک لینک اینترنتی استفاده می شود. دو فیلد Address و Comment به ترتیب آدرس اینترنتی لینک و توضیحات آن استفاده می شود. پس از درج لینک، کاربر می تواند با دوبار کلیک بر روی لینک، از طریق یک تب جداگانه به صفحه اینترنتی مرتبط با آن دست یابد.

Parent: این پیوند باعث اتصال آیتم والد به آیتم جاری می شود. به عنوان مثال اگر آیتم جاری یک Task باشد با استفاده از پیوند Parent می توان این آیتم را به یک آیتم بک لاگ متصل کرد. توجه شود که هر کدام از پیوندها حکم آیتم پیوندی را خواهند داشت و نسبت به آیتم جاری نام گذاری می شود.

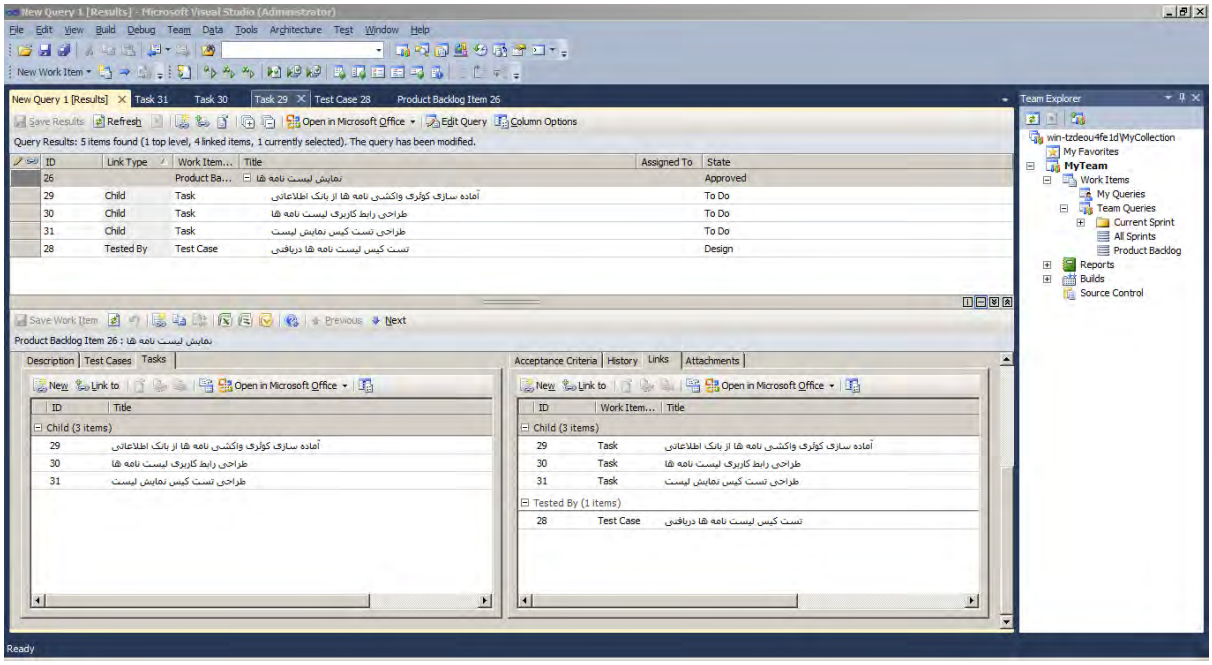
Prodecessor: این پیوند برای اتصال آیتم هایی استفاده می شود که باید از نظر زمانی پیش از آیتم جاری تکمیل شوند.

Related: این پیوند برای اتصال دو آیتم مرتبط مورد استفاده قرار می گیرد. این اتصال بر خلاف معمول جهت ارتباطی خاصی ندارد.

Seuccessor: از این پیوند برای اتصال آیتم هایی استفاده می شود که کار روی آنها پس از تکمیل آیتم جاری آغاز خواهد شد. به عبارت دیگر شروع چنین آیتم هایی منوط به تکمیل شدن آیتم جاری است.

Test Case: این پیوند دقیقاً بر عکس تب Test Case می باشد. در تب Test Case آیتم پیوندی یک تست کیس بود. اما در این پیوند آیتم جاری یک تست کیس است و آیتمی که به آن پیوند می خورد یک آیتم بک لاگ محصول می باشد.

کلید View as Query در جعبه ابزار برای مواقعی تعبیه شده است که بخواهیم لیست پیوندها را به عنوان نتایج کوئری مشاهده کنیم با کلیک روی این دکمه تب جدیدی باز می شود و لیست پیوندها به صورت کوئری ارائه می شود. این کار باعث می شود که در یک صفحه همزمان به گزینه های بیشتری دسترسی داشته باشیم(شکل 21-9).



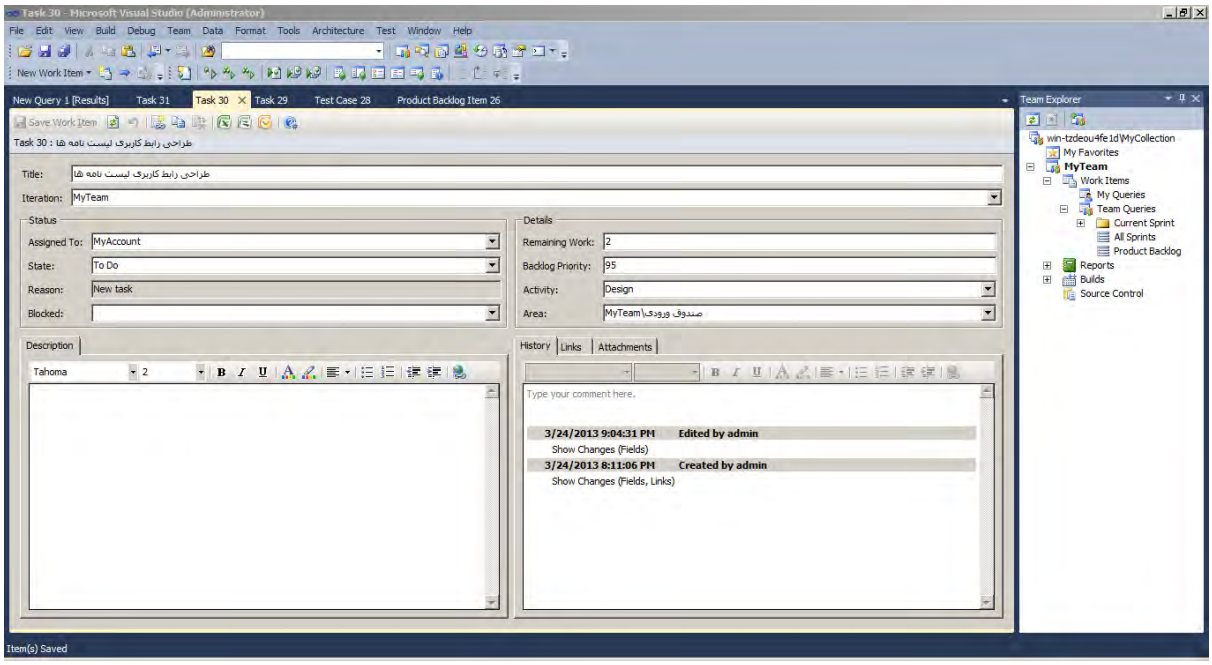
شکل 21-9: نمایش لیست پیوندها به عنوان نتایج یک کوئری

آخرین تب صفحه آیتم بک لاگ Attachment می باشد. همانطور که در بخش آیتم اسپرینت گفته شد از این تب برای ضمیمه کردن فایل های خارجی به آیتم ها استفاده می شود.

سه تب آخر یعنی History، Links و Attachment برای همه آیتم ها مشترک است؛ از این رو در معرفی آیتم های بعدی از تشریح مجدد آنها صرف نظر خواهد شد.

Task

Taskها بخش بندی آیتم های بک لاگ به عملیات فنی توسعه می باشند(شکل 22-9).

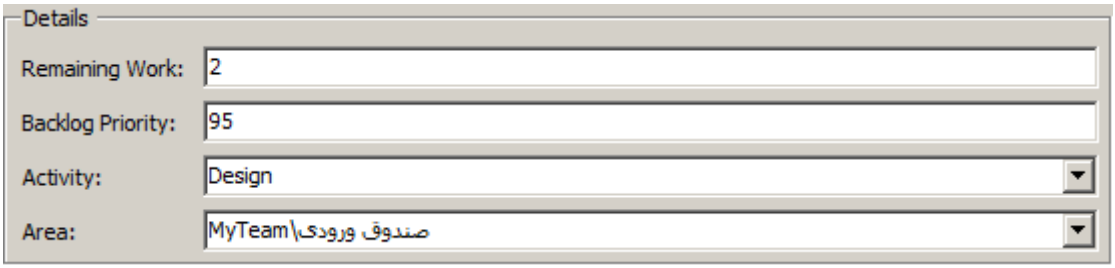


شکل 22-9: یک آیتم Task

با Taskها در TFS به صورت آیتم مستقل رفتار می شود. معمولاً Taskهای یک آیتم به صورت مجزا ایجاد می شوند و سپس به آیتم پیوند می خورند. راه های متفاوتی برای ایجاد Taskها وجود دارد. مرسوم ترین آنها ایجاد کردن یک Task جدید برای یک آیتم بک لاگ است. معمولاً Taskها در تب Task آیتم بک لاگ با کلیک بر روی دکمه New ایجاد می شوند. مزیت این کار این است که علاوه بر ایجاد یک آیتم Task، به صورت خودکار پیوند مورد نظر نیز ایجاد می شود. روش دیگر استفاده از زیر منوی New Work Item از منوی Team است. با کلیک روی گزینه Task از زیر منوی مذکور تب جدید با عنوان New Task به علاوه یک شماره ترتیبی باز خواهد شد. این آیتم نیز مانند آیتم یک لاگ نیاز به یک عنوان دارد و این موضوع از طریق یک پیام اخطار به اطلاع می رسد.

مانند همه آیتم ها، این آیتم نیز دارای فیلد Itreration می باشد. این فیلد تعیین می کند که وظیفه جاری در کدام تکرار(اسپرنیت) تکمیل می شود. سایر فیلدهای آیتم Task در دو بخش Status و Destails دسته بندی می شوند(شکل 23-9 و 24-9):

شکل 23-9: بخش Status در آیتم Task



شکل 23-9: بخش Details در آیتم Task

فیلد Assigned To همان طور که پیش تر گفته شد مربوط به تخصیص وظیفه به یکی از توسعه دهندگان است. Taskها به علت کوچکی شان معمولاً توسط یک نفر انجام خواهند شد. نام توسعه دهنده ای Task را تکمیل می کند باید در فیلد Assigned To وارد شود.

فیلد State در آیتم Task گزینه های متفاوتی دارد. اولین حالت آن To Do می باشد. To Do حالتی است که آیتم آماده انجام باشد. همه آیتم های Task برای اولین بار پس از ایجاد در این حالت هستند و در لیستی با همین نام قرار می گیرند. توسعه دهندگان هنگامی گزینش Taskها به این لیست که همه آیتم های Task در آن می باشند نگاه می کنند و آیتم های با اولویت را انتخاب می کنند.

آیتمی که در حالت To Do قرار گرفته باشد به دو حالت دیگر می تواند تبدیل شود: Removed و In Progress. حالت Removed حالتی است که نیاز به حذف آیتم باشد. مجدداً تأکید می شود که آیتم های ایجاد شده در TFS قابل حذف شدن نمی باشند اما حذف یک آیتم Task تفاوتی با حذف یک آیتم بک لاگ دارد. اگر یک آیتم بک لاگ به حالت Removed برود در بخش State علاوه بر خود Removed گزینه New نیز برای بازیابی مجدد آیتم وجود دارد. اما حالت Removed برای یک آیتم Task از چنین قابلیتی برخوردار نیست و آیتم به صورت دائمی در حالت Removed باقی خواهند ماند. اما همچنان حذف نخواهد شد. پس زمانی باید از این حالت استفاده شود که در خصوص حذف آیتم Task اطمینان کامل حاصل شده باشد.

حالت In Progress زمانی مورد استفاده قرار می گیردکه تیم کار بر روی آیتم را آغاز کرده باشد. هر آیتم در حال تکمیل به حالت In Progress تبدیل خواهد شد. آیتم از این حالت به سه حالت To Do، Remove و Done قابل تبدیل است. برای آیتم های Task این قابلیت تعبیه شده است تا کاربران بتوانند حین تکمیل یک وظیفه(Task)، در صورت نیاز به حذف آن، آیتم را مستقیماً به حالت Remove ببرند. حالت To Do نیز برای زمانی مورد استفاده قرار می گیرد که تیم به صورت موقت دست از کار روی یک Task بکشد. گزینه آخر Done است که موردِ استفاده آن هنگامی است که کار بر روی وظیفه تمام شده باشد. هنگامی که بتوان یک وظیفه را تکمیل شده قلمداد کرد، وظیفه مذکور باید به حالت Done تبدیل شود. در صورتی که تیم تشخیص دهد که یک وظیفه تکمیل شده نیاز به کار بیشتری دارد، نیاز است که آن وظیفه از حالت Done به حالت In Progress تبدیل شود. فیلد فقط خواندنی Resone نیز برای توضیحات تکمیلی راجع به حالت های مختلف می باشد. این فیلد صرفاً جهت اطلاعات بیشتر مورد استفاده قرار می گیرد(شکل Task States).

شکل Task States

فیلد Blocked دارای یک گزینه می باشد و آن هم Yes است. این فیلد تعیین می کند که Task جاری مسدود شده یا خیر. در صورت مسدود بودن یک Task باید مقدار Yes را برای فیلد Blocked آن انتخاب شود. به آیتمی مسدود شده گفته می شود که به علت مشکل، کار بر روی آن امکان پذیر نباشد. Taskهای مسدود کاندیدای خوبی برای Impediment در پروژه هستند. مشکل این آتیم ها باید سریعاً تشخیص داده شده و رفع شود. برای خارج کردن یک آیتم از حالت مسدود کافی است مقدار Yes موجود در فیلد Blocked پاک شود.

فیلد Remaining Work فیلدی است که خاص Taskها می باشد. این فیلد کار باقی مانده را برای Task ذخیره می کند. مقدار این فیلد معمولاً بر حسب ساعت می باشد. در پایان هر روز کاری این برای هر یک از وظایفی که روی آنها کار می شود باید به روز رسانی شود. با صفر شدن مقدار کار باقی مانده یک وظیفه، بدیهتاً آن وظیفه به حالت تکمیل شده (Done) تبدیل می شود.

فیلد Backlog Priotity نیز برای اولویت بندی Taskها مورد استفاده قرار می گیرد. و بسته به این که اولویت مهمتر عددی بالا انتخاب یا عددی پایین، لیست وظایف نهایی می تواند بر اساس این فیلد به صورت نزولی و یا صعودی مرتب شود.

فیلد Activity یک فیلد اختیاری است. این فیلد نوع کاری را روی Task صورت می گیرد، مشخص می کند. لیست کارهای فیلد Activity عبارت اند از :Development، Design، Documention، Requirement و Test. این فیلد باعث شفاف سازی نوع کار انجام شده در وظیفه خواهد شد و بیشتر جنبه اطلاع رسانی دارد.

برای این آیتم نیز می توان سطحی را از طریق Area تعریف کرد. سطح واژه تعریف شده برای یک وظیفه معمولاً از آیتم بک لاگ آن مرتبط با آن وظیفه تبعیت می کند.

لازم به ذکر است با رفتن یک Task به حالت Done فیلدهای Blocked و Remaining Work غیر فعال خواهند شد.

آیتم Task همچنین شامل چهار تب Description، History، Links و Attachment می باشد. این بخش ها سابق بر این برای آیتم های دیگر تشریح شده اند و از توضیح مجدد آنها صرف نظر می شود.

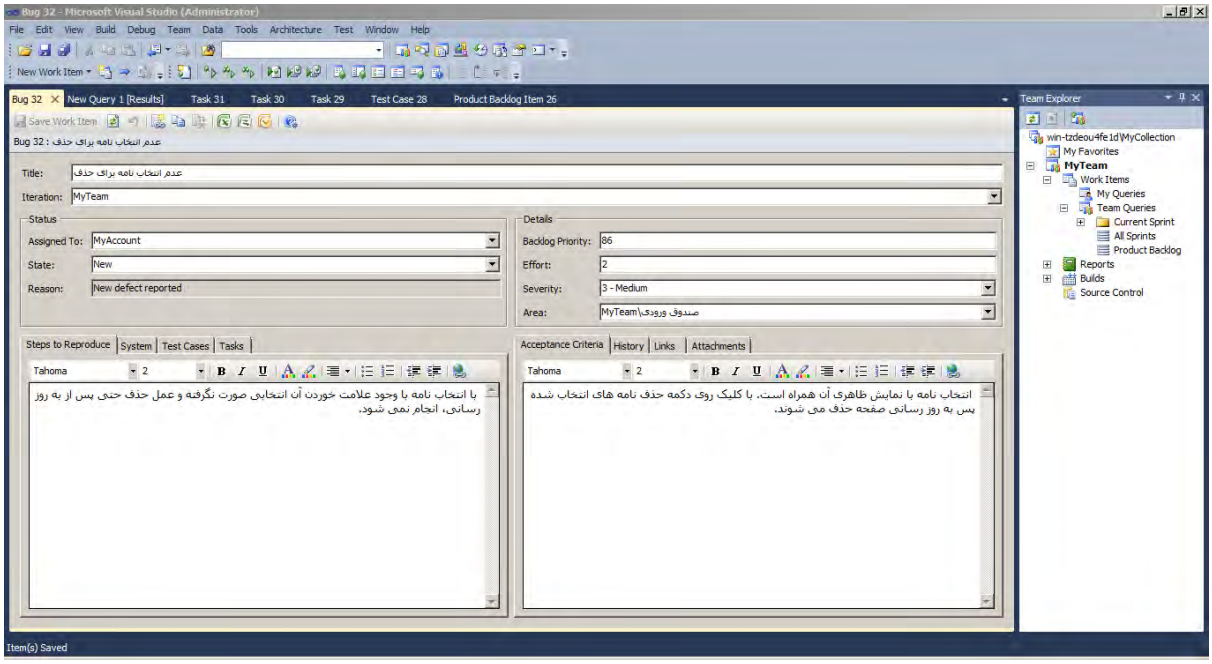
Bug

باگ به معنی اشکال در صحت کارکرد بخشی از نرم افزار می باشد و هنگامی تشخیص داده می شود که عملی، نتیجه ای قابل انتظار را ایجاد نکند. باگ ها عموماً قابل حل هستند. برخی از آنها پس از تشخیص به

سرعت بر طرف می شوند و برخی به کار بیشتری نیاز دارند. آیتم باگ در TFS برای تعریف و مدیریت باگ هایی استفاده می شود که نیازمند برنامه ریزی و کار زیاد هستند.

از دیدگاه اسکرام باگ ها و آیتم های بک لاگ آیتم هایی عیناً مشابه هستند. در اسکرام با هر دو آیتم مذکور به طور یکسان برخورد می شود. آیتم بک لاگ بخشی از نرم افزار است که باید ساخته شود و شامل وظایفی می باشد و برای آن تست کیس هایی تعریف می شود آیتم باگ نیز همانند آیتم بک لاگ اولویت بندی می شود و در لیست بک لاگ قرار می گیرد. با تمام شباهت هایی بین این دو آیتم، تفاوت هایی نیز وجود دارد که از طبیعت هر یک بر گرفته شده است. در خصوص این تفاوت ها در ادامه بحث خواهد شد.

برای اضافه کردن یک آیتم باگ مشابه قبل از مسیر Team->New Work Work Item->Bug استفاده می شود. با این عمل تبی با عنوان New Bug به علاوه یک شماره ترتیبی ظاهر می شود(شکل 24-9).



شکل 24-9: یک آیتم باگ

در این تب نیز اولین و ضروری تری فیلد، فیلد عنوان است. از آنجایی که این آیتم و آیتم بک لاگ محصول همزمان در یک لیست قرار می گیرند پس بهتر است عنوان به گونه ای انتخاب شود که این تفاوت را آشکار کند. مشابه همه آیتم های TFS فیلد Iterations در آیتم نیز وجود دارد.

نکته مهمی که هنگام تعریف یک باگ وجود دارد این است که قرار است در کدام اسپرینت روی باگ کار شود. برخی باگ ها دارای اولویت رسیدگی بالایی هستند و بنابراین تیم تشخیص می دهد که در اسپرینت جاری به باگ رسیدگی کند. برخی باگ ها نیز اولویت پایین تری دارند و می توانند در اسپرینت های دیگری حل شوند. باگ هایی نیز وجود دارند که نیاز به برنامه ریزی دقیق تری دارند، این دسته باگ ها را باید به بک لاگ محصول منتقل کرد(فیلد Iterations آن را باید برابر یک Release قرار دارد).

فیلد State دارای مقادیر مشابهی با PBI می باشد با این تفاوت که در قسمت Reason دلایلی مرتبط با باگ ها مطرح می شود. آیتم باگ مانند آیتم بک لاگ دارای فیلد Backlog Priority برای اولویت بندی و Effort جهت ثبت برآورد آیتم می باشد. فیلد جدیدی که در آیتم باگ وجود دارد فیلد Severity می باشد. این فیلد میزان حساسیت خدمت پذیری آیتم را تعیین می کند. فیلد Severity از چهار مقدار Critical، High، Medium و Low با شماره های نزولی 1 تا 4 می باشد. حساسیت این مقدار از بالا به پایین کاهش می یابد. مورد استفاده این فیلد جهت تعیین اولویت سرویس دهی به آیتم باگ می باشد.

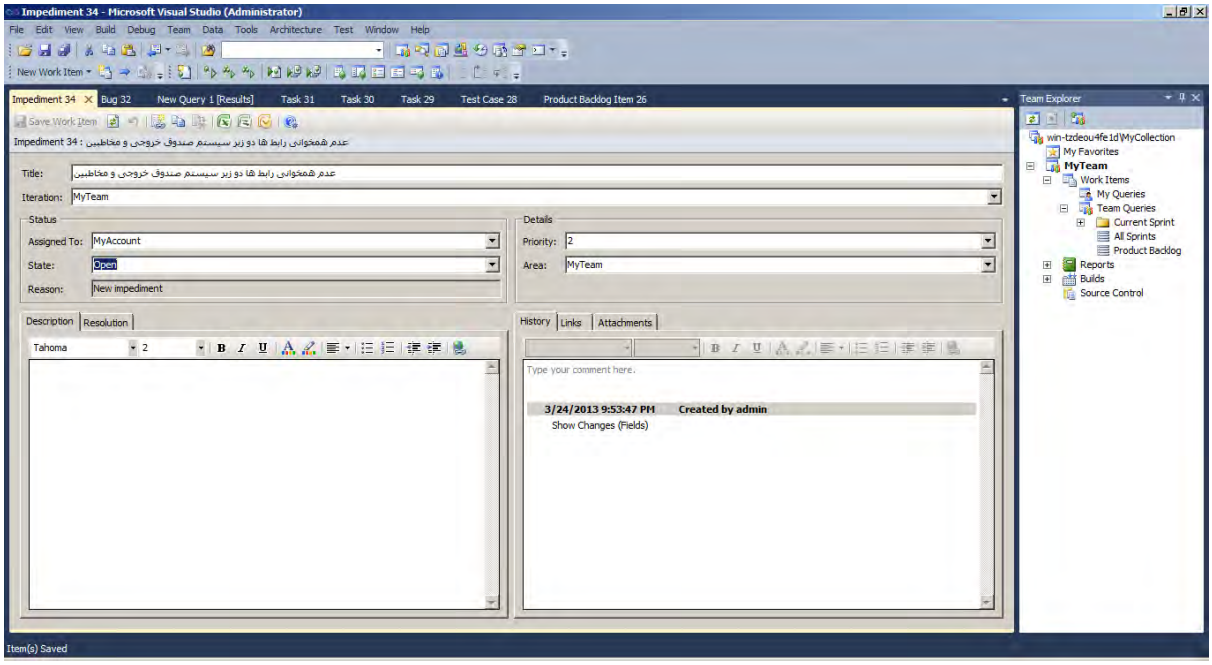
تبی به نام Steps to Reproduce در آیتم باگ وجود دار که خاص این آیتم است. این تب دارای یک فیلد ورود متن است . توسعه دهنده ای که باگ را تشخیص داده و برای آن آیتمی ایجاد کرده، مسئول تکمیل این فیلد است. توسعه دهنده باید مراحلی را که پس از انجام دادن آنها باگ ظاهر می شود، مرحله به مرحله ثبت کند. دیگر

توسعه دهندگان با مشاهده این بخش می توانند از چگونگی تشخیص باگ مذکور اطلاع پیدا کنند(در فصل های پیشین از این بخش با نام چگونگی کشف یاد شده است).

همانطور که پیش تر گفته شد باگ ها شبیه با آیتم های بک لاک هستند؛ از این رو کار روی آباگ ها مستلزم شکستن آنها به وظایف است. تب Task برای مدیریت وظایف یک باگ در این آیتم تعبیه شده است. تب Test Case نیز برای این آیتم وجود دارد. از آنجایی که دیگر نیازی به ایجاد تست کیس جداگانه برای آن نمی باشد، می توان از همان تست کیس های قبلی که برای PBI طراحی شده بود استفاده کرد. اما در برخی واقع نیاز است تست کیس جداگانه ای برای یک بک لاگ طراحی کرد. در هر دو حالت (پیوند تست کیس قبلی و یا ایجاد تست کیس جدید) می توان از تب Test Case واقع در آیتم باگ بهره برد.

Impediment

این آیتم موانع و مشکلاتی را ثبت می کند که به نحوی موجب توقف روند جاری توسعه خواهند شد. ثبت این موانع به صورت یک آیتم مجزا آنها را مدیریت پذیر می کند(شکل 25-9)



شکل 25-9: یک آیتم Impediment

اغلب فیلدهای آیتم Impediment در آیتم های پیشین شرح داده شده است. در این آیتم دو فیلد و دو تب جدید وجود دارد که در ادامه توضیح داده خواهند شد.

یکی از این دو فیلد، فیلد State است. در حقیقت این فیلد، جدید نیست بلکه دارای مقادیری جدید است. فیلد State دارای دو مقدار Open و Close می باشد. در ابتدای تعریف یک Impediment مقدار Open به صورت پیش فرض برای این فیلد درج می شود. حالت Open به این معنی است که موانعی به وجود آمده و هنوز راه حلی برای آن یافت نشده است. پس از حل مانع حالت Impediment از Open به Closed تغییر پیدا می کند. با مقدار گرفتن فیلد State با Close آیتم عملاً از این پس غیر قابل با استفاده می شود چرا که آخرین حالت فیلد State است. به عبارت دیگر مانعی که یکبار حل شده است را نمی توان مجدداً به لیست موانع برگرداند.

برای هر کدام از موانع اولویتی تعیین می شود. برخی موانع حساس و اضطراری هستند و نیاز به توجه سریع تری دارند. برخی دیگر از اولویت پایین تری برخوردارند. فیلد Proority این اولویت را برای آیتم به ثبت می رساند. مقادیر این فیلد عددی هستند و در محدوده 1 تا 4 قرار دارند به نحوی که 1 دارای بالاترین اولویت و 4 دارای پایین ترین اولویت می باشد.

در بخش پایین سمت چپ دو تب وجود دارند. تب Description دارای یک ورودی متنی است و برای درج توضیحات راجع مانع مورد استفاده قرار می گیرد. تب Resolution نیز مانند تب قبلی دارای یک ورودی متن است

و برای ثبت راه حلِ بر طرف سازی مانع تعبیه شده است موانع دارای لیست جداگانه ای در اسپرینت هستند و در لیست بک لاگ اسپرنیت قرار نمی گیرند.

Test Case

تست کیس ها یکی دیگر از آیتم های توسعه می باشند که به کرات مورد استفاده قرار می گیرند. به عقیده بسیاری از توسعه دهندگان به ازای هر آیتم بک لاگ باید یک تست کیس وجود داشته باشد. تست کیس ها دارای Step(گام)هایی هستند که مرحله به مرحله یک PBI را مورد آزمایش قرار می دهند. با این که در Team Explorer می توان یک Test Case را ایجاد نمود اما اساساً کار روی تست کیس ها بر عهده نرم افزار Test Manager می باشد. نرم افزار Test Management، نرم افزاری است که همراه با Visual Studio نصب می شود(البته فقط با نسخه Ultimate آن) و برای برنامه ریزی و اجرای تست ها مورد استفاده قرار می گیرد. برای اجرای این نرم افزار نیاز به TFS می باشد و بدون نصب TFS قابل اجرا نیست. Test Manager دارای بخش های زیادی می باشد که شرح همه آنها از حوصله این کتاب خارج است.

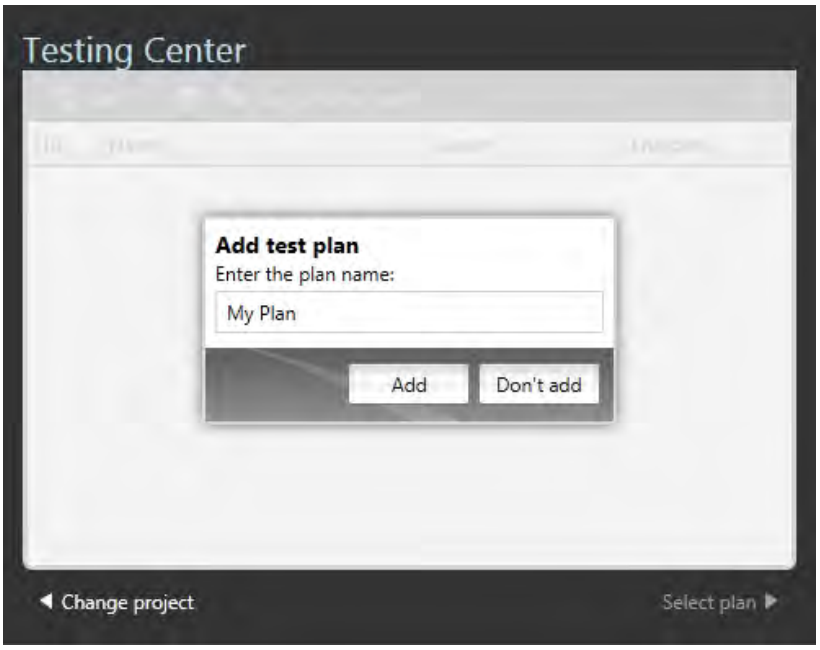
به دو شیوه می توان یک تست کیس را ایجاد کرد: در Team Explorer و در Test Manager. از آنجایی که هر دو نرم افزار به TFS متصل هستند تغییر در آیتم های یک نرم افزار، در دیگر نرم افزار نیز اعمال می شود. توصیه می شود که تست کیس ها به وسیله نرم افزار Test Manager ایجاد شوند چرا که مدیریت و اجرای آنها بهتر صورت می پذیرد.

ابتدا شیوه ایجاد تست کیس در Test Manager مورد بحث قرار می گیرد. با اجرای نرم افزار Test Manager در صورتی که پیش از این اجرا نشده باشد صفحه ای مبنی بر انتخاب یک سرور TFS ظاهر خواهد شد(شکل 26-9).

شکل 26-9: صفحه اتصال به Team Project در Test Manager

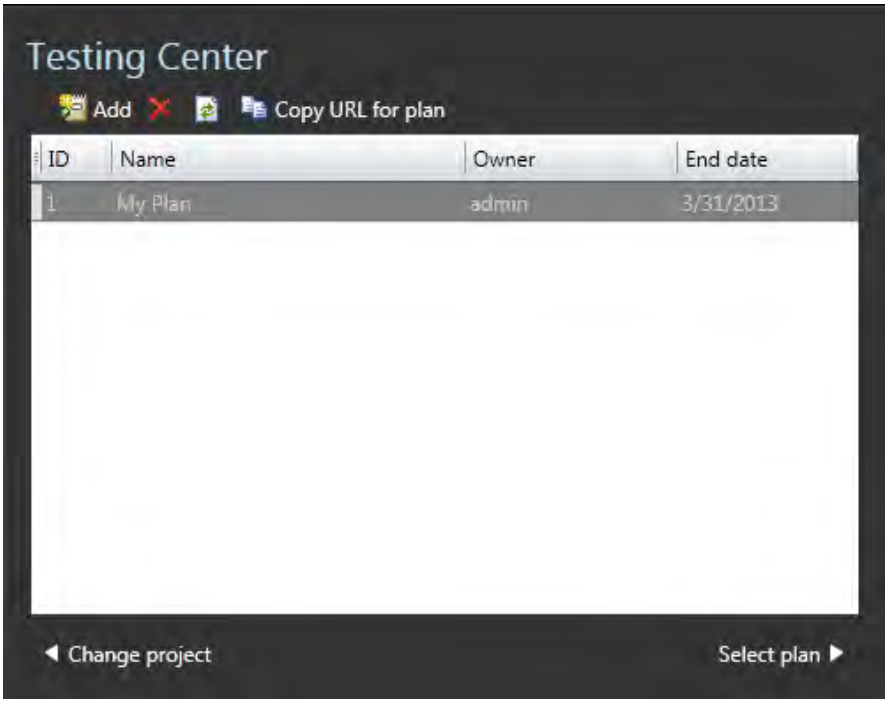
با اتصال به سرور دلخواه TFS، این بار صفحه ای جهت انتخاب تیم گشوده خواهد شد. لیست ظاهر شده شامل همه Collectionها به علاوه تیم های آنها است. با انتخاب تیم و فشردن کلید Connect به صفحه ای برای انتخاب Plan منتقل می شویم. تست کیس ها عموماً در دسته های بزرگتری قرار می گیرند که Plan نام گذاری

می شوند. برای ایجاد تست کیس ها حداقل باید یک Plan وجود داشته باشد. در صورتی که لیست Plan خالی باشد به راحتی می توان با کلید Add یک Plan جدید ایجاد کرد(شکل 27-9).



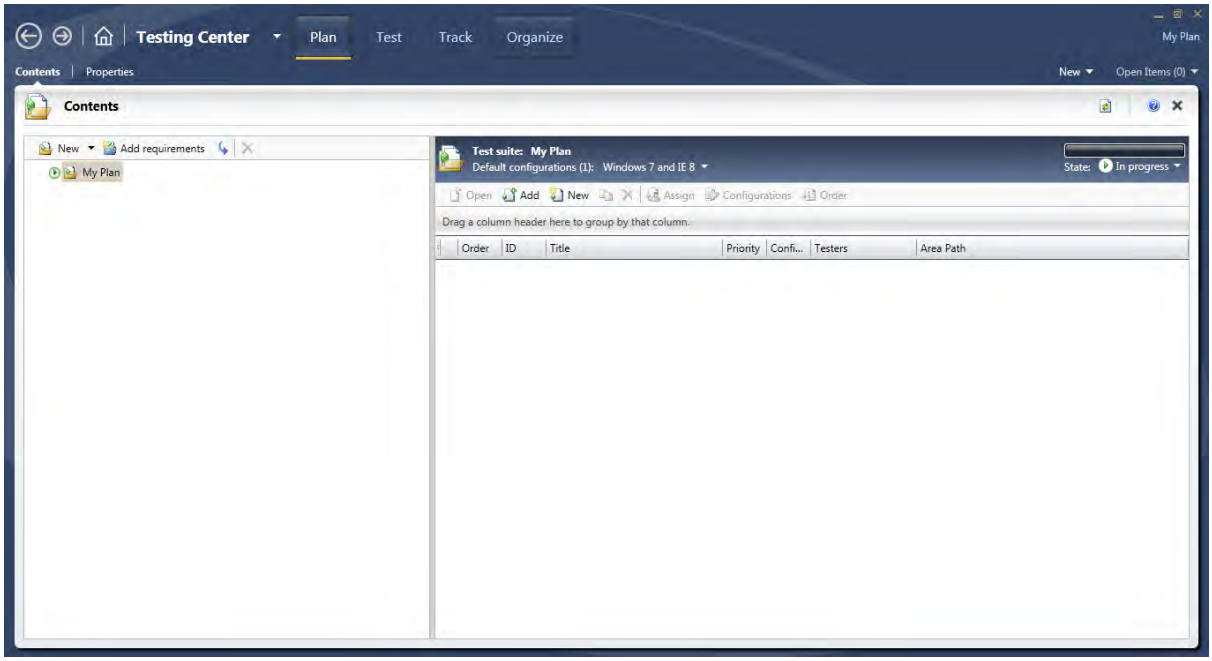
شکل 27-9: افزودن یک Test Plan به پروژه در Test Manager

با کلیک روی Plan مورد نظر و فشردن کلید Select وارد محیط Testing Center می شویم(شکل 28-9).



شکل 28-9: انتخاب Test Plan در Test Manager

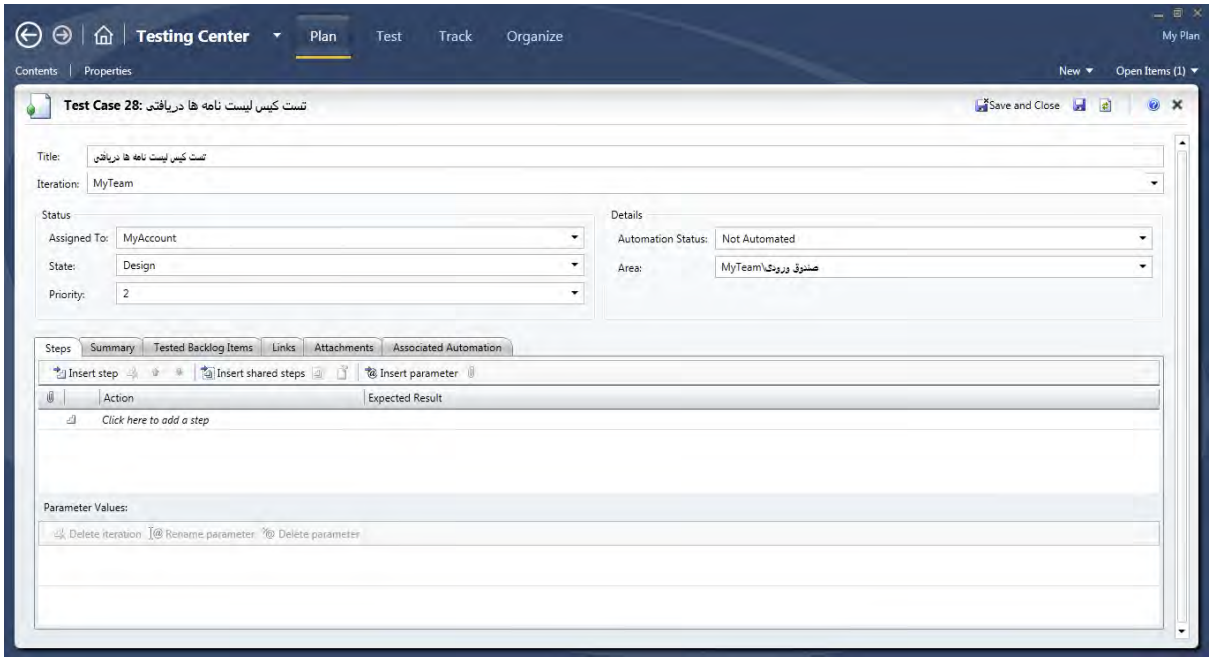
این محیط دارای چهار تب Plan، Test، Track و Organize می باشد(شکل 29-9).



شکل 9-29: محیط Test Manager

تب Plan که به طور پیش فرض پس از اجرای نرم افزار باز می شود مربوط به تعریف و مدیریت تست کیس ها است. این تب شامل دو پنل است: پنل سمت چپ دسته بندی Planها را نشان می دهد و پنل سمت راست لیست و ابزاری را برای مدیریت تست کیس ها مهیا می کند.

برای اضافه کردن یک تست کیس به سادگی می توان روی کلید New در نوار ابزار پنل سمت راست کلیک کرد. با کلیک روی آن پنجره ای جدید گشوده خواهد شد که فیلدهای مانند آنچه در Team Explorer مشاهده کردیم را ارائه می کند(شکل 9-30).



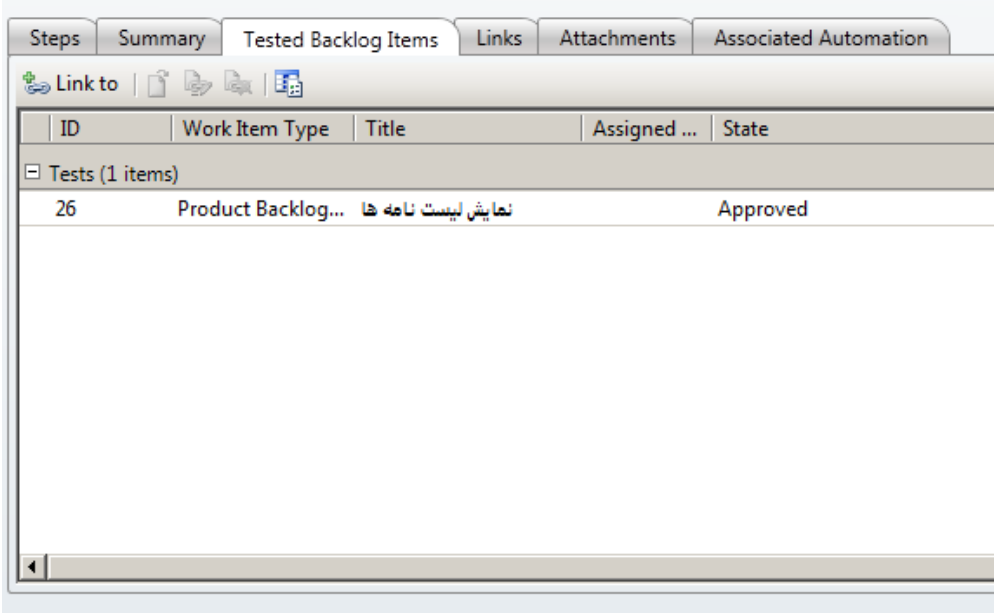
شکل 9-30: افزودن تست کیس جدید در Test Manager

سه فیلد قابل توجه در این پنجره وجود دارد. فیلد State برای این آیتم نیز دارای مقادیر متفاوتی است. اولین حالت قابل مشاهده در این فیلد Design می باشد. این مقدار برای حالتی است که تست کیس در دست طراحی است. چنین حالتی به دو حالت دیگر قابل انتقال است. حالت Close برای بستن تست کیس و هنگامی

به صورت موقت یا دائم به تست کیس نیاز نداریم مورد استفاده قرار می گیرد. حالت دیگر، حالت Ready است و مواقعی استفاده می شود که تست کیس آماده اجرا باشد. کل حالت های تست کیس به همین سه حالت محدود می شوند و می توان از هر کدام به حالت دیگری منتقل شد فیلد Priority نیز برای تعیین اولویت تست کیس ها مورد استفاده قرار می گیرد. مقادیر این فیلد عددی هستند و در محدوده 1 تل 4 قرار دارند. عدد 1 بالاترین اولویت برای تست کیس محصوب می شود و عدد 4 پایین ترین اولویت می باشد.

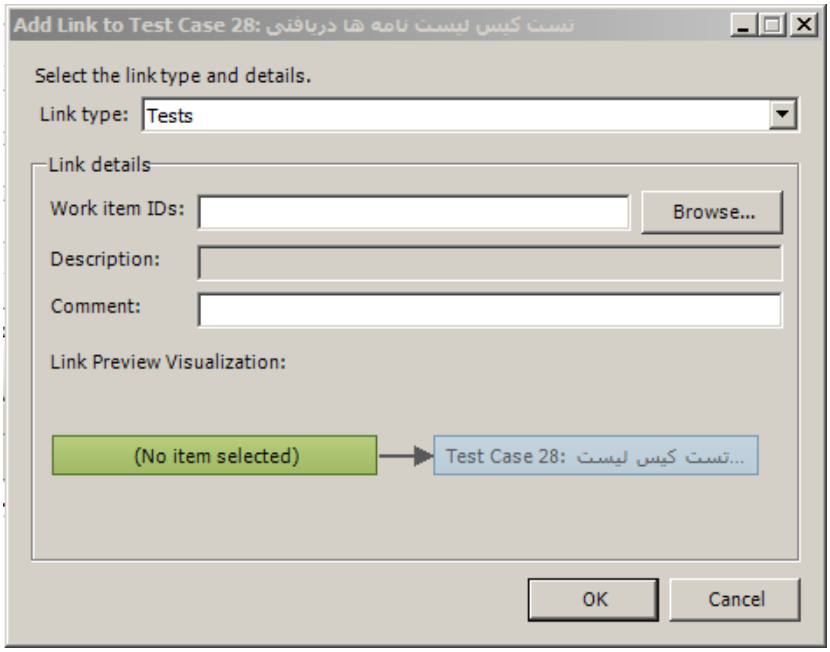
در بخش Details فیلد جدیدی به نام Automation Status وجود دارد که شامل مقادیری چون Not Automated و Palned می باشد. برخی از تست کیس ها به صورت خودکار و در یک Palned ایجاد خواهند شد. این تست کیس ها حالت Palned را به خود می گیرند. برخی از تست کیس ها به صورت مستقل و غیر خودکار انجام خواهند شد. در این تست کیس ها مقدار Not Automated برای فیلد Automation Status انتخاب می شود.

دو تب مهم در این پنجره وجود دارد به نام Steps و ...Test. تب Tested Backlog Items یک پنجره مدیریت پیوند است که به صورت اختصاصی برای آیتم های بک لاگ که توسط این تست کیس آزموده می شوند، تعبیه شده است(شکل 31-9).



شکل 31-9: تب Tested Backlog Items مرتبط با آیتم Test Case در Test manager

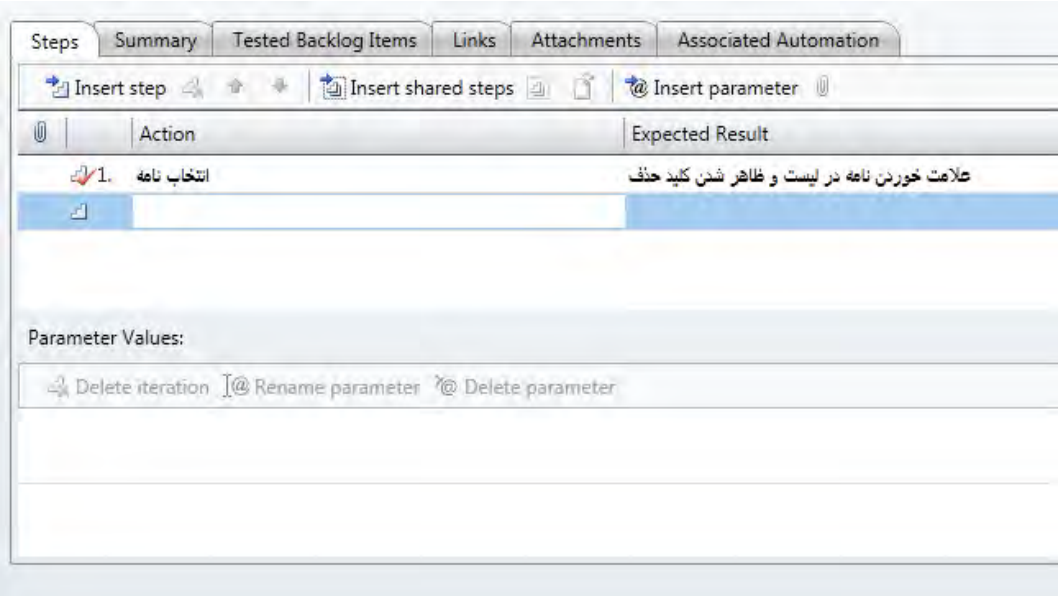
پس از طراحی تست کیس کافی است توسط بخش Add Link to Test Case که از طریق فشردن دکمه Link To ظاهر می شود اقدام به ایجاد پیوند بین تست کیس مذکور و یک PBI کرد(شکل 32-9).



شکل 32-9: افزدون یک PBI به آیتم Test Case

تب Steps مهمترین بخش آیتم Test Case می باشد. در این تب گام هایی که باید برای آزمایش یک PBI پیموده شوند به تفکیک درج می شوند.

نحوه اجرای گام به این صورت است که با اجرای هر گام عملی از سوی آزمون گر (Tester) انجام می شود. هر عمل نتیجه ای به همراه دارد و این نتیجه باید با نتیجه ای که برای هر گام پیش بینیشده است مطابقت نماید. برای تعریف گام ها و نتایج شان باید به تب Steps رجوع کنیم. محتوای این تب یک لیست دو ستونی همراه یک جعبه ابزار است. برای اضافه کردن گام (Step)ها کافی است بر روی قسمت Click here to add a step کلیک شود. با این عمل سطر جاری کاندید درج آیتم قرار خواهد گرفت(شکل 33-9).

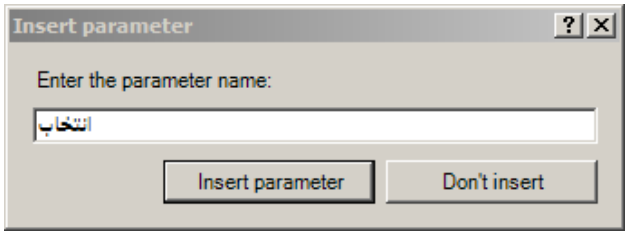


شکل 33-9: تعریف گام برای یک تست کیس در Test Manager

ستون اول این لیست Action نام دارد و عملی که باید برای آزمون انجام شود را تعیین می کند. ستون دوم Expected Result می باشد و مکانی است برای ثبت نتایج مورد انتظار هر عمل. با زدن کلید Enter از صفحه کلید این سطر(گام) ثبت درج خواهد شد. راه دیگر درج یک گام استفاده دکمه Insert step می باشد. با کلیک روی این دکمه، سطری همراه یک شماره ترتیبی درج می شود. استفاده از این کلید بیشتر برای درج یک گام در

بین گام های دیگر است. برای درج یک گام در بین گام ها دیگر باید روی یک گام قرار گرفت و کلید Insert step را کلیک کرد. برای حذف یک سطر(گام) کافی است با انتخاب سطر کلید Remove از نوار ابزار فشرده شود. برای جابجایی ترتیب گام ها نیز می توان از کلیدهای (بالا و پایین) در نوار استفاده کرد.

برخی اعمال در گام ها به نحوی هستند که با استفاده از چند مقدار باید آزموده شوند. در حالت معمول برای هر مقدار باید یک گام ایجاد شوند. اما در Team Managment یک قابلیت جالب وجود دارد جالب وجود دارد به نام پارامتر. پارامترها در حقیقت لیستی از مقادیر هستند. اگر به یک گام یک پارامتر نسبت داده شود گام مذکور برای کل مقادیر موجود در پارامتر باید تکرار شود. برای پارامتردار کردن یک گام، کافی است در انتهای نام آن نام پارامتر را با حائل @ درج کرد. برای درج یک پارامتر باید از نوار ابزار بر روی دکمه Insert parameter کلیک شود. این عمل باعث باز شدن پنجره کوچکی خواهد شد که نام پارامتر را به عنوان ورودی می پذیرد(شکل 34-9).



شکل 34-9: افزودن یک پارامتر

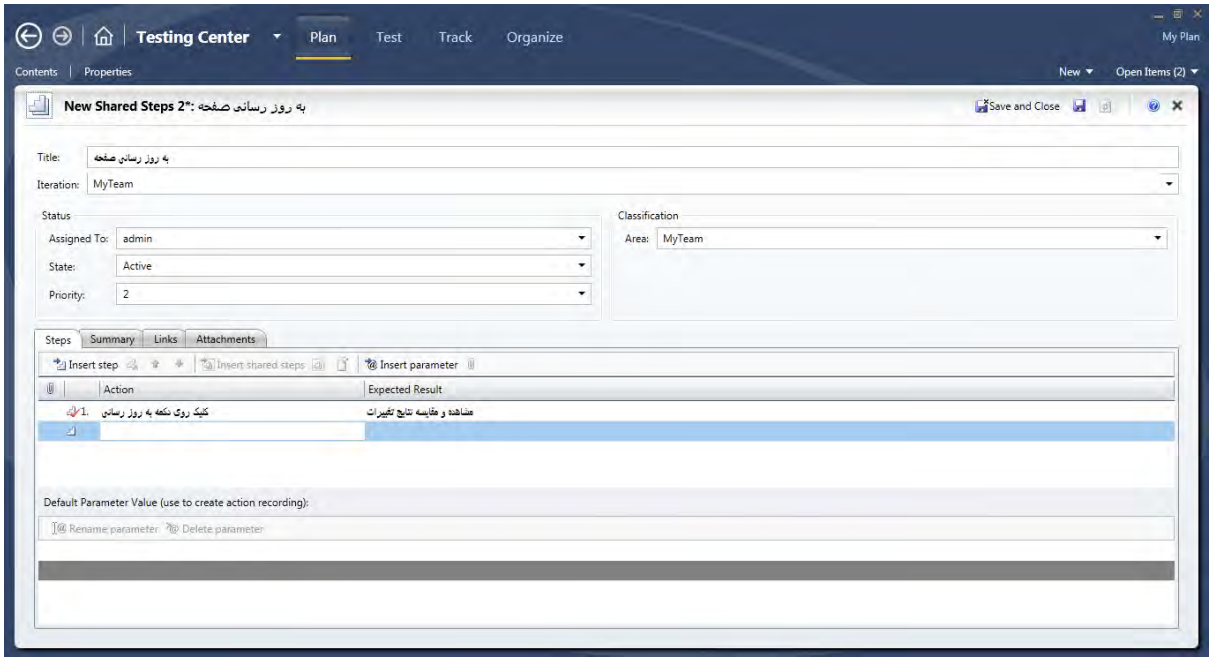
با اضافه کردن یک پارامتر، پارامتر افزوده شده در لیستی که پایین لیست گام ها وجود دارد نمایش داده می شود(شکل 35-9).

شکل 35-9: پارامتر افزوردن شده به یک گام

توجه شود که لیست پارامترها به صورت افقی رشد می کند. در زیر هر پارامتر می توان مقادیر آن پارامتر را لیست کرد. بخش پارامترها دارای یک جعبه ابزار سه کلید می باشد. کلید اول این جعبه ابزار Delete iteration می باشد و برای حذف یک مقدار استفاده می شود(توجه شود که به تعداد تکرار هر کدام از پارامترها بر اساس تعداد مقدار پارامتری است که بیشترین مقدار را دارد). دو کلید بعدی Rename parameter و Delete parameter هستند و به ترتیب برای تغییر نام پارامتر و حذف پارامتر مورد استفاده قرار می گیرند.

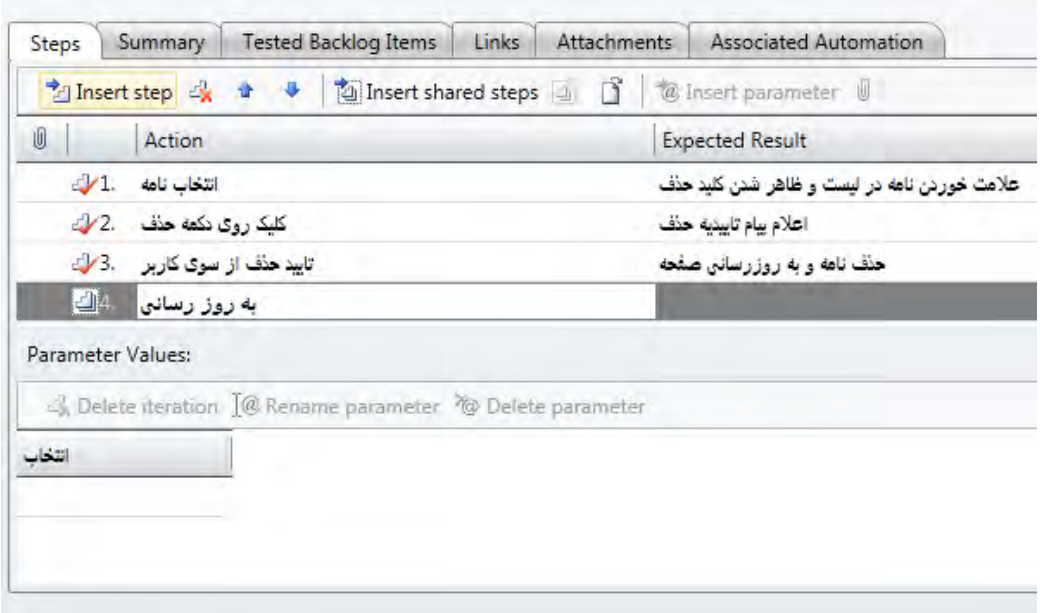
قابلیت جالب دیگری که در بخش Steps گنجانده شده است Shared Steps می باشد. تعریف و ثبت گام ها برای تست کیس کاری تکراری و پرمشقت است. از آنجایی که احتمال زیادی وجود دارد که برخی از گام های تست کیس های مختلف با هم یکسان باشد بهتر است گام های مشترک یک بار به صورت کلی تعریف شوند و در مکان های دیگر صرفاً از آنها استفاده شود.

برای ثبت این گام های مشترک آیتمی معرفی شده است به نام Shared Steps. این آیتم یک آیتم مستقل است و آن را می توان روش متعارف مانند دیگر آیتم ها به TFS اضافه کرد. برای افزودن آن در Team Explorer می توان از مسیر Team->New Work Work Item گزینه Shared Steps را انتخاب کرد و در Test management می توان از زیر منوی New در بالای صفحه گزینه Shared Steps را انتخاب نمود(همانطور که مشاهده می شود آیتم های دیگری نیز در این لیست وجود دارد و از طریق نرم افزار Test Managment می توان اقدام به درج سایر آیتم های TFS نمود). همچنین از طریق کلید Create shared steps واقع در نوار ابزار بخش Steps از صفحه Test Case می توان یک آیتم Shared Steps را اضافه نمود. از هر روشی که استفاده می شود نهایتاً صفحه آیتم جدید Shared Steps گشوده خواهد شد(شکل 36-9).



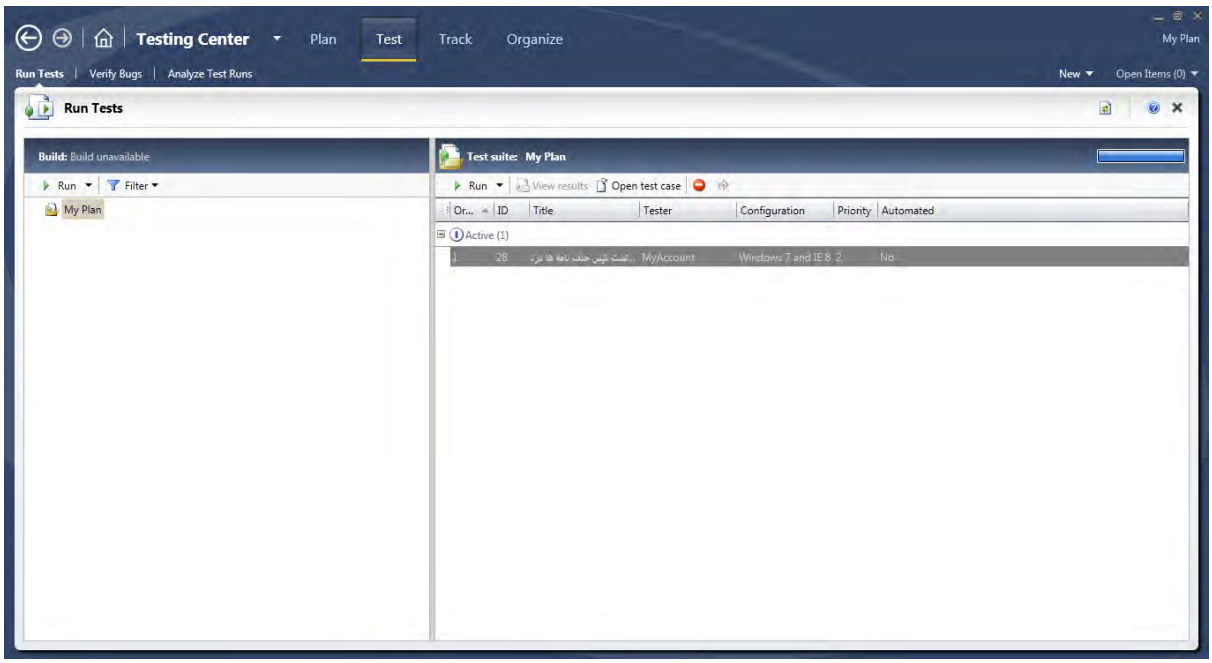
شکل 36-9: آیتم Shared Steps جدید در Test Manager

این صفحه دارای فیلدهایی است که سابق بر این در آیتم های دیگر شرح داده شده است. فیلد State این آیتم دارای دو حالت Active و Close است که به ترتیب حالت فعال بودن و بسته بودن آیتم را تعیین می کنند. تب Steps نیز عیناً مشابه تب Steps بخش Test Case می باشد. همانطور که مشاهده می شود این تب ابزاری برای Shared Steps نیز می باشد؛ به این معنی که هر آیتم Shared Steps می تواند خود دارای یک آیتم Shared Steps دیگر باشد. هر گامی که در این آیتم قرار می گیرد در قسمت Steps تست کیس هایی که حاوی این آیتم Shared Steps می باشند قرار می گیرد. به این ترتیب با یک بار افزودن یک گام می توان به کرات از آن استفاده نمود(شکل 37-9).



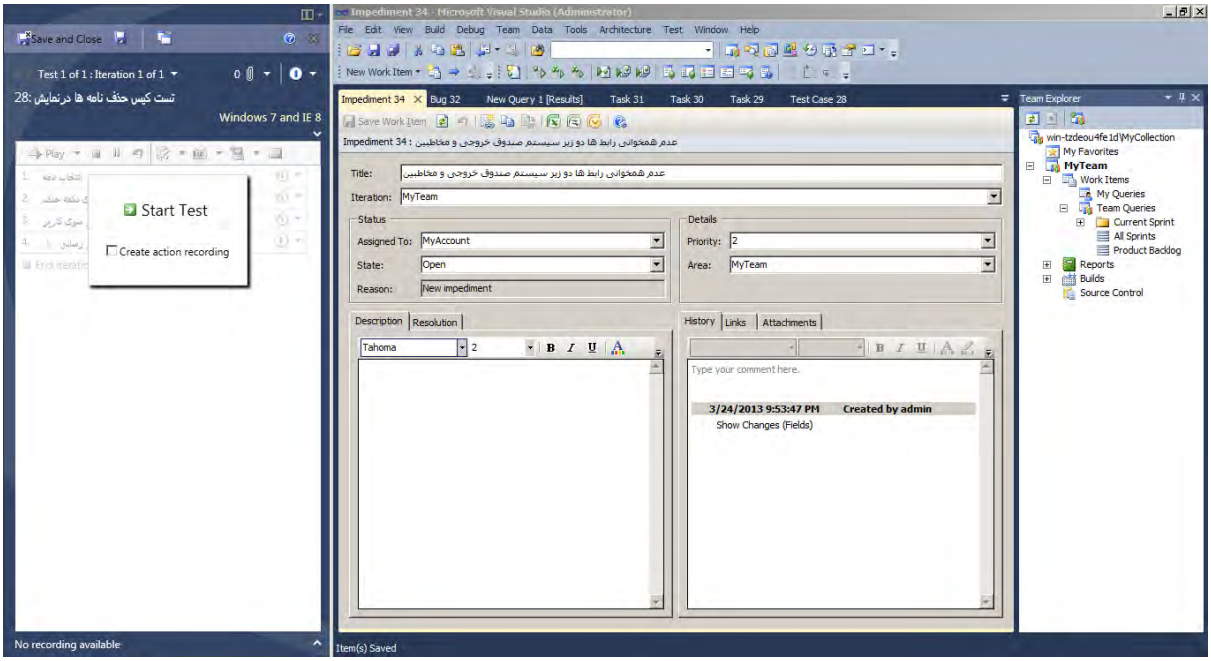
شکل 37-9: افزودن یک آیتم Shared Steps به گام ها در Test Manager

برای اجرا کردن تست کیس ها (در راستای آزمایش بخش های نرم افزار) ابتدا باید ار بالای صفحه تب Test را انتخاب نمود و سپس از پنجره باز شده از قسمت جعبه ابزار بر روی Run کلیک نمود (بدیهی است که پیش از آن باید یکی از تست کیس ها را انتخاب شده باشد)(شکل 38-9).



شکل 38-9: بخش Test در Test Manager

این کار باعث تغییر شکل صفحه نرم افزار Test Managment خواهد شد(شکل 39-9).



شکل 39-9: اجرای یک تست کیس در Test Manager

در این حالت صفحه نمایش به دو قسمت تقسیم خواهد شد. در بخش سمت چپ تست کیس ها به همراه گام هایشان برای اجرا شدن ظاهر خواهد شد. در بخش سمت راست دسکتاپ و برنامه های اجرا شده نمایش داده می شود(در شکل به طور مثال از نرم افزار VS استفاده شده است). با کلیک روی دکمه Start Test آزمون نرم افزار شروع می شود. فرض بر این است در بخش سمت راست از قبل، نرم افزار که قرار است مورد آزمایش قرار گیرد اجرا شده است.

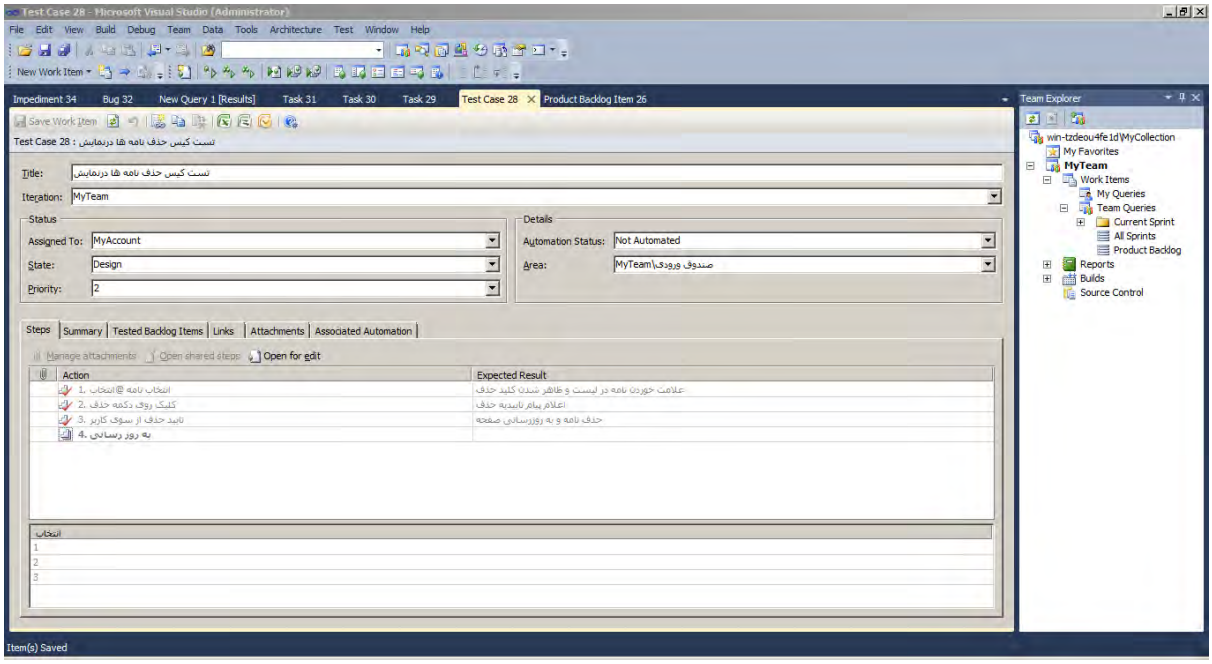
آزمون گر یک به یک اعمالی را که در بخش Active گام ها وجود دارد به صورت دستی بر روی نرم افزار اجرا می کند و نتیجه عمل با نتیجه ذکر شده در گام مورد نظر مقایسه می کند. اگر نتیجه مقایسه شده یکسان باشد از بخش انتهای گام گزینه Pass و اگر متفاوت باشد گزینه Fail انتخاب می شود. در صورتی که گزینه Fail انتخاب شود یک کادر ورودی باز خواهد شد تا در صورت نیاز آزمون گر بتواند توضیحی درج نماید. با پایان یافتن همه همه گام ها باید برای تثبیت نتایج گام های اجرا شده، کلید End Test کلیک شود. با پایان یافتن اجرای یک تست کیس دو حالت ممکن پیش خواهد آمد. اگر همه گام ها پاس(Pass) شده باشند، خود تست کیس نیز پاس شده تلقی می شود اما اگر حداقل یکی از گام ها شکست بخورد (Fail) کل تست کیس شکست خواهد خورد. اگر برای گامی پاس شدن و یا شکست خوردن آن تعیین نشود، به طور پیش فرض پاس شده تلقی می گردد(شکل 9-40).

شکل 40-9: فرآیند پاس کردن گام ها در Test Manager

با هر بار اجرای مجدد یک تست کیس نتایج قبلی برای آن پاک شده و تست کیس به حالت اولیه بر خواهد گشت.

اگر هر با از پارامترها استفاده شود گامی که دارای پارامتر است به تعداد مقادیر پارامتر تکرار خواهد شد. این پارامترها مقادیر و نتایج منتظره را نمایش می دهند(توجه شود که امکان عدم اجرای پارامتر در صورت فارسی بودن آن وجود دارد. هنگام رویارویی با این مشکل صرفاً نیاز است نام پارامتر به صورت لاتین نوشته شود). برای ردگیری و مشاهده گزارشات تست کیس ها می توان از دو تب دیگر به نام های Track و Organize استفاده نمود.

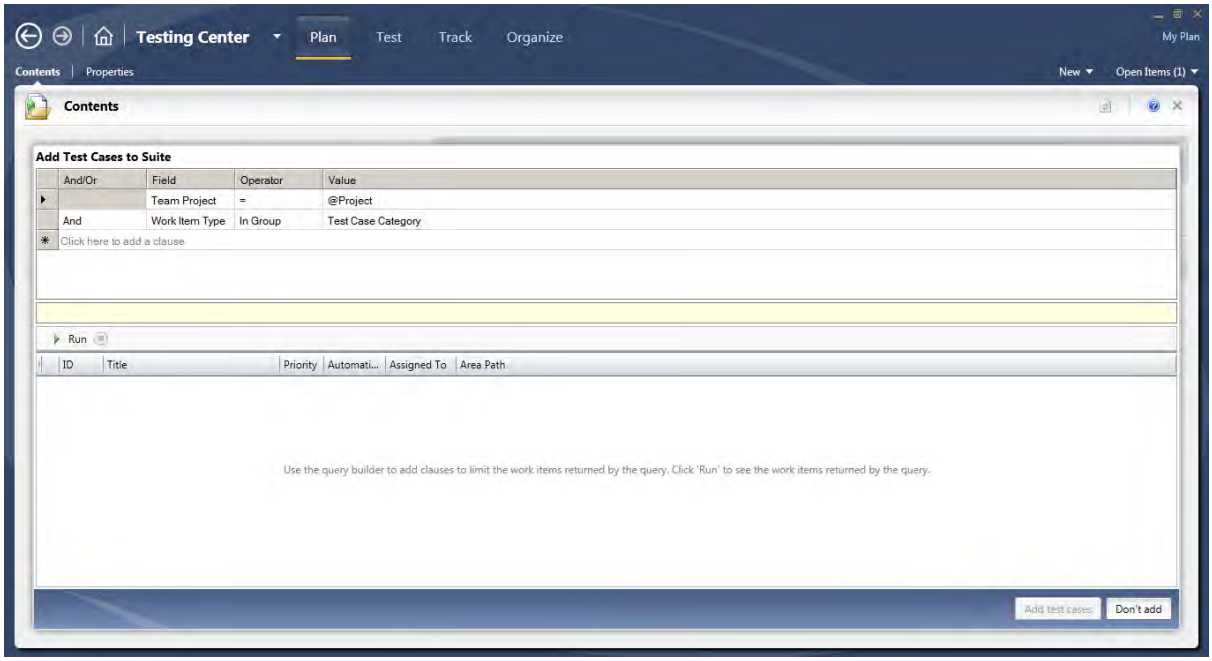
ابتدای این بخش ذکر شد که برای ایجاد تست کیس دو راه وجود دارد. راه اول استفاده از Test Managment بود که به طور مفصل بحث شد. راه دوم استفاده از Team Explorer می باشد. با اضافه کردن یک تست کیس در Team Explorer، مشاهده می شود که محیط تب Test Case عیناً مشابه محیط Test Managment است(شکل 9-41).



شکل 41-9: آیتم Test Case در Visual Studio

اما یک تفاوت مهم در این بین وجود دارد؛ Team Explorer قادر به ایجاد Shared Steps ها نمی باشد. ایجاد و تعریف Step ها خاص نرم افزار Test Manager است. اگر به تب Step در محیط Team Explorer دقت شود، کلیدی مشاهده می شود به نام Open for edit این کلید باعث اجرای نرم افزار Test Manager می شود. با اجرای شدن این نرم افزار می توان از تب Plan و با استفاده از کلید Add که در جعبه ابزار وجود دارد به اضافه کردن تست کیس مورد نظر(تست کیسی که در Test Manager ایجاد شده است) مبادرت کرد. کلیک روی این گزینه پنجره ای را برای جستجوی آیتم های تست کیس موجود در TFS ظاهر خواهد کرد. این پنجره در بخش های فوقانی خود شامل یک کوئری برای یافتن همه تست کیس های پروژه جاری است.

در بخش پایین نیز لیست آیتم های تست کیس نتیجه نمایش داده می شود. جهت اجرای این کوئری (جستجو)باید دکمه Run واقع در میانه صفحه فشرده شود. پس از اجرای کوئری اگر جستجو، نتیجه ای در بر داشت، در بخش پایین لیست خواهد شد. از لیست ظاهر شده می توان بر روی تست کیس مورد نظر کلیک کرده و برای اضافه شدن آن تست کیس به نرم افزار Test Manager کلید ...Add فشرده شود. با این عمل تست کیس مذکور در لیست تست کیس در تب Plan ظاهر می شود. حال تست کیس اضافه شده قابل مدیریت و اجرا خواهد بود. سایر موارد تست کیس برای هر دو نرم افزار مشترک است(شکل 42-9).

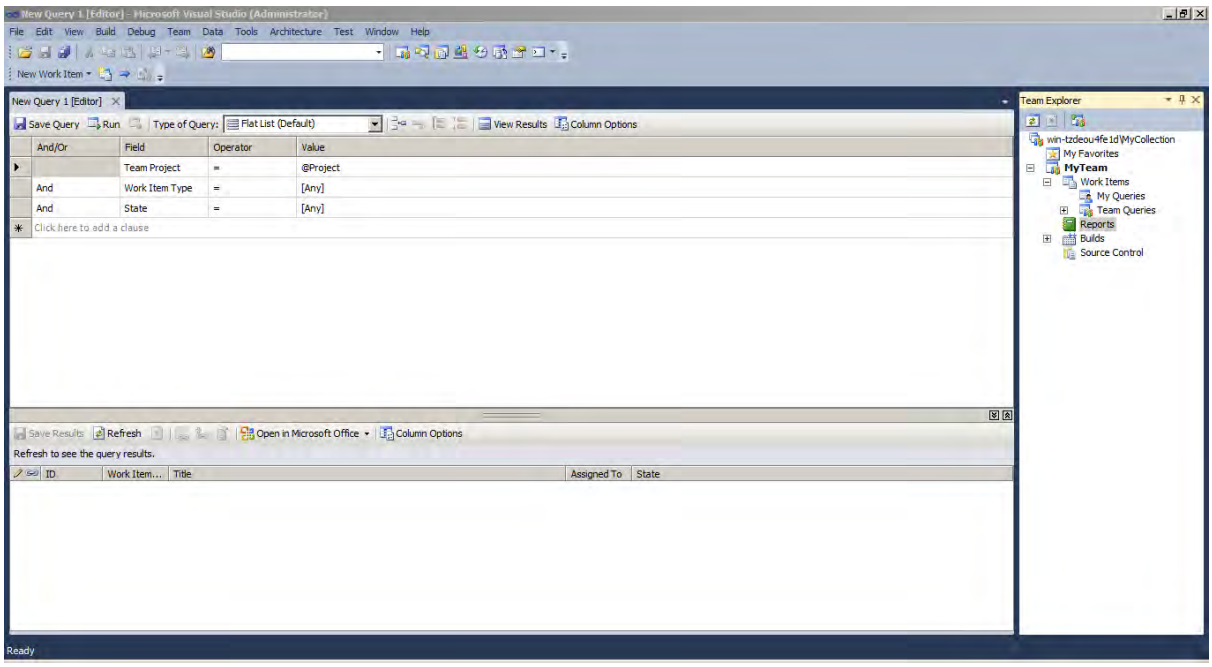


شکل 42-9: افزودن یک آیتم Test Case از پیش تعریف شده به پروژه در Test Manager

فصل دهم: کوئری ها

کوئری ها ابزاری هستند که جستجوی آیتم ها را در TFS ممکن می سازند. جستجو توسط کوئری ها توسط محیطی ویژوال(بصری) صورت می پذیرد. هر آیتمی با هر حالتی که در TFS وجود داشته باشد از طریق کوئری ها قابل دستیابی است. کوئری ها آیتم ها را نسبت به شرط ها و مقادیر آنها در خروجی لیست می کنند. شرط ها در کوئری ها همان فیلدهایی هستند که برای آیتم ها مقدار دهی شده اند. TFS از تعداد شرط های زیادی برای اجرای درخواست (کوئری)ها استفاده می کند بنابراین یک آیتم در بدترین حالت نیز با افزایش تعداد شرط های مورد جستجو قابل ردیابی است.

برای ایجاد یک کوئری جدید کافی است از منوی Team بر روی گزینه New Query کلیک شود. این عمل باعث باز شد تب جدیدی خواهد شد(شکل 1-10).



شکل 1-10: یک کوئری جدید

این تب شامل دو قسمت افقی است: ایجاد کوئری (بخش فوقانی) و نتایج جستجو(بخش زیرین). هر دو بخش شامل ابزارهای هستند که در ادامه تشریح خواهند شد.

بخش فوقانی تب کوئری شامل یک جدول برای ورود شرط های جستجو است. هر سطر در حقیقت معادل یک شرط جستجو می باشد. برای افزدون یک شرط کافی است بر روی قسمت Click here to add a clause که به صورت کم رنگ نوشته شده است کلیک شود. این عمل به صورت خودکار باعث اضافه شدن یک شرط به لیست شرط های جستجو می شود. البته باید دقت شود که اگر برای ستون Field سطر جدید مقداری تعیین نشود سطر ثبت نخواهد شد.

جدول شرط ها شامل 4 ستون می باشد. And/OR، Field، Operator و Value. ستون اول مشخص کننده حالت شرکت پذیری شرط در جستجو است. این شرکت پذیری می تواند دو حالت داشته باشد: And و OR. این حالت ها نسبت به شرط های قبلی خود تعیین می شوند و حالات مستقلی نیستند. حالت And شرکت پذیری شرط را نسبت به شرط قبلی خود اجباری می کند. به این معنی که از مجموع آیتم هایی که از جستجوی شرط های قبلی منتج می شوند آیتم هایی در خروجی نمایش داده می شوند که مقدار شرط ی که حالت And دارد در آنها صدق کند. حالت OR حالتی عکس حالت قبل دارد. این حالت نیز بر اساس شرط های پیشین خود تعیین می شود، با این تفاوت که از آیتم هایی در خروجی استفاده می شوند که یا در مقدار شرط قبلی صدق کند و یا در مقدار شرط ی که حالت OR را دارد و یا در هر دوی آنها.

ستون Field در حقیقت همان شرط را تعیین می کند. اگر لیست این ستون در یکی از سطرها گشوده شود ملاحضه می شود که همه فیلدهایی که در فصل قبل راجع به آنها صحبت شده، در آن وجود دارد. TFS از این

طریق قابلیت جستجوی بسیار پیشرفته ای را در اختیار قرار می دهد. لیست مذکور بر اساس حروف الفبا مرتب شده اند. قابلیت جالبی که در این بخش تعبیه شده است این است که لیستی از شرط (فیلد)هایی که اخیراً استفاده شده است در بالای لیست اصلی شرط ها توسط حائلی نمایش داده می شود.

ستون Operator عملگر مقایسه را برای هر کدام از شرط ها تعیین می کند. عملگرهای متفاوتی برای دقیق تر کردن جستجو تعبیه شده است. در ابتدای لیست عملگرها، عملگرهای معمول مقایسه اعداد و حروف قرار دارد. در انتهای لیست عملگرهایی شبیه به عملگرهای مقایسه ای معمولی وجود دارد با این تفاوت که در سمت راست هر کدام عبارت [Field] قرار داده شده است. این عملگرها هنگامی استفاده می شوند که نیاز باشد بین دو فیلد مختلف از یک آیتم واحد، مقایسه ای صورت پذیرد. فیلد اول در ستون Field انتخاب می شود و برای تعیین فیلد دوم ابتدا باید عملگر مقایسه ای را برابر یکی از این عملگرها قرار داد.

فیلد دوم در ستون مقدار (Value) تعیین می شود. توجه شود که هر فیلدی را نمی توان در این لیست انتخاب نمود و لیست فیلدهای مقایسه دوم لیست لیستی محدود و از پیش تعیین شده ای است. علاوه بر این عملگرها، عملگرهای دیگری نیز در این لیست مشاهده می شود. عملگر Countains عملگری است که بیشتر برای مقایسه کاراکتری استفاده می شود. اگر مقدار شرط (آنچه که در ستون Value ذکر شده) در مقدار آیتمی موجود باشد، آن آیتم برگردانده می شود. به عبارت دیگر آیتمی در خروجی نمایش داده می شود که کاراکترهای ستون مقدار، زیر مجموعه ای از فیلد آن آیتم باشند.

عملگر In Group عضویت فیلد را در گروهی خاص بررسی می کند. در این لیست عملگر جالبی وجود دارد به نام Wes Ever که مقادیر قبل یک فیلد را بررسی می نماید. این عملگر برای مواقعی که لازم باشد در سابقه فیلد آیتم ها جستجویی انجام شود. برخی عملگرها حالت منفی دارند که بدیهی است نتایجی عکس حالت طبیعی آنها نمایش داده می شود. نکته بسیار مهمی در استفاده از این عملگرها وجود دارد؛ کارکرد عملگرها نسبت به هر فیلد متفاوت است. حتی یک عملگر می توان برای فیلدهای مختلف لیست مقادیر متفاوتی داشته باشد. برخی دیگر از عملگرها برای فیلدهای خاصی تعریف شده اند و ممکن است برای هر فیلدی نمایش داده نشوند.

ستون Value آخرین ستون از جدول تعریف کوئری است. پس از تعیین حالت And/OR، شرط (فیلد) و عملگر حال نوبت آن است که مقدار مورد نظر جهت جستجو تعیین شود. نوع ورودی این ستون بر حسب فیلد(شرط) و نوع عملگر مورد استفاده متفاوت است. ورودی ستون Value می تواند متن و یا یک لیست باشد. ورودی لیست برای مواقعی استفاده می شود که مقادیر تعیین شده و محدود باشند. ورودی متن نیز هنگامی استفاده می شود که نیاز به ورودی مقداری از سوی کاربر باشد.

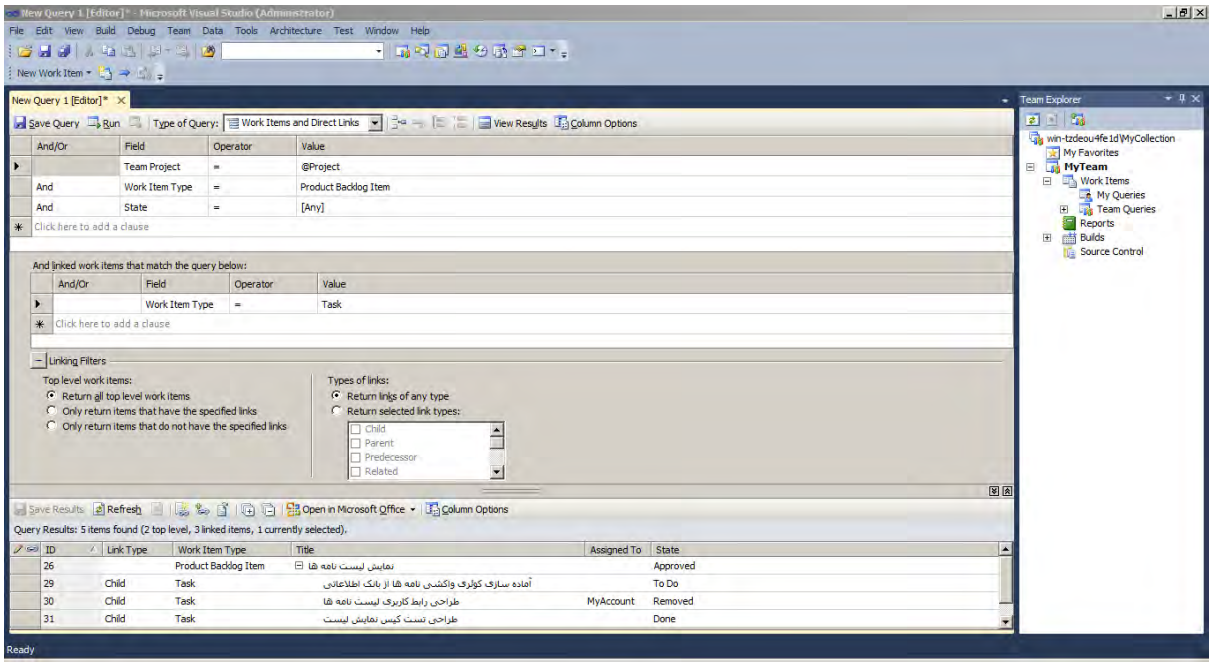
انوع کوئری

یافتن آیتم های مستقل یکی از اعمالی است که کوئری ها می توانند انجام دهند. از آنجایی که همه آیتم های درون TFS آیتم های مستقلی نیستند بنابراین نیاز به قابلیتی است تا با استفاده از آن پیوند آیتم ها نیز جستجو شود. نوعی از کوئری که در قسمت قبل مشاهده شد لیست مسطح نام دارد. به این معنی که کوئری در لیستی از آیتم های بدون پیوند و مستقل تشکیل می شود. علاوه بر لیست های مسطح دو نوع کوئری دیگر نیز وجود دارد: Work Items and Direct Links(آیتم های کاری و لینک های مستقیم) و Tree of Work Items(درخت آیتم های کاری).

برای هر کوئری جدید اولین سطر، انتخاب پروژه تیم می باشد. این شرط با فیلدی به نام Team Project مشخص می شود. عملگر آن عملگر معمول مساوی(=) است. در بخش مقدار نام تیمی که قرار است جستجو در آن انجام شود نوشته می شود. اگر به جای نام تیم از عبارت @Project نوشته شود از تیم جاری به عنوان تیم مورد جستجو استفاده می شود.

در نوع Work Items and Direct Links آیتم های کاری به همراه پیوندهای مستقیم آنها نمایش داده می شوند. منظور از پیوند مستقیم، پیوندی است که بدون واسطه به آیتم متصل شده است مثلاً پیوند آیتم بک لاگ و وظایفش پیوندی مستقیم است و پیوند بین آیتم بک لاگ و وظایف تست کیس مرتبط با آن پیوندی غیر مستقیم است. پیوندهای یک آیتم کاری هنگام نمایش بلافاصله پس از آن آیتم قرار می گیرند. وجه تمایز این پیوندها و آیتم های اصلی، وجود تورفتگی در آنها می باشد.

برای انتخاب نوع Work Items and Direct Links باید از جعبه ابزار کوئری در لیست Type of Query گزینه Work Items and Direct Links را گزیش کرد. این کار باعث تغییر محیط کاربری کوئری می شود(شکل 2-10).



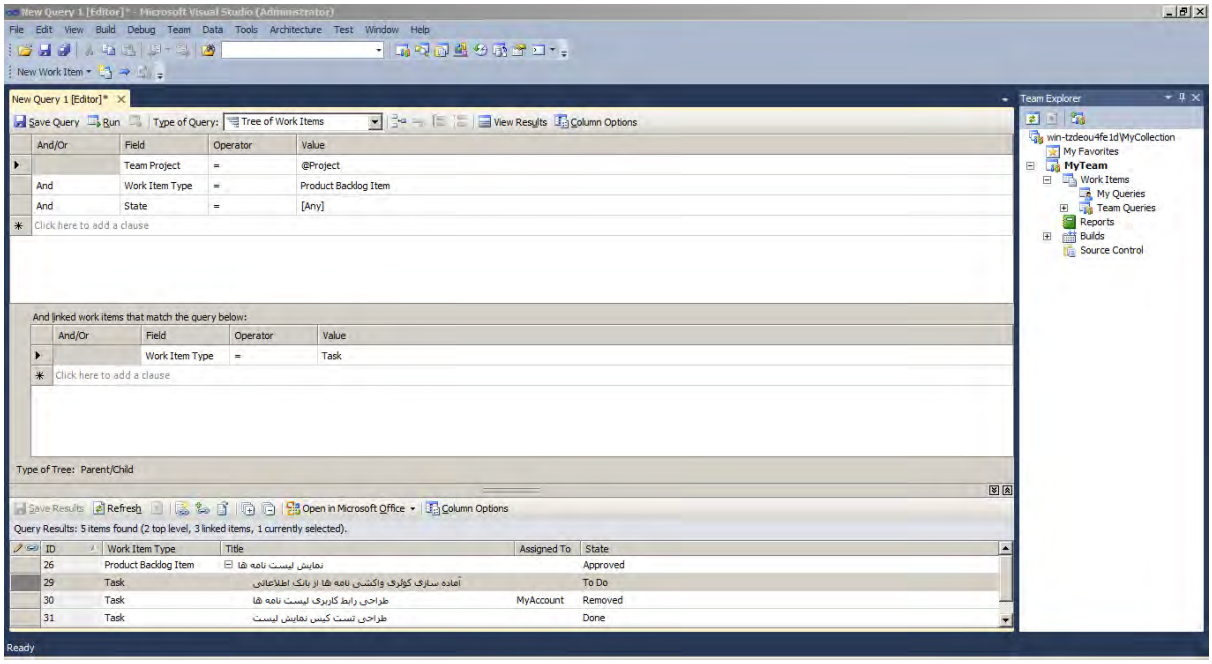
شکل 2-10: انتخاب نوع کوئری Work Items and Direct Links

این تغییر با اضافه شدن یک بخش کوئری جدید و یک سری تنظیمات دیگر همراه است. بخش کوئری جدید با تورفتگی ای نسبت به بخش اصلی نمایان می شود. این بخش برای فیلتر کردن (جستجو) آیتم های پیوندی در آیتم اصلی می باشد. این دو بخش به این صورت استفاده می شود که کاربر ابتدا شرط های آیتم های مورد نظر خود را در بخش اول وارد می کند سپس شرط های آیتم هایی را که مایل است به ازای پیوندن با آیتم اصلی نمایش داده شود تنظیم می کند. به طور پیش فرض همه انواع آیتم های پیوندی که شرط ها برای آنها صدق می کنند نمایش داده می شود.

در پایین بخش کوئری دوم تنظیماتی به نام Linking Filters گنجانده شده است. این تنظیمات شامل دو بخش است. بخش سمت چپ که Top level work items نام دارد مخصوص فیلتر کردن نمایش آیتم های اصلی بر اساس داشتن آیتم های پیوندی است. در قسمت Top level work items سه انتخاب وجود دارد: انتخاب اول Return all top level work items می باشد که تمام آیتم های کاری یافت شده در قسمت کوئری دوم اعم از آنهایی که دارای پیوند می باشند و آنهایی که نمی باشند را بر می گرداند. انتخاب دوم Only returns items that have the specified links فقط آیتم هایی را بر می گرداند که دارای پیوند هستند و انتخاب سوم Only returns items that do not have the specified links آیتم هایی را که دارای هیچ گونه پیوندی نمی باشند را بر می گرداند. این انتخاب ها برای زمان هایی که نیاز است بین آیتم های دارای پیوند و آیتم هایی که پیوندی ندارند تمیز قائل شویم مفید واقع می شوند.

بخش سمت راست Linking Filters با عنوان Type of Links، تنظیماتی ارائه می دهد که توسط آن می توان نوع آیتم های پیوندی را تعیین کرد. این تنظیمات شامل دو انتخاب است. انتخاب اول Return links of any type همه انوع پیوندها اعم از Child، Parent و... را برمی گرداند. انتخاب دوم Return selected link types می باشد که از طریق لیستی می توان به انتخاب انواع خاصی از پیوندها مبادرت نمود.

نوع سوم کوئری ها Tree of Work Items نام دارد. با انتخاب آن از لیست Type of Query محیطی شبیه به حالت دوم نمایش داده می شود. با این تفاوت که بخش تنظیمات از آن حذف شده است(شکل 3-10).



شکل 3-10: انتخاب نوع کوئری Tree of Work Items

این نوع کوئری همان طور که از نامش پیدا است آیتم های پیوندی را به صورت درختی نمایش می دهد. تنها نوع پیوند پشتیبانی شده در این کوئری والد/فرزندی (Parent/Child) می باشد. کادردوم مخصوص جستجو فرزندان آیتم هایی می باشند که در بخش کوئری اول یافت شده است. فرزند یا فرزندانش که در بخش دوم تطبیق شوند به صورت درختی در پایین آیتم مربوطه خودشان لیست می شود. نکته مهمی که در کوئری دوم وجود دارد این است که با جستجو هر فرزند، خود فرزند بر اساس شرط های کوئری دوم برای داشتن فرزند جستجو می شود و گاهی سلسله مراتب چند لایه ای را ایجاد می کند. فرزندان یک فرزند به شرطی نمایش داده می شوند که شرط های کوئری دوم در آنها نیز صدق کند. این سلسله مراتب تا جایی که رابطه والد فرزندی وجود داشته باشد ادامه می یابد. از طریق این نوع کوئری می توان آیتم هایی را که رابطه مستقیم با آیتم اصلی ندارند را نیز جستجو کرد؛ همانند رابطه یک آیتم با نوه خود!(فرزندِ فرزند آن آیتم).

اجرای کوئری

پس از تعیین شرط های لازم برای جستجو باید روی دکمه Run در جعبه ابزار کلیک کرد. با این عمل جستجو در کل آیتم های موجود در TFS بر اساس شرط ها وارد شده صورت می پذیرد و نتیجه آن در قسمت پایین صفحه نمایان می شود. در جعبه ابزار کلیدهای دیگری چون اضافه کردن یک شرط در بین شرط ها، حذف یک شرط ، گروه بندی شرط ها و خذف گروه بندی شرط ها به چشم می خورد. گروه بندی شرط ها برای مواقعی مفید است که بخواهیم ترکیب خاصی از And و OR را اعمال کنیم.

با گروه بندی، شرط ها با ترتیب خاصی اجرا می شوند. شرط های هر گروه صرف نظر از مکان آنها نسبت به یکدیگر انجام می شوند و سپس نتیجه آنها با دیگر گروه های شرط اجرا می شود. گروه ها مانند پرانتز در زبان کوئری هایی همچون SQL هستند. برای گروه بندی شرط ها ابتدا باید شرط های مورد نظر انتخاب شوند(توجه شود که شرط های هم گروه باید کنار یکدیگر قرار بگیرند). برای انتخاب شرط ها می بایست از یکی از کلیدهای ctrl و یا Shift استفاده شود. پس از انتخاب شرط ها با کلیک روی دکمه Group Clause می توان شرط ها را گروه بندی کرد. شرط های گروه بندی شده با خط خمیده ای (شبیه به پرانتز) در سمت چپ آنها قابل تشخیص هستند(شکل 4-10).

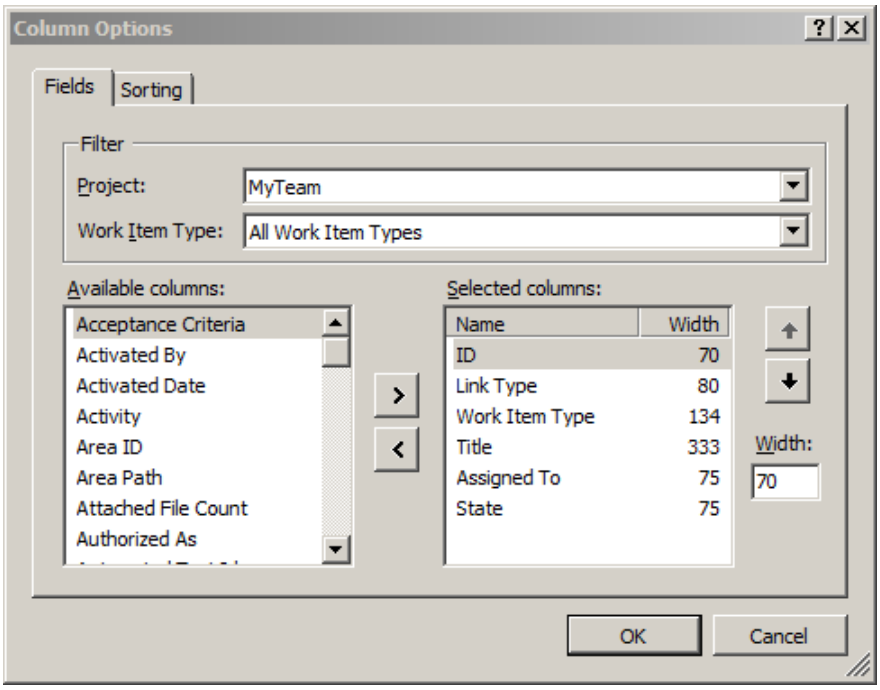
	And/Or	Field	Operator	Value
▶		Team Project	=	@Project
	And	Work Item Type	=	Product Backlog Item
	And	State	=	[Any]
*	Click here to add a clause			

شکل 4-10: گروه بندی شرط ها

کلیدی به نام View در جعبه ابزار موجود است که کار آن نمایش جستجو در یک تب جداگانه است. در تب جدید خبری از بخش کوئری نمی باشد. نتایج در بخش فوقانی نمایش داده می شوند و در بخش پایین نمایشی از آیتم های انتخابی در بخش نتیاج دیده می شود. این نمایش عیناً مشابه همان نمایشی است که در تب های جداگانه آیتم دیده می شود.

در بخش نتایج جستجو نیز ابزارهایی وجود دارد که به معرفی برخی از آنها خواهیم پرداخت. کلید Refresh، همانطور که از نام آن پیدا است باعث تازه سازی نتایج جستجو می شود. این کلید در حقیقت باعث اجرای مجدد جستجو خواهد شد و با کلید Run کاکرد مشابهی دارد. در جعبه ابزار سه کلید دیگر به نام های New Linked Work Item، Link to an Existing Item و Open نیز به چشم می خورد. کاکرد این کلید سابقاً مورد بررسی قرار گرفته است. کلیدهای مذکور به ترتیب باعث ساختن و لینک کرد یک آیتم جدید به آیتم انتخاب شده، لینک کردن یک آیتم موجود به آیتم انتخاب شده و باز کردن آیتم انتخاب شده از لیست نتایج می شوند.

دو کلید برای کار با سلسله مراتب آیتم ها تعبیه شده است. کلیدهای Collapse All و Expand All به ترتیب باعث بسته شدن و باز شدن همه زیر مجموعه های آیتم ها به صورت یکجا خواهند شد. کلید دیگری که در این جعبه ابزار وجود دارد کلید Column Options می باشد. مشابه این کلید در بخش های دیگر نیز مشاهده شده است. فشردن این کلید باعث اجرای پنجره Column Options خواهد شد(شکل 5-10).

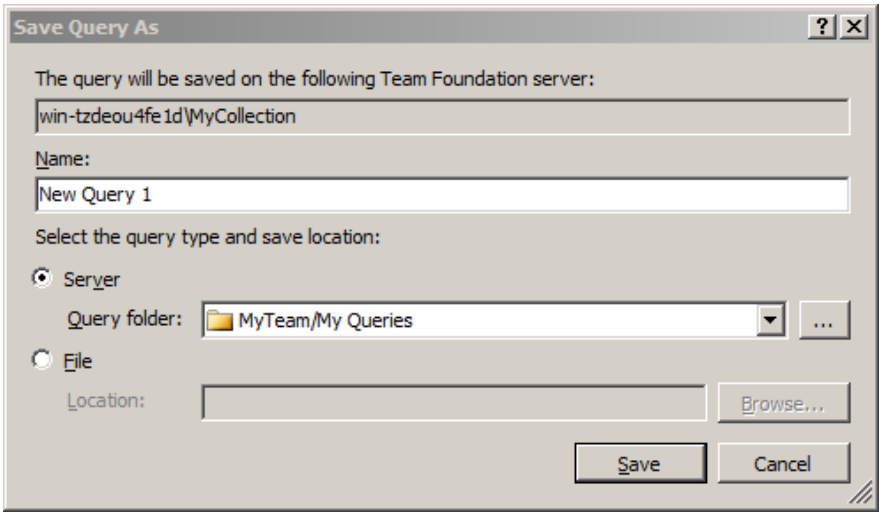


شکل 5-10: پنجره انتخاب ستون های نتایج

کاربران می توانند از این بخش به حذف و اضافه ستون های قابل نمایش در نتایج اقدام کنند. به صورت پیش فرض در این قسمت ستون های ID، Work Item Type و ... وجود دارد. از طریق این پنجره می توان ستون ها، نوع مرتب سازی، ترتیب و ستون ها و اندازه ستون ها را معین نمود.

کوئری های ذخیره شده

احتمال آن وجود دارد که برخی از کوئری ها به کرات مورد استفاده قرار بگیرد؛ از این رو در TFS قابلیت ذخیره کوئری ها مهیا شده است. هنگام ذخیره کوئری ها صرفاً شرط ها و تنظیمات ذخیره خواهد شد و نه نتایج آنها. این موضوع باعث می شود که هر گاه یک کوئری ذخیره شده اجرا می شود اطلاعاتی که در نتایج نمایش می دهد به روز می باشد. برای ذخیره یک کوئری کافی است روی کلید Save Query کلیک شود. با ایت عمل پنجره Save Query As گشوده خواهد شد(شکل 6-10).

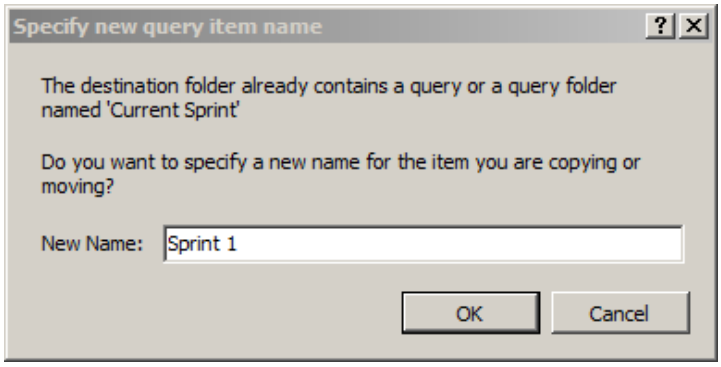


شکل 6-10: پنجره ذخیره کوئری

این پنجره نام و مکان ذخیره کوئری را به عنوان ورودی از کاربر دریافت می کند. برای انتخاب مکان ذخیره دو گزینه موجود است. از طریق گزینه اول کوئری بر روی سرور TFS و در پوشه ای کاربر مشخص می کند ذخیره می شود. این روش بهترین و معمول ترین روش ذخیره سازی کوئری ها است. در روش دوم کوئری به صورت یک فایل فیزیکی ذخیره می شود. پسوند این فایل wiq. می باشد. کاربر به وسیله فیلد Location و با کلید Brows می تواند مکان فایل را گزینش نماید.

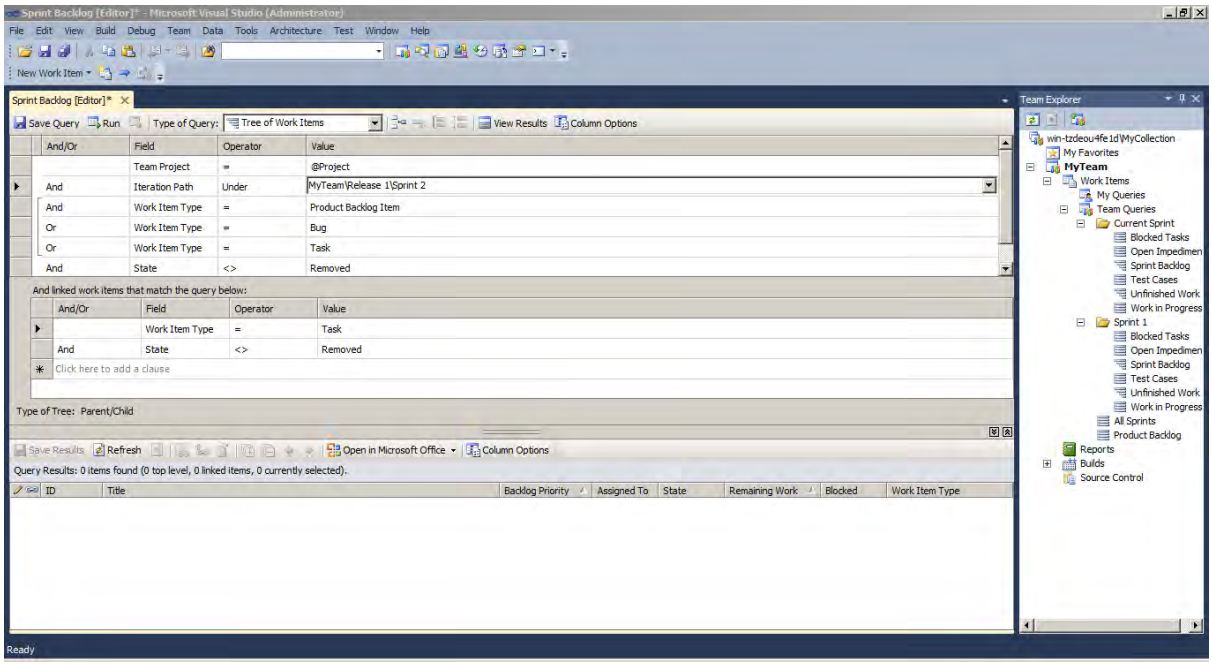
پوشه از پیش ساخته شده ای وجود دارد به نام My Favorities که در صورت انتخاب مکان ذخیره Server، مسیر پیش فرض ذخیره می باشد. کاربر می تواند با ساختن پوشه های مجزا، مسیر دلخواه خود را در بخش Query Folder به جای مسیر پیش فرض وارد کند.

پس از ذخیره سازی کوئری به راحتی می توان با دو بار کلیک روی آیکن آنها به اجرای کوئری ها اقدام کرد. علاوه بر کوئری هایی که کاربران ایجاد می کنند، تعدادی کوئری آماده و پر استفاده وجود دارد که کاربران از آنها در طول توسعه بهره می جویند. این کوئری ها را می توان در بخش Team Queries یافت. اگر در این لیست دقت شود، دو کوئری با نام های All Sprint و Product Backlog در ریشه و سایر کوئری ها در پوشه Current Sprint قرار دارند. کوئری هایی که در ریشه هستند مربوط به کل پروژه و کوئری هایی که در Current Sprint می باشد به اسپرینت جاری مرتبط هستند. پس از اتمام یک اسپرینت بدیهی است که باید کوئری ها با اسپرینت جدید و آیتم های آن همراه شوند. برای اضافه کردن کوئری های اسپرینت جدید ابتدا باید با راست کلیک روی پوشه Current Sprint و انتخاب گزینه Copy یک رونوشت از تهیه کرد. سپس باید این رونوشت را در بخش Team Queries الصاق (Past) نمود. از آنجایی که این کار باعث ایجاد دو نسخه هم نام از پوشه کوئری ها می شود، پنجره ای برای تغییر نام رونوشت جدید گشوده خواهد شد(شکل 7-10).



شکل 7-10: پنجره تغییر نام اسپرینت

پس از ورود نام جدید و فشردن کلید OK نسخه جدید الصاق می شود(بهتر است نام هر نسخه مطابق با نام اسپرینت آن انتخاب شود). این عمل باعث می شود که نسخه های اختصاصی از کوئری ها را برای هر کدام از اسپرینت ها داشته باشیم. اما نباید فراموش شود که همچنان کوئری Current Sprint به اسپرینت قبلی (که از آن رونوشت گرفته شد) اشاره می کند و آیتم های آن را در خروجی نشان می دهد. برای تغییر این وضعیت باید روی هر کدام از کوئری ها ویرایشی صورت بگیرد. با راست کلیک بر روی یک کوئری و انتخاب گزینه Edit، پنجره کوئری/نتایج برای آن کوئری گشوده خواهد شد(شکل 8-10).



شکل 8-10: تغییر کوئری Sprint Backlog

در شرط دوم مشاهده می شود شرط Iteration path مقداری برابر با Release 1\Sprint 1\... را داراست. تغییر این مقدار با اسپرینت دیگری باعث می شود از این پس اجرای کوئری، آیتم هایی مطابق با آن اسپرینت را به نمایش بگذارند. این عمل یک به یک برای همه کوئری ها باید اجرا شود.

تذکر: بک لاگ محصول یک لیست واحد است که در طول پروژه از آن استفاده می شود. با شروع هر Release این لیست پر و پس از پایان Release خالی می شود. پس بنابراین این لیست نیازی به کپی برداری و ثبت به ازای هر Release ندارد.

فصل یازدهم: گزارشات

گزارشات اسناد پر اهمیتی برای تصمیم گیری مدیران می باشند. مزیت بزرگ گزارشات مستند بودن آنها است. گزارشات باید به گونه ای تنظیم شوند که افراد مختلف با سطح اطلاعات متفاوت هنگام خواندن با مشکلی مواجه نشود. گزارشات باید دقیق، مرتب، صریح و به دور از هر گونه ابهامی تهیه شوند. این موضوع باعث می شود که نوشتن گزارشات به اندازه خواندشان عمل آسانی نباشد برای تهیه گزارشی مناسب باید زمان زیادی با آن اختصاص داده شود. این در حالی است که ما همواره برای توسعه نرم فزار با کمبود وقت مواجه هستیم. لازمه دقت و سرعت بالا در تهیه گزارش دو دلیلی هستند که استفاده از نرم افزار (کامپیوتر) را برای تهیه گزارشات نسبت به انسان برتری می دهد. برای این امر خطیر TFS ابزارو و امکاناتی را مهیا کرده که توسط آن افراد می توانند اقدام به تولید گزارشاتی جامع و خودکار نمایند.

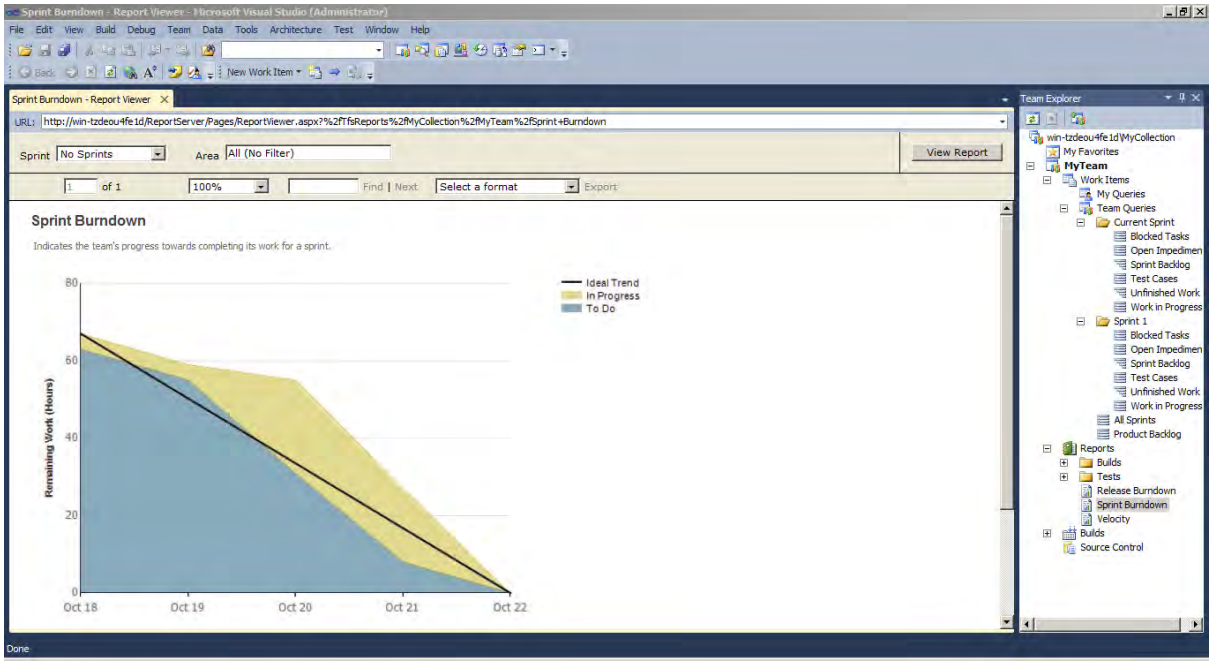
TFS از عناصر بصری برای تولید گزارشات بهره می جوید. به این معنی که گزارشات را به صورت نمودارها و جداول گرافیکی ایجاد می نماید. بخشی از گزاراشات نیز از طریق نرم افزارهای جانبی چون Excel تولید می شوند و برخی دیگر توسط بخش Report. برای استفاده از بخش Report باید پیکربندی TFS به صورت Advanced صورت گرفته باشد. همچنین نیاز به نصب Reporting Service بر روی SQL Server نیز می باشد. سیستم عاملی که TFS روی آن نصب می شود نیز حتماً باید از نوع سرور باشد. اگر چنین ملزوماتی تهیه شده باشد بخش Report در Team Explorer فعال خواهد شد.

انواع گزارشات

بخش Report شامل سه گزارش Release Burn Down، Sprint Burn Down، Velocity و دو پوشه دیگر به نام های Builds و Tests می باشد. با کلیک روی هر کدام از آن ها گزارش مربوطه تولید و نمایش داده می شود. با هر باز اجرای جدید این گزارشات TFS مجدداً گزارش را ایجاد می کند. این کار باعث می شود همیشه گزارشی به روز نمایش داده شود.

Sprint Burn Down

اجرای گزارش Sprint Burn Down باعث نمایش نمودار Sprint Burn Down می شود(شکل 1-11).



شکل 1-11: گزارش Sprint BurnDown

همانطور که در فصل های پیشین گفته شد Sprint Burn Down نموداری است که روند توسعه نرم افزار را در طول یک اسپرینت نمایش می دهد. این نمودار دارای خطی فرضی می باشد که اجرای ایده آل اسپرینت را نشان می دهد. این نمودار به صورت خودکار بر اساس اسپرینت جاری اجرا می شود. نمودار Sprint Burn Down دارای دو محور می باشد. محور افقی بر اساس زمان اسپرینت می باشد. نمودار برای عددگذاری این محور از تاریخ ابتدا و انتهای اسپرینت بهره می جوید. محور افقی بر اساس ساعت می باشد و مجموع کار باقی مانده

وظایف آیتم های موجود در بک لاگ اسپرینت نشان می دهد. این مجموع در حقیقت کل کاری می باشد که تیم باید در طول اسپرینت انجام دهد.

نمودار برای کارهایی که باید انجام و کارهای در حال انجام، نمایشی حجمی دارد. حجم زیرین میزان کارهایی را نشان می دهد که باید انجام شوند(To Do). این حجم در شروع اسپرینت تقریباً کل فضای نمودار را از آن خود دارد. پس از شروع اسپرینت، بخشی از فضای نمودار به حجم کارهای در حال انجام اختصاص می یابد. با پیشرفت اسپرینت حجم نمودار کاهش پیدا می کند. ایده آل این است که حجم نمودار تا رسیدن به تاریخ پایان صفر شود. در این نمودار برای کارهای انجام شده حجمی نشان داده نمی شود. برای بدست آوردن میزان کار انجام شده در یک زمان کافی است که کل کارها (نقطه شروع خط ایده آل) از حجم نمودار در زمان مذکور کم شود. آنچه بدست می آید میزان کارهای انجام شده (Done) تا زمان مشخص شده است. مزیت نمودارهای حجمی این است که مقایسه بین آیتم های نمودار را سهولت می بخشد. از طرفی به دلیل بصری بودن نمودار سریع الانتقال تر می شود.

Optionهایی برای اجرای گزارشات وجود دارد که انعطاف پذیری نمایش گزارش ها را بالا می برد:

Sprint: ترکیب Release/Sprint موجود را در یک لیست نمایش می دهد. با انتخاب هر کدام می توان Burn Down های مربوط به آن را مشاهده کرد. لازم به ذکر است که اسپرینت هایی در این لیست نمایش پیدا می کنند که دارای تاریخ شروع و پایان باشند(اسپرنت های سپری شده و اسپرینت جاری).

Area: فیلتری است برای محدود کردن وظایف در یک سطح خاص.

Current Page: در صورت چند صفحه بودن گزارش از طریق این فیلد می توان بین صفحات حرکت نمود.

Zoom: مقیاس بزرگ نمایی گزارش را تنظیم می کند. این مقیاس می تواند بین 10% تا 500% تغییر کند.

Find Text: قابلیتی است برای جستجوی متنی در گزارش.

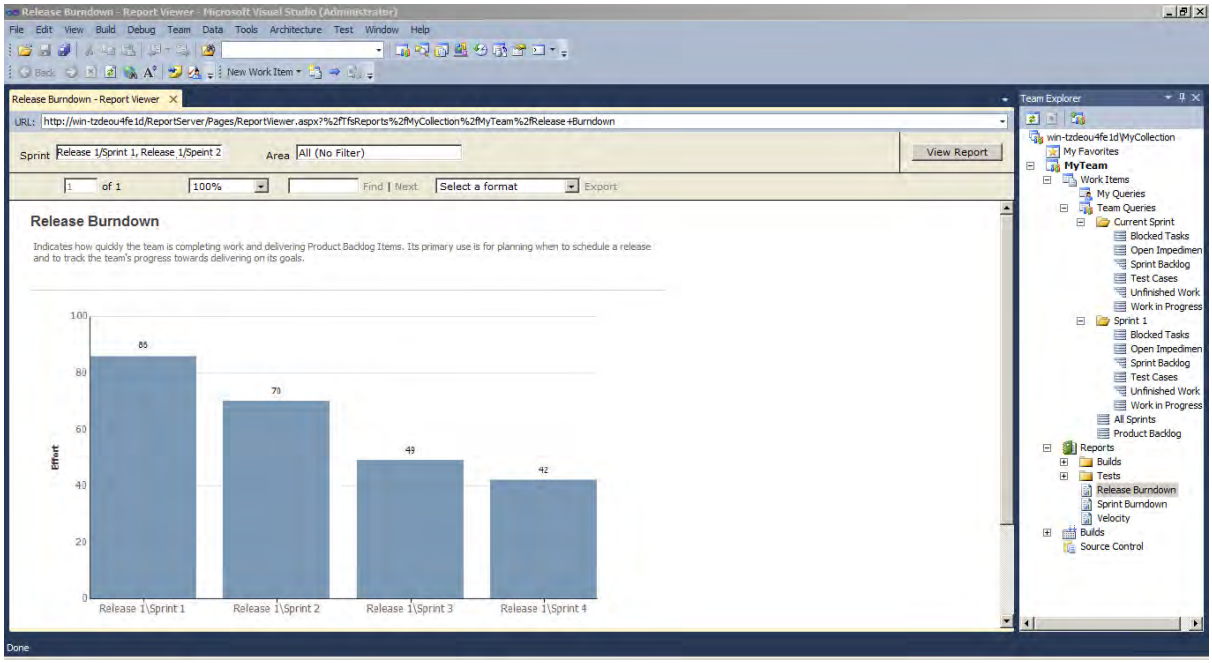
Export Format: این لیست نوع قالب خروجی نمودار را تعیین می کند. قالب هایی چون Xml، Excel و PDF از جمله پر کابردترین قالب ها هستند.

پس از تعیین Option های مورد نیاز می توان با کلیک بر روی دکمه View Report نمودار را به روزرسانی نمود.

Release Burn Down

این گزارش یکی دیگر از پرکاربردترین گزارشات موجود در TFS است. این گزارش به علت نوع خاصش بیشتر جنبه مدیریتی دارد. Release Burn Down نمودار ساده ای است. این نمودار نسبت کار انجام در اسپرینت را در Releaseهای مختلف با یکدیگر مقایسه می کند. Release Burn Down همانند Sprint Burn Down از دو محور تشکیل می شود. محور افقی ترکیب همه Release/Sprint های به کار گرفته شده می باشد(توجه شود که Release/Sprint هایی که تا کنون اجرا نشده اند در این گزارش وجود ندارند). محور عمودی بر حسب Story Point است و از مجموع برآورد (Effot) آیتم های یک اسپرینت بدست می آید. بدیهی است که اسپرینتی که بیشترین مقدار Effort را دارا است، بزرگترین عدد محور را تعیین می کند.

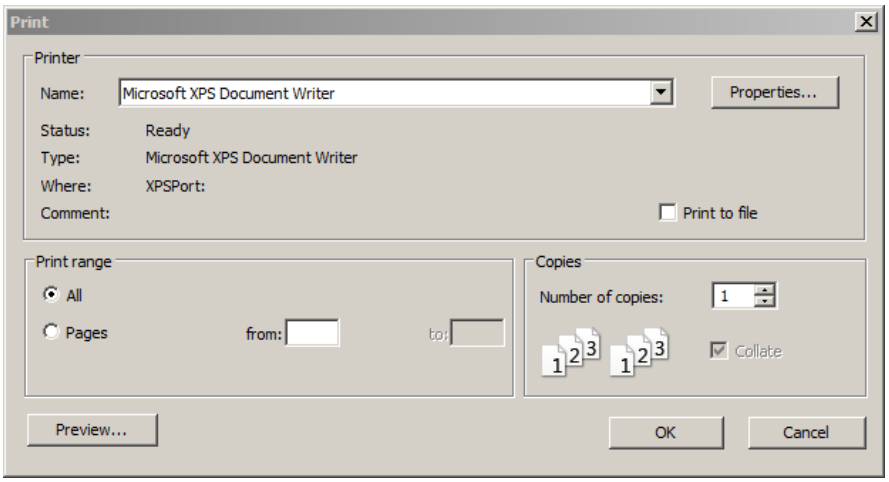
نمایش این نمودار به صورت ستونی می باشد. برای هر Release/Sprint یک ستون رسم می شود که میزان کار انجام شده (Effort) آن Release/Sprint را نشان می دهد(شکل 2-11).



شکل 2-11: گزارش Release BurnDown

Option هایی که در این گزارش قرار دارند همانند گزارش Sprint Burn Down می باشند، با این تفاوت که کارکرد فیلد Sprint در این گزارش کمی متفاوت است. با کلیک روی فیلد Sprint لیستی از Release/Sprint های موجود باز خواهد شد و در کنار هر کدام یک چک باکس (Check Box) وجود دارد. انتخاب هر کدام از Release/Sprint ها باعث می شود آن Release/Sprint در نمودار Release Burn Down درج شود. Release/Sprint هایی که انتخاب می شوند در فیلد Sprint به ترتیب نمایش داده می شوند. این Release/Sprint با حائل "," از یکدیگر تفکیک می شوند.

علاوه بر View Report دو دکمه دیگر نیز وجود دارد: Print و Refresh. دکمه Print باعث اجرای پنجره معروف Print می شود(شکل 3-11).

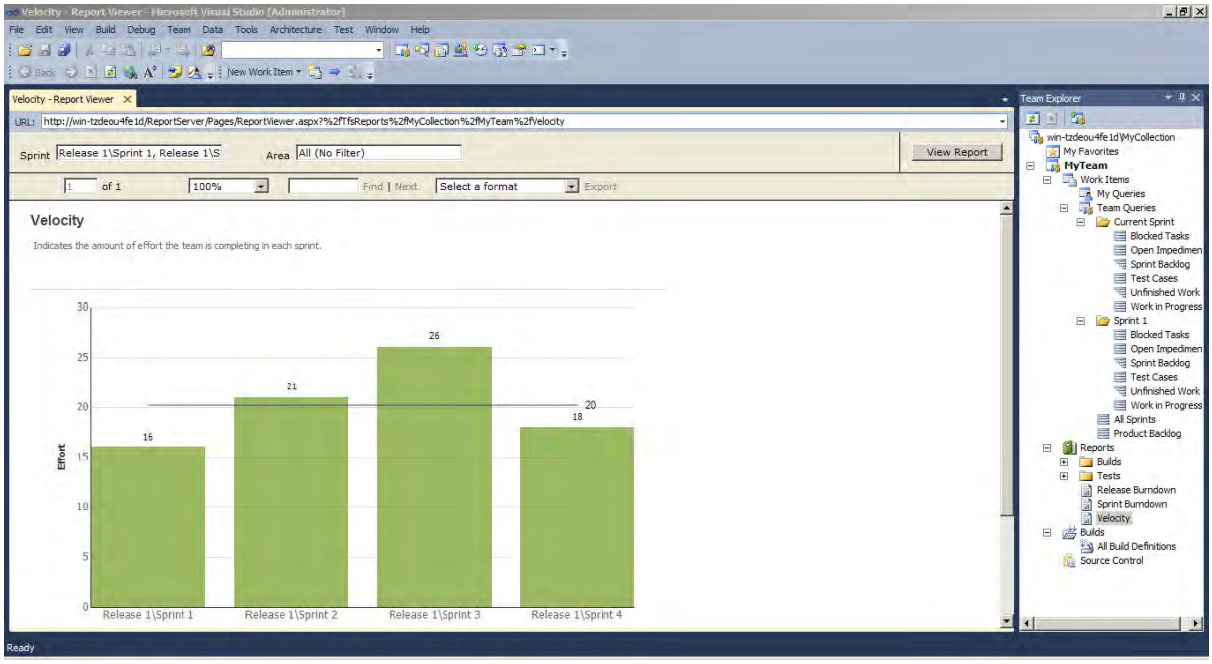


شکل 3-11: پنجره Print

از طریق این پنجره کاربر می تواند اقدام به چاپ گزارش تولید شده نماید. کلید Refresh نیز برای تازه سازی گزارش مورد استفاده قرار می گیرد. این کلید در مواردی عیناً مشابه کلیدView Report عمل می کند.

Velocity

Velocity گزارشی است که جهت تخمین سرعت اسپرینت های آینده مورد استفاده قرار می گیرد. شکل این نمودار درست مشابه نمودار Release Burn Down می باشد(شکل 4-11).



شکل 4-11: گزارش Velocity

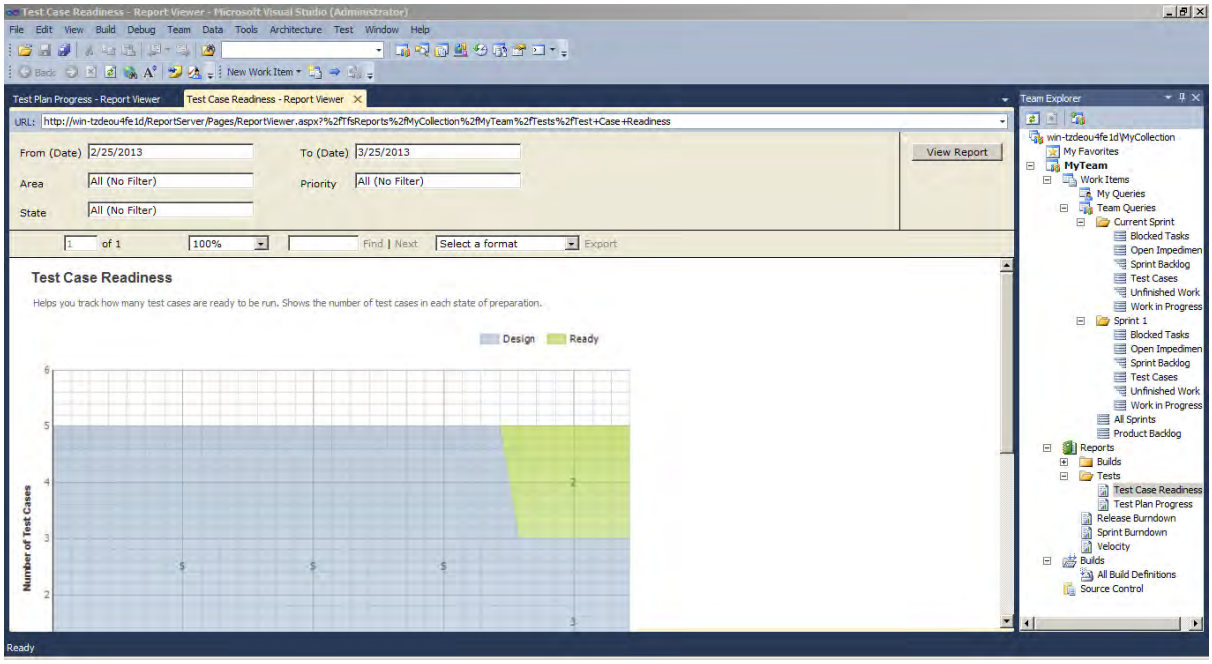
Velocity نیز مانند دو نمودار قبلی از دو محور تشکیل می شود. محور افقی لیستی از Release/Sprint های موجود را نمایش می دهد و محور عمودی مجموع Effort های یک اسپرینت را نشان می دهد.

میانگین این سرعت های نمایش داده شده توسط خطی افقی ارائه می شود. استفاده از این میانگین برای زمانی که میزان Effort های اسپرینت های مختلف نزدیک به یکدیگر هستند، یکی از مهمترین روش های تخمین سرعت اسپرینت های آینده می باشد.

Test: در این بخش دو گزارش موجود است که اختصاص به تست کیس ها دارد. با استفاده از این گزارشات می توان طراحی و اجرای تست کیس های موجود در TFS را تحت نظر داشت.

Test Case Readiness

این بخش مربوط به ارائه گزارشی از آماده سازی تست های توسعه است(شکل 5-11).



شکل 5-11: گزارش Test Case Reediness

این گزارش نموداری است دو محوره با جدولی در بین آن. محور افقی بر حسب زمان است و طراحی و آماده سازی تست ها در طول توسعه را نشان می دهد. محور عمودی نیز تعداد تست ها را نشان می دهد. نوع این نمودار حجمی است و شامل دو بخش Design و Ready می باشد. این نمودار نشان که چند تست در حال طراحی و چند تست در حال آماده باش هستند. تیم با دانستن میزان تست های آماده در برهه های زمانی مختلف، می تواند برنامه ریزی بهتری را برای اجرای آن تست ها در توسعه صورت دهد.

علاوه بر Option های شرح داده شده در بخش های قبل چهار Option جدید در بخش Test Case Readness وجود دارد:

From(Date): فیلتری است که تاریخ شروع طراحی تست ها را مشخص می کند.

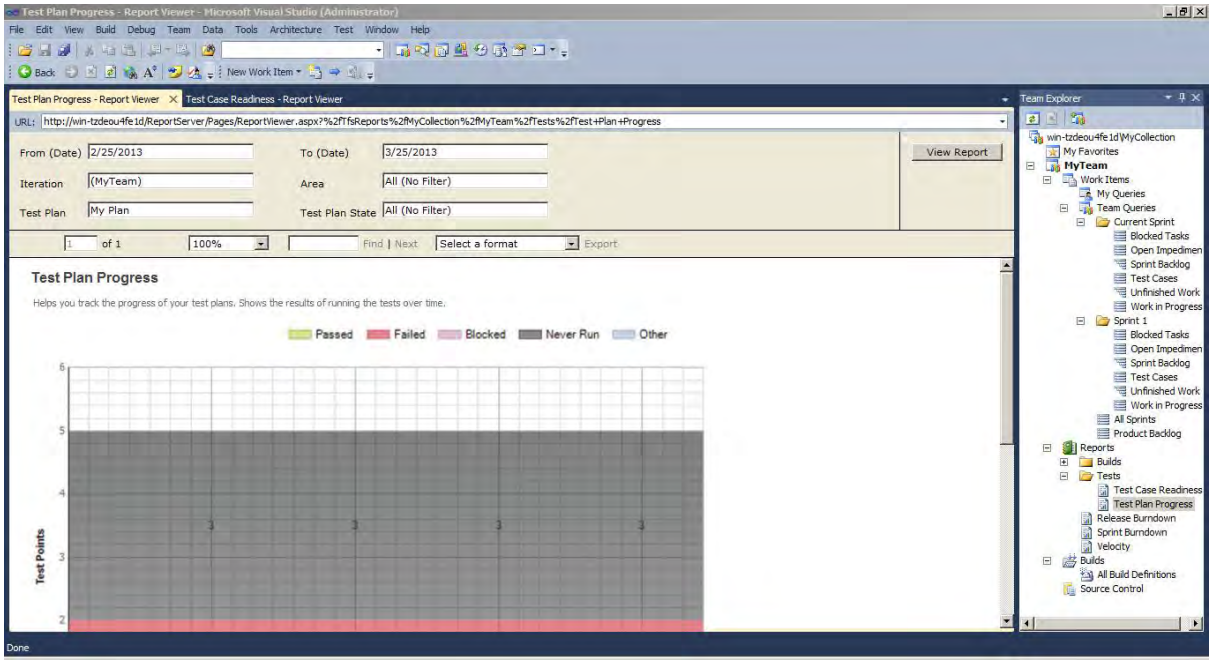
To(Date): فیلتری است که تاریخ پایان طراحی تست را مشخص می کند.

Priority: بر اساس این فیلد تنها اولویت های مشخص شده در آن نمایش داده می شوند.

State: این فیلد برای فیلتر سازی حالتی خاص از تست ها مورد استفاده قرار می گیرد.

TestPlan

این گزارش برای زمانی مناسب است که تیم نیاز به ردگیری اجرای تست ها دارد(شکل 6-11).



شکل 6-11: گزارش Test Plan

نموداری که از این گزارش تولید می شود، چگونگی اجرای تست ها را در طول توسعه نشان می دهد. گزارش TestPlan اجرای تست ها را در دسته بندی های زیر نمایش می دهد:

Passed: تعداد تست هایی که توسط آزمون گر پاس شده اند.

Failed: تعداد تست هایی که توسط آزمون گر تست شده اند.

Blocked: تعداد تست هایی که مسدود شده اند.

Never Run: تعداد تست هایی که هرگز اجرا نشده اند (لازم به ذکر است هر تست می تواند چند بار اجرا شود مثلاً پس از چند شکست تست پاس شود).

Other: تعداد تست هایی که اجرا شده اند و نتایج آنها حالتی غیر از Fail و Pass می باشد.

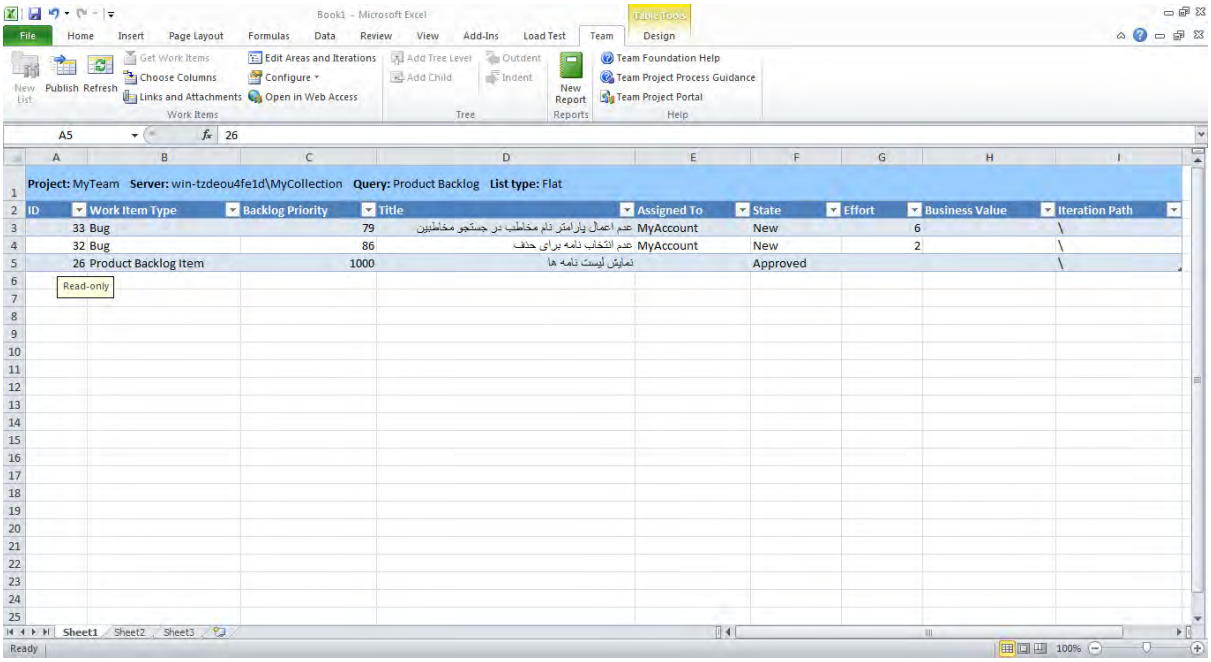
یکی از کاربردهای این گزارش نظارت بر کیفیت کلی نرم افزار از طریق بررسی چگونگی اجرای تست ها در طول توسعه می باشد.

در بیشتر بخش های Team Explorer ابزار به نام Refresh وجود دارد که باعث تازه سازی بخش های مربوطه می شود و آخرین تغییرات را در نمایش جاری اعمال می کند. دلیل طراحی چنین ابزاری اعمال تغییرات دیگر کاربران در نمایش کاری است. از آنجایی که TFS به صورت سرور اجرا می شود کاربران مختلفی ممکن است همزمان به آن وصل شوند. تغییر آیتم ها توسط یک بار به صورت مستقیم برای دیگر نمایش کاربران اعمال نمی شود. هر کدام از کاربران پس از تغییر یک آیتم توسط دیگر کاربران، می توانند با استفاده از ابزار Refresh به بروز رسانی آیتم مربوطه اقدام کنند.

گزارش گیری با استفاده از ابزار جانبی

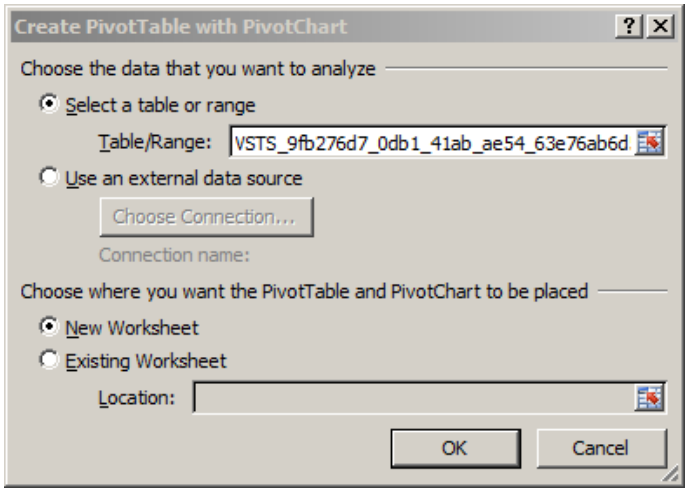
در فصل های پیش گفته شد که TFS پس از نصب با چند نرم افزار یکپارچه می شود. یکی از این نرم افزار MS Excel می باشد. برخی تیم ها از طریق این نرم افزار به تنهایی و نیاز به Team Explorer برای مدیریت توسعه استفاده می کنند. اما مهمترین ویژگی این نرم افزار گزارش گیری بسیار جامع و حرفه ای آن است. با استفاده از نرم فزار Excel می توان گزاراشاتی سفارشی با سرعت بالا و سهولت زیاد تهیه کرد. این گونه گزارشها را می توان از هر آیتمی اعم از نتیجه یک کوئری یا حتی به طور مستقیم خود کوئری ایجاد نمود.

برای ساخت گزارشات سفارشی کافی است در جعبه ابزار هر لیستی بر روی گزینه Open in Microsoft Office کلیک شود. برای تهیه یک گزارش از یک کوئری باید روی کوئری مورد نظر کلیک راست نموده و سپس از منوی باز شده Open in Microsoft Excel (Flat) انتخاب شود. این عمل باعث باز شدن نرم افزار MS Excel خواهد شد. با اجرای شدن نرم افزار جدولی شبیه آنچه در Team Explorer دیده شد، برای این لیست انتخابی مشاهده می شود(شکل 7-11).



شکل 7-11: نتایج یک کوئری در نرم افزار MS Excel

برای ایجاد یک نمودار باید از منوی(تب) Insert بر روی PivotChart (اولین گزینه از سمت چپ) کلیک شود. این کار باعث باز شدن پنجره Create PivotChart خواهد شد(شکل 8-11).



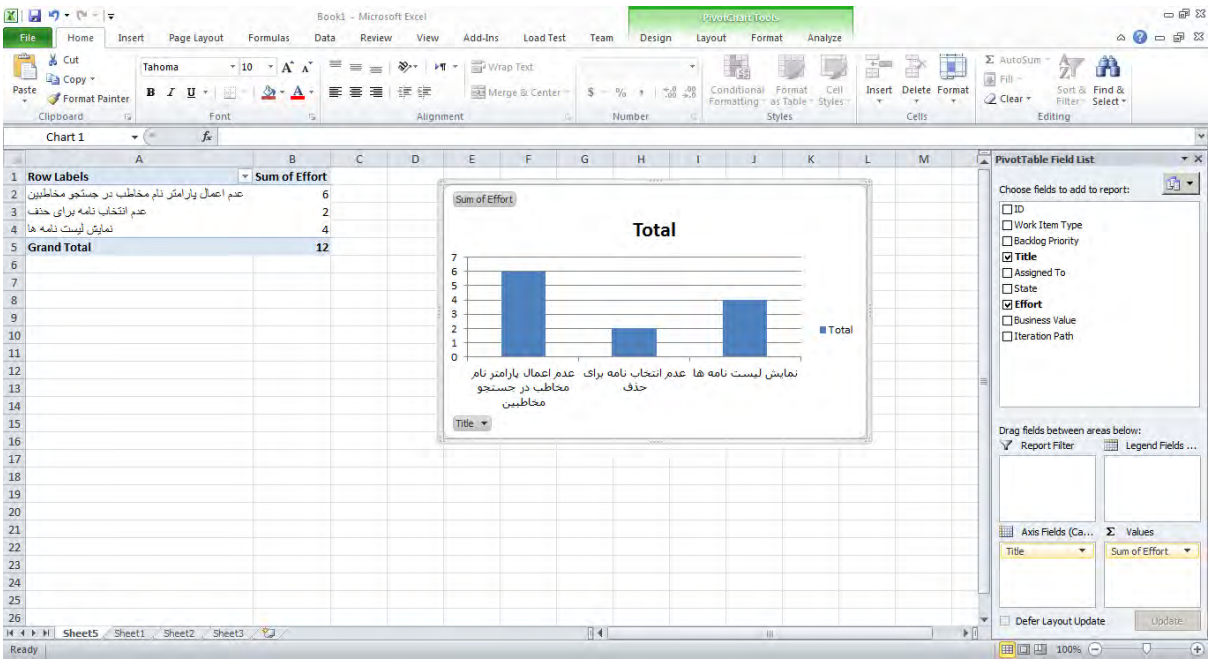
شکل 8-11: پنجره ساخت جدول و نمودار Pivot

این پنجره شامل دو بخش است: Choose the data... و Choose Where... در Choose the data... منبع داده ای که قرار است برای نمودار استفاده شود تعیین می گردد. این بخش شامل دو گزینه است. گزینه اول (Select a table or range) برای انتخاب یک جدول یا قسمتی (محدوده ای) از یک جدول استفاده می شود. محدوده انتخابی از طریق کادر خط چینی قابل مشاهده می باشد. برای انتخاب قسمتی از جدول کافی است بر روی دکمه ای که در انتهای فیلد Table/Range قرار دارد، کلیک شود. این عمل باعث کوچک شدن پنجره و تغییر فوکوس روی جدول پس زمینه می شود. حال کاربر می تواند محدوده ای از جدول را از طریق موس و کیبرد انتخاب نمود. با کلیک مجدد بر روی دکمه قبلی آدرس بخش انتخابی در فیلد Table/Range قرار می گیرد. گزینه

دوم این بخش Use an external data source می باشد و زمانی مورد استفاده قرار می گیرد که بخواهیم از یک منبع خارجی برای تولید نمودار استفاده کنیم.

بخش ...Choose Where مربوط به حل اجرای نمودار PivotChart می باشد. این بخش هم شامل دو انتخاب است. گزینه New Worksheet باعث اجرای نمودار در یک Sheet(برگه) جدید می شود و گزینه Existing Worksheet نمودار را در یک Sheet موجود اجرا می کنند. انتخاب گزینه Existing Worksheet باعث فعال شدن فیلد Location می شود. این فیلد kdcدر انتهای خود دارای کلیدی است که انتخاب یک Sheet را میسر می سازد. پس از انتخاب گزینه های مورد نظر کافی است برای اجرای نمودار PivotChart روی کلید OK کلیک شود.

PivotChart شامل یک بخش مرکزی است که نمودار در آن قرار می گیرد و یک پنل کناری که شامل فیلدها می باشد(شکل 9-11).

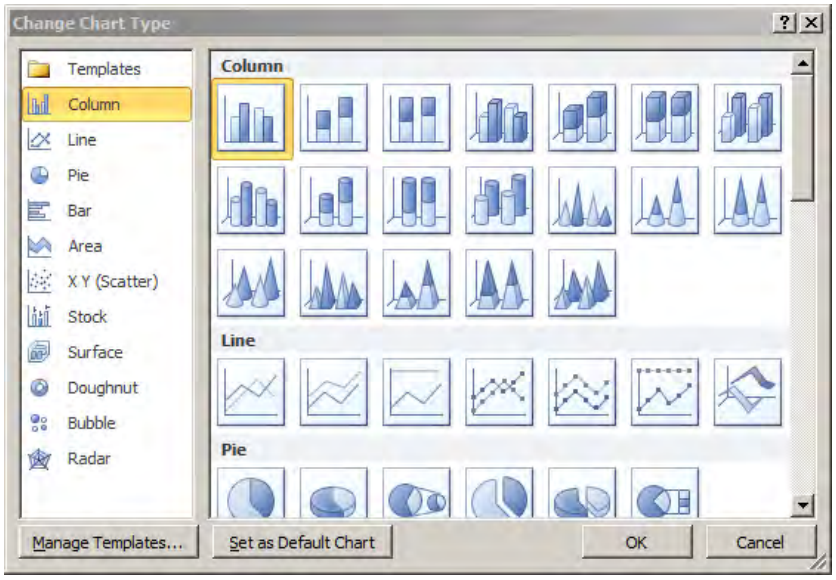


شکل 9-11: ایجاد یک گزارش به وسیله PivotChart

با درگ کردن (کشیدن) فیلدها در بخش های پایینی نمودار ایجاد می شود. چهار بخش برای ساختن نمودار در پنل کناری وجود دارد. هر کدام از این چهار بخش قادر هستند تا چند فیلد را درون خود جای دهند. هر فیلدی که به یکی از بخش ها تخصیص می یابد در لیست فوقانی علامت می خورد. برخی فیلد ها دو یا چند بار در لیست تکرار می شوند به این دلیل که آنها را بتوان در چند بخش به طور همزمان استفاده نمود.

بخش Axis Fields(Categories) مربوط به دسته بندی محور افقی می باشد. به عبارت دیگر محور افقی بر اساس فیلدی که در بخش Axis Fields(Categories) موجود است شکل می گیرد. بخش Values واحد محور عمودی را تعیین می کند. فیلدهای بخش Values مشخص می کنند که نمودار بر حسب چه مقادیری باید محاسبه گردد. این فیلد با برحسب مجموع مقدار آیتم ها و تعداد آیتم ها کار می کند. بخش Legend Fields(Series) مربوط به شرح نمودار است. فیلدهای درون آن چگونگی رسم ستون ها و محتوای نمودار را تعیین می کنند. بخش Report Filter نیز برای اعمال فیلترهای اضافی بر گزارش مورد استفاده قرار می گیرد.

نمودارهای ستونی تنها نوع نمودارهای قابل پشتیبانی در Excel نمی باشند. نرم افزار Excel از نمودارهای بسیار زیادی پشتیبانی می کند. این نمودار اغلب برای ساختن گزارشات سفارشی مورد نظر ما قابل استفاده هستند. برای تغییر نوع نمودار باید از منوی(تب) Design بر روی گزینه Change Chart Type (گزینه اول) کلیک شود. پیروی این عمل پنجره Change Chart Type گشوده خواهد شد(شکل 10-11).



شکل 10-11: پنجره تغییر نوع چارت

انواع نمودارهای مختلف از قبیل دایره ای، مخروطی، خطی و ... در این پنجره بر حسب نوع دسته بندی شده اند. با انتخاب هر کدام، نمودار جاری به نوع انتخابی تبدیل می شود. توجه شود که فیلد انتخابی برای یک نوع نمودار ممکن است برای دیگر نمودارها قابل استفاده نباشد.

ضمیمه 1: بیانیه آجیل

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.