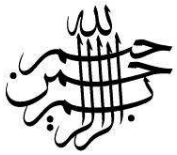
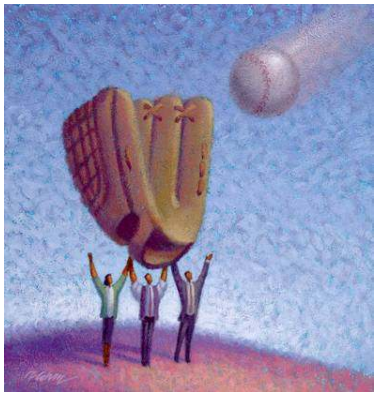


فرآیند عمومی متدولوژی‌های چابک

نعیم اصفهانی
esfahani A@T ce.sharif.edu

چکیده

متدولوژی‌های چابک به مجموعه‌ای از متدولوژی‌ها می‌گویند که سعی می‌کنند انعطاف پذیر باشند تا پیش‌بینی کننده. این متدولوژی‌ها را اصولا با متدولوژی XP می‌شناسند؛ حال آن‌که این متدولوژی تنها یکی از چندین متدولوژی مطرح در این زمینه است.



بر خلاف بحث اولیه‌ای که برخی از این متدولوژی‌ها در مورد عدم وجود فرآیند در متدولوژی می‌کردند بعد از مدتی شاهد بودیم که فرآیند البته با نام‌های دیگر در مورد این متدولوژی‌ها هم مطرح شد. فرآیندهای این متدولوژی‌ها شباهت‌ها و تفاوت‌هایی با هم دارد. امبلر یک چرخه‌ی حیات برای این نوع از متدولوژی‌ها[1] ارائه داده است که در آن سعی کرده تمام این متدولوژی‌ها را بپوشاند. در این نوشتار ما به بررسی هفت متدولوژی چابک می‌پردازیم، انطباق آن‌ها با این چرخه را بررسی می‌کنیم و چرخه را اصلاح می‌کنیم و گسترش می‌دهیم تا این متدولوژی‌ها را بپوشاند. در نهایت نیز سعی می‌کنیم شباهت‌های این متدولوژی‌ها را در قالب الگوهای فرآیند ارائه کنیم.

[1] Agile System Development Lifecycle

فهرست

فهرست جدول‌ها

فهرست شکل‌ها

مقدمه

متدولوژی‌های چابک متعدد هستند ولی در این‌جا به بررسی زیرمجموعه‌ای از این متدولوژی‌ها که در طی درس معرفی شده‌اند می‌پردازیم. این متدولوژی‌ها عبارتند از DSDM، اسکرام[۱]، XP، ASD، dX، کریستال[۲] و FDD. در معرفی این متدولوژی‌ها در این نوشتار بیش‌تر سعی می‌شود به معرفی فرآیند آن‌ها پرداخته شود؛ از بیان جزئیات نیز پرهیز می‌کنیم. این فرآیندها را تنها بیان می‌کنیم و قصد بررسی مزایا و معایب و مقایسه‌ی آن‌ها را نداریم. شایان ذکر است مطالب فصل ۱ و فصل ۲ به ترتیب خلاصه‌هایی از منابع ۱ و ۲ هستند، بنابراین از تکرار ارجاع به مراجع در این فصل‌ها خودداری کرده‌ایم.

برای برهم نهی این متدولوژی‌ها ابتدا باید فعالیت‌هایی که انجام می‌دهند را به دقت بررسی کنیم؛ به این منظور خلاصه‌ای از کارهای انجام شده در هر متدولوژی را در بخش یک ارائه می‌کنیم. برای آشنایی با چرخه‌ی حیات متدولوژی‌های چابک در بخش دو خلاصه‌ای از آن را نیز ارائه می‌دهیم. در بخش سوم به انطباق متدولوژی‌ها وچرخه‌ی بیان شده می‌پردازیم و آن را اصلاح می‌کنیم. در نهایت در بخش چهارم به ارائه‌ی الگوهای فرآیند برای این متدولوژی‌ها می‌پردازیم. در این نوشتار کلمات مرحله، فاز، قسمت، تکرار به یک معنا در نظر گرفته شده اند؛ مواردی که مفهوم این کلمات تفاوت دارند از خود متن قابل تشخیص است.

۱. متدولوژی‌های چابک

متدولوژی‌های چابک[۳] دسته‌ای از متدولوژی‌ها هستند که خصوصیات مشترکی دارند. این متدولوژی‌ها در سه فعالیت طراحی برنامه[۴]، برنامه‌نویسی و آزمون قدرت بیش‌تری را دارا هستند. فرآیند آن‌ها فرآیندی سبک و تکراری افزایشی[۵] است. پایه‌ی این متدولوژی‌ها در مفاهیمی است که در متدولوژی‌های مارپیچی[۶]، تکاملی[۷]، بر مبنای پروتوتایپ[۸] و شیءگرا وجود دارد. این متدولوژی‌ها سعی می‌کنند انعطاف‌پذیر[۹] باشند و تغییرات را بپذیرند بر خلاف برخی متدولوژی‌ها که پیش‌بینی کننده[۱۰] هستند.

[1] Scrum
[2] Crystal
[3] Agile
[4] Program Design
[5] Iterative Incremental
[6] Spiral
[7] Evolutionary
[8] Prototyping
[9] Flexible
[10] Predictive

در این نوع متدولوژی‌ها مدل اهمیت کمی پیدا کرده و به جای آن سعی می‌شود از سیستم قابل اجرا و قابل استفاده به عنوان یک مدل ملموس‌تر[1] استفاده کنند. با استفاده از تکامل تدریجی سیستم را به تدریج کامل می‌کنند تا تمام نیازمندی‌ها را بپوشاند. در این متدولوژی‌ها بر مشارکت فعال کاربران حساب باز می‌شود و سعی می‌شود به افرادی از تیم که فعال هستند میدان داده شود. در این متدولوژی‌ها به افراد اهمیت بیش‌تری داده می‌شود و فرض می‌شود که ارتباطات انسانی برای انتقال اطلاعات سیستم به کل تیم کافی هستند. متدولوژی‌ها چابک سعی می‌کنند همیشه روند یکنواخت پیشرفت را حفظ کنند و سرعت یکنواختی در تولید سیستم داشته باشند. طراحی برنامه در این دسته از متدولوژی‌ها باید در نهایت تکنیک باشد و در عین حال سادگی در آن لحاظ شود؛ لزومی ندارد همه‌ی پیچیدگی‌ها پیش‌بینی شود و کارهایی انجام شود که ممکن است هیچ وقت استفاده نشوند؛ با حداقل‌های ممکن شروع می‌کنیم و آن را با توجه به نیاز اصلاح می‌کنیم. جلسات بررسی، اصلاح و بهبود روند کار در این متدولوژی‌ها وجود دارد و حتی خود متدولوژی و فرآیند را می‌توان بررسی کرد و بهبود داد.

DSDM ۱ - ۱.

این متدولوژی قبل از معرفی متدولوژی‌ها چابک ارائه شد ولی در عمل شباهت زیادی با این متدولوژی‌ها دارد و استانداردی برای پروژه‌های RAD است. نمودارهای گرافیکی محبوبیت زیادی ندارند و همانند بقیه‌ی متدولوژی‌های چابک در این‌جا نیز مدل‌گریز هستیم. عملیات به صورت تکراری افزایشی انجام می‌شود و پروتوتایپ‌های ایجاد شده به تدریج کامل می‌شوند و دور ریخته نمی‌شوند. تیم‌های موجود نیز می‌توانند به صورت موازی و مستقل از هم به انجام کارهای خود بپردازند.

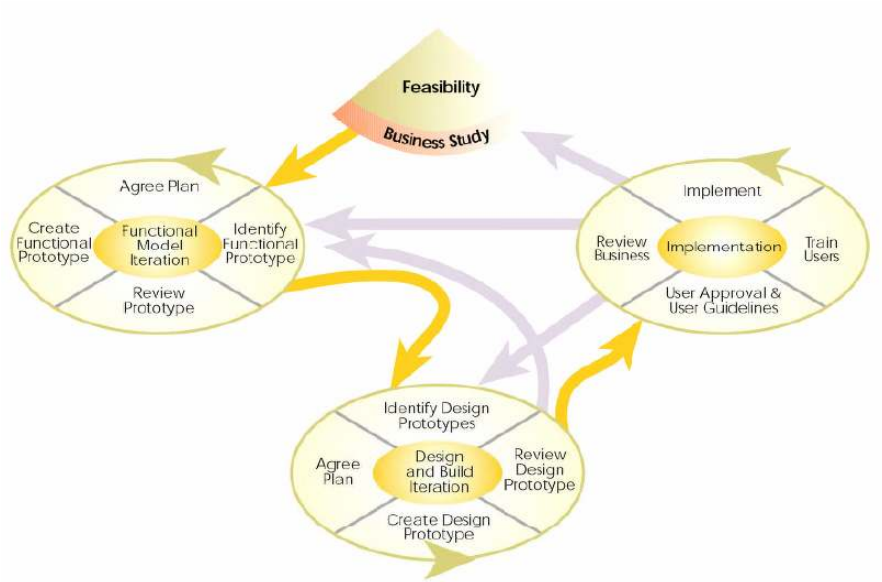
پروژه به سه قسمت قبل پروژه[2]، خود پروژه[3] و پس از پروژه[4] تقسیم می‌شود. در قبل پروژه به تامین منابع و برنامه‌ریزی اولیه می‌پردازیم. خود پروژه دو مجموعه فاز دارد؛ دو فاز امکان سنجی[5] و بررسی کسب و کار[6] که پشت سرهم هستند و سه فاز مدل‌کردن عملیات[7]، طراحی و ساخت[8] و پیاده‌سازی[9] که به صورت تکراری سعی در تکمیل پروتوتایپ‌ها دارند. هر بار شروع تا اتمام این فازهای تکراری از دو هفته تا دو ماه طول می‌کشند. در انتها نیز در قسمت پس از پروژه به نگهداری سیستم با تکرار فاز اصلی می‌پردازیم. در شکل ۱ قسمت خود پروژه از این فرآیند را مشاهده می‌کنیم.

در فاز امکان سنجی برنامه‌ی اولیه‌ی فازهای بعدی را می‌دهیم و به وسیله‌ی فیلترهای تناسب، امکان استفاده از DSDM در پروژه‌ی مورد نظر را بررسی می‌کنیم. فیلترهای تناسب به این قرارند:

---

[1] Tangible
[2] Pre-Project
[3] Project-Proper
[4] Post-Project
[5] Feasibility Study
[6] Business Study
[7] Functional Model Iteration
[8] Design and Build Iteration
[9] Implementation

- از واسط بتوان نیازمندی‌ها را استخراج کرد.
- کاربری که دقیقا مشخص باشد داریم.
- نباید خیلی پیچیده باشد: بر مبنای کسب و کار[1] باشد نه بر مبنای پردازش[2].
- نیازمندی‌ها نباید پیچیده باشند: بتوان جدا برنامه‌ریزی و پیاده‌سازی شوند.
- نیازی به استخراج کل نیازمندی‌ها قبل برنامه‌نویسی نباشد.
- اگر سیستم بزرگ است بتوان آن را قسمت‌بندی کرد.
- درک DSDM در سازمان وجود داشته باشد.



شکل ۱. فرآیند متدولوژی DSDM [1]

در فاز بررسی کسب و کار به تعریف و اولویت بندی نیازمندی‌های سطح بالا پرداخته، معماری سیستم و برنامه‌ی تولید آن را به صورت محدوده‌های زمانی[3] ایجاد می‌کنیم. باید توجه کرد که این متدولوژی شدیدا منعطف است و می‌توان فرآیند انجام کار را برای پروژه‌ای که در دست است عوض کرد. نیازمندی‌ها نیز قابل جا به جا کردن هستند اما نمی‌توان در محدوده‌های زمانی و منابع دست برد. همان‌طور که می‌دانیم باید از سه بعد تولید سیستم (زمان، نیازمندی‌ها و منابع) به یکی درجه‌ی آزادی دهیم و در این‌جا به نیازمندی‌ها و جا به جا شدن آن‌ها این آزادی اعطا شده است. در این فاز اولویت بندی میان نیازمندی‌هایی که باید پیاده سازی شوند انجام می‌شود.

[1] Business-Oriented
[2] Process-Oriented
[3] Time-Box

فاز مدل کردن عملیات به ریز کردن برخی از نیازمندی‌های سطح بالا با استفاده از پروتایپ‌ها می‌پردازد. عملا با این کار یک پوسته می‌سازیم که عملیات سیستم را نشان می‌دهد ولی درون آن چیزی نیست. بدیهی است که نیازمندی‌های غیروظیفه‌ای در این‌جا مورد توجه واقع نشده‌اند. همان طوری که دیده می‌شود مدل‌های ما همین پروتوتایپ‌ها هستند که نگهداری می‌شوند و در اثر بلوغ به صورت سیستم نهایی ارائه می‌شوند؛ این‌گونه بودن نهایت ملموس بودن مدل‌ها را به بار می‌آورد. در این مرحله و در صورت نیاز می‌توان از مدل‌های ساختاری مانند نمودار کلاس‌ها (انتصاب عملیات و صفات به کلاس‌ها) برای مدل کردن این جنبه‌ی دامنه استفاده کرد. این مدل‌های ساختاری نیز به تدریج کامل می‌شوند.

فاز بعدی که طراحی و ساخت است به کامل کردن و پر کردن پوسته‌ی ساخته شده در مرحله‌ی قبل می‌پردازد تا بتوان از آن محصول را به دست آورد. در این مرحله آزمون مستمر[1] داریم و نیازمندی‌های غیر وظیفه‌ای مورد توجه قرار می‌گیرند. پروتوتایپ‌های میانی نیز به عنوان مستندات نگهداری می‌شوند.

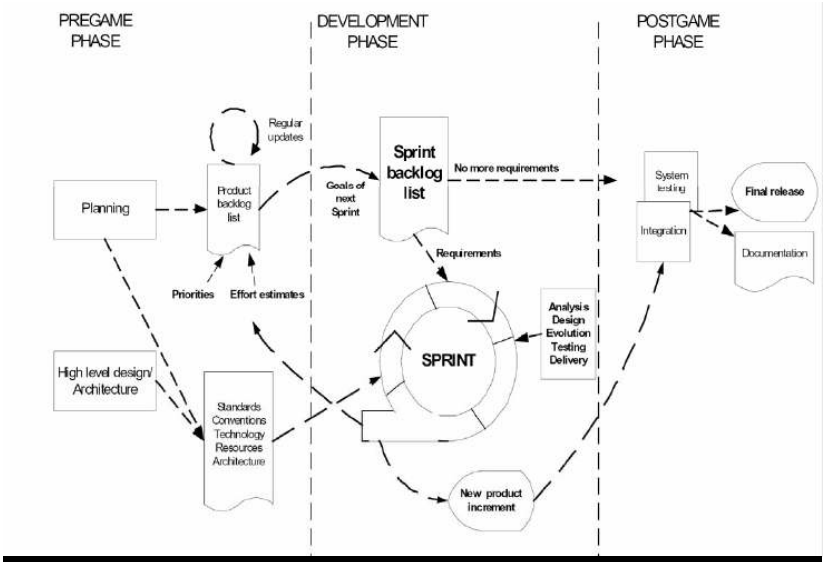
هدف از فاز پیاده‌سازی ارائه‌ی کارهای انجام شده به صورت یک افزودنی[2] در محیط واقعی و اعتبارسنجی آن‌هاست. با ورود افزودنی به سیستم در حال کار اعتبار سنجی[3] مستمر داریم. در انتهای این فاز بررسی می‌کنیم که اوضاع از چه قرار است و چه کاری باید انجام شود. آیا نیازمندی‌ها تمام شده‌اند یا اصولا نیازمندی جدیدی کشف شده است. پس از بررسی‌های لازم تصمیم می‌گیریم که پس از این فاز چگونه عمل کنیم. آیا یک چرخه‌ی دیگر را شروع کنیم (پرش به فازهای پشت سر هم) یا در همین مرحله و به فازهای دیگر (فازهای تکراری) بپریم.

۱ - ۲. اسکرام

اسکرام اساسا یک روش مدیریت پروژه است و وابستگی شدیدی به انسان‌ها، توانایی برنامه‌ریزی آن‌ها و ارتباط برقرار کردن آن‌ها دارد. در این متدولوژی سه مرحله‌ی تولید سیستم داریم (شکل ۲): قبلِ بازی[4]، تولید (بازی)[5] و بَعدِ بازی[6]؛ مرحله‌ی قبل بازی خود شامل برنامه‌ریزی[7] و معماری و طراحی سطح بالا[8] می‌شود. مرحله‌ی تولید نیز سه عمل تکراری برنامه‌ریزی اسپرینت[9]، تولید اسپرینت[10] و مرور اسپرینت[11] را در بر می‌گیرد. اسپرینت قسمت تکراری فرآیند را تشکیل می‌دهد و بازه‌ی زمانی آن یک ماه می‌باشد. کار تولید سیستم در اسپرینت‌ها انجام می‌شود.

[1] Continuous Test
[2] Increment
[3] Continuous Validation
[4] Pre-Game
[5] Development (Game)
[6] Post-Game
[7] Planning
[8] Architecture/High-level Design
[9] Sprint Planning
[10] Sprint Development
[11] Sprint Review

در برنامه‌ریزی لیست نیازمندی‌ها به نام بک‌لاگ محصول[1] توسط کاربر تولید می‌شود. مسئول بک‌لاگ خود کاربر است و در طول زمان پروژه اوست که آن را نگهداری می‌کند. اولویت بندی نیازمندی‌ها در این مرحله انجام می‌شود. این اولویت بندی بر اساس ریسک که همان ارزش نیازمندی برای کاربر است انجام می‌دهند. در این مرحله تامین منابع انجام می‌شود و تیم‌هایی شامل اعضایی با توانایی‌های متفاوت تشکیل می‌شوند. در نهایت نیز زمان‌بندی ترخیص‌های[2] پروژه تعیین می‌شود. مفهومی به نام امکان سنجی وجود ندارد و باید خود تیم این مساله را تشخیص دهد. در هر تیم یک استادِ اسکرام![3] وجود دارد تا کارها و اجرای اسکرام را تسهیل کند.



شکل ۲. فرآیند متدولوژی اسکرام [1]

مرحله‌ی بعد مرحله‌ای است که در آن بر پایه‌ی نیازمندی‌ها (بک‌لاگ محصول) به ایجاد طراحی سطح بالای سیستم مبادرت می‌ورزیم. برای فهم بهتر سیستم و نیازمندی‌ها می‌توان از پروتوتایپ‌ها نیز استفاده کرد. البته این پروتوتایپ‌ها بعدا دور ریخته می‌شوند. در طول زمان ممکن است تغییراتی در بک‌لاگ محصول ایجاد شوند که توسط کاربر این عمل به روز رسانی انجام می‌شود. در اسکرام طراحی مشترک که مرجع همگان شود نداریم، بنابراین برای هماهنگی از اسکرام‌هایی از اسکرام‌ها[4] استفاده می‌شود تا هماهنگی لازم به گونه‌ای تزریق شود. برای جلوگیری از برهم نهی کارها نیز سعی می‌کنند تا حد امکان بک‌لاگ را ریز کنند تا با مشخص شدن دقیق کارها و رفع ابهام بین تیم‌ها، کار مشابه انجام نشود.

در برنامه‌ریزی اسپرینت جلسه‌ای تشکیل می‌شود که تمام اعضای تیم در آن هستند. در این جلسه هدف اسپرینت بر اساس اجزای بک‌لاگ محصول تعیین می‌شود و یک بک‌لاگ برای اسپرینت ایجاد می‌شود. این

[1] Product Backlog
[2] Release
[3] Scrum Master
[4] Scrums of Scrums

بک‌لاگ زیرمجموعه‌ای از بک‌لاگ محصول است که قرار است در اسپرینت به آن پرداخته شود و به اندازه‌ای است که در یک ماه بتوان آن را به صورت یک افزودنی معرفی کرد. این بک‌لاگ در آینده جزئی‌تر می‌شود و برای پیاده‌سازی مناسب می‌شود.

در تولید اسپرینت هر روز جلساتی تشکیل می‌شود و در آن به بررسی مسائلی که از جلسه‌ی پیش تا کنون اتفاق افتاده و مشکلات پیش آمده می‌پردازند. سپس برنامه‌ای که تا جلسه‌ی آینده قرار است پیش گرفته شود و کارهایی که انجام خواهد شد ارائه می‌شود. به فراخور نیاز مدیریت هم در این جلسات و به دعوت استاد اسکرام شرکت می‌کند تا از مشکلات تیم مطلع شده و بتواند آن‌ها را رفع کند.

در مرور اسپرینت افزودنی ایجاد شده به مشتری تحویل داده می‌شود. در این مرحله کار انجام شده را به زیرمجموعه‌ی مهمی از مشتریان نمایش می‌دهیم و تطابق آن با هدف اسپرینت را بررسی می‌کنیم. بر اساس نتایج به‌دست آمده بک‌لاگ محصول به روز می‌شود و نیازمندی‌ها اضافه و کم می‌شوند. در نهایت به بررسی و تحلیل روند پیشرفت کار می‌پردازیم تا در صورت وجود مشکل برنامه‌ریزی‌های بعدی را تغییر دهیم.

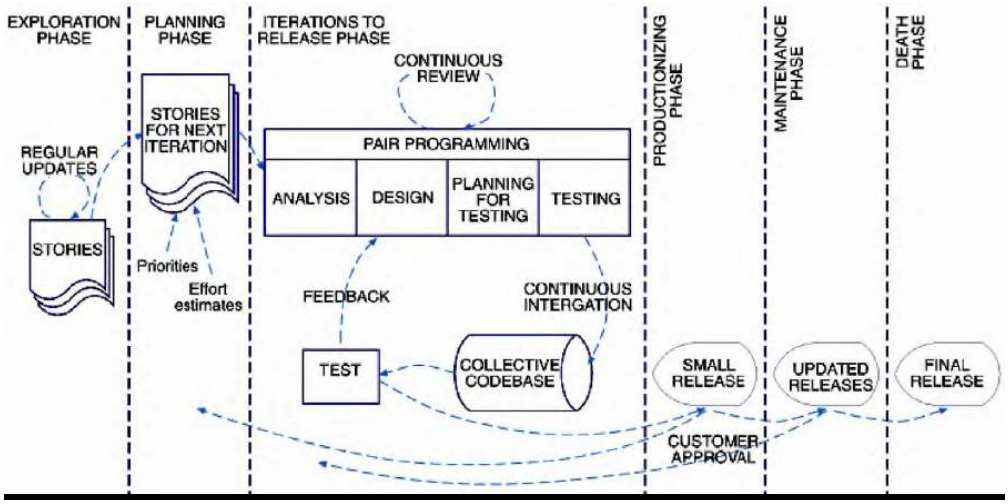
آخرین مرحله در اسکرام بعد از بازی است. در این مرحله افزودنی‌های ایجاد شده با هم ادغام شده و سپس تست سطح سیستم[۱] انجام می‌شود. پس از اطمینان از درست کار کردن سیستم به تهیه‌ی مستندات، آموزش کاربران و تبدیلات لازم[۲] برای استقرار سیستم مبادرت ورزیده می‌شود. در نهایت نیز آزمون مقبولیت[۳] از کاربران به عمل می‌آید.

۱ - ۳. XP

متدولوژی XP که پدر این نوع از متودلوژی‌هاست، معروف‌ترین متدولوژی چابک است. با معرفی آن و قوانینش بود که مفهوم چابک بودن پا گرفت و حتی متدولوژی‌های قدیمی‌تر مانند DSDM که شباهت‌هایی با آن‌ها داشتند را چابک نامیدند. این متدولوژی اصولا علاقه‌ای به تخمین و زمان‌بندی ندارد. این متدولوژی دارای ۶ فاز (شکل ۳) است: اکتشاف[۴]، برنامه ریزی (برای ترخیص)[۵]، تکرارها برای اولین ترخیص[۶]، محصول سازی[۷]، نگهداری[۸] و مرگ[۹]. در این متدولوژی اصالت با انجام کار است تا رعایت محدوده‌ی زمانی (اسکرام و DSDM)، یعنی سعی می‌شود که یک واحد کاری کامل شود و زمان گسترش می‌یابد در حالی که در برخی متدولوژی‌ها بعضی کارها را حذف می‌کردند و به آینده موکول می‌کردند تا محدوده‌ی زمانی تعیین شده را رعایت کنند. مدت زمان هر تکرار در این متدولوژی بین ۱ تا ۲ هفته است.

---

[1] System-wide Test
[2] System Conversion/Packaging
[3] Acceptance Test
[4] Exploration
[5] Planning (Release Planning)
[6] Iterations to first Release
[7] Product ionizing
[8] Maintenance
[9] Death



شکل ۳. فرآیند متدولوژی XP [1]

در فاز اکتشاف به تامین منابع و درک نیازمندی‌های سطح بالا پرداخته می‌شود. در این مرحله تیم تشکیل می‌شود. در این تیم نماینده‌ی کاربر نیز حضور دارد. اصولا در این متدولوژی علاقه‌ای به چند تیمی بودن وجود ندارد. نیازمندی‌های سیستم از دید کاربر و به صورت تکه‌ای وظیفه‌مندی در قالب داستان‌های کاربر[1] ارائه می‌شوند. در نهایت هم معماری سیستم به نام متافور[2] ایجاد می‌شود که در سطح بالا می‌گوید سیستم چه می‌کند. این معماری در این مرحله باید برای کاربر خوانا باشد. معماری در آینده کامل‌تر می‌شود و جزئیاتی که ممکن است کاربر نفهمد نیز به آن اضافه می‌شود تا برای پیاده‌سازی مناسب گردد. برای درک بهتر از سیستم از پروتوتایپ‌های دور ریختنی به نام اسپایک[3] استفاده می‌شود.

برای تخمین زمان کل تولید و اولویت بندی کارها فاز برنامه‌ریزی را داریم. در این فاز تخمین زمانی برای هر داستان کاربر که بر روی کارتی[4] نگاشته شده انجام می‌دهیم. این کار توسط برنامه‌نویسان انجام می‌شود و کاری که تخمین انجام دادنش بیش از ۳ هفته باشد به کارهای کوچک‌تر شکسته می‌شود. در این‌جا برای دقیق‌تر کردن تخمین‌ها می‌توان از اسپایک‌ها نیز استفاده کرد. پس از تخمین زمانی، نیازمندی‌ها که در قالب داستان‌های کاربر هستند، اولویت بندی می‌شوند و این کار توسط خود کاربر انجام می‌شود؛ فرض می‌شود که کاربر بهترین مرجع برای تشخیص ریسک است و هرچه برای او مهم‌تر است ریسک بیش‌تری دارد. در نهایت نیز تاریخی برای ارائه‌ی اولین ترخیص تعیین می‌شود. در اولین ترخیص حداقل نیازمندی‌های مهم از دید کاربر پیاده‌سازی می‌شوند. در این مرحله زمان یک تکرار (بین ۱ تا ۳ هفته) مشخص می‌شود و تا آخر پروژه مدت یک تکرار ثابت می‌ماند.

[1] User Story
[2] Metaphor
[3] Spike
[4] User Story Index Card

در فاز بعدی که تکرارها برای ترخیص اول است، ابتدا و در شروع هر تکرار جلسه‌ای برای برنامه‌ریزی آن تکرار برگزار می‌شود. در این جلسه کارهایی که باید در تکرار انجام شوند شامل داستان‌های کاربر و آزمون‌های مشکل پیدا کرده از مرحله‌ی قبل معرفی می‌شوند و سعی در رفع آن‌ها داریم. از تجربه‌ی مراحل قبل استفاده می‌شود که برنامه‌ریزی‌ها بهتر انجام شود. قابل توجه این‌که کارهایی که باید انجام شوند در سطح سیستم مشخص می‌شوند و در سطح کلاس ریز نمی‌شوند. تخصیص کارها هم به این صورت است که پیاده‌سازها خود اقدام به دریافت تکه‌های کار می‌کنند. در ادامه تولید در قالب تکرار انجام می‌شود. در ابتدای هر روز از تکرار جلسات ایستاده[۱] برگزار می‌شود. در انجام تکرار و به صورت روزمره عملیات تحلیل، طراحی، برنامه نویسی و آزمون انجام می‌شود. مفاهیمی مانند آزمون مستمر و مالکیت عمومی کد[۲] در این مرحله اعمال می‌شوند.

پس از ارائه‌ی هر قسمت از سیستم (که از یک تکرار بیرون می‌آید)، اقدام به وارد کردن آن به محیط کاربر می‌کنیم. به این ترتیب که تکه‌ی ایجاد شده به کل سیستم اضافه می‌شود و بررسی اعتبار و درستی سیستم[۳] انجام می‌شود. در این مرحله مطمئن می‌شویم که سیستم مورد قبول کاربر است. ممکن است برای وارد کردن محصول در محیط کاربر نیاز به انجام تکرار با مدت زمان یک هفته باشد. در این تکرار مشکلات مربوط به وارد کردن سیستم به محیط کاربر رفع می‌شود و حتی ممکن است کد نیز زده شود.

پس از ارائه‌ی اولین ترخیص وارد فاز نگهداری می‌شویم. دید فرآیند به این ترتیب است که اصولا هرگونه تغییر و افزایش در سیستم پس از اولین ترخیص چه برای غنی‌تر کردن آن نسبت به ترخیص اول و چه بهبود کارکرد آن و رفع نقص به عنوان عمل نگهداری انجام می‌شود. برای انجام این‌کار موتور ایجاد[۴] که شامل سه مرحله‌ی قبلی (برنامه‌ریزی، تکرارها برای تولید و محصول‌سازی) است دوباره به راه می‌افتد. این کار تا اتمام تمام داستان‌های کاربر و عدم امکان تکامل سیستم (مرگ سیستم) ادامه می‌یابد.

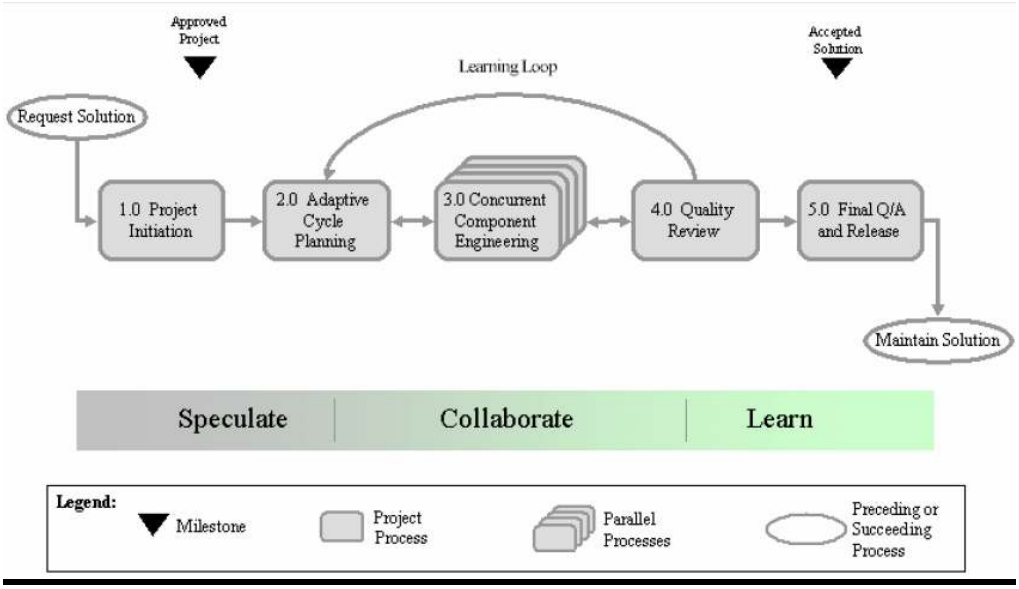
مرگ سیستم وقتی رخ می‌دهد که یا تکامل سیستم لازم نیست و یا اصولا امکان‌پذیر نیست. در این مرحله پروژه بسته می‌شود و بررسی‌ها و مستندسازی‌های پس از مرگ سیستم[۵] انجام می‌شود.

۱ - ۴. ASD

ASD با تاثیر از DSDM سعی می‌کند تغییر را بپذیرد. عملیات به صورت حدس-همکاری-یادگیری[۶] در این متدولوژی انجام می‌شود. این روند در مورد همه چیز حتی خود فرآیند نیز استفاده می‌شود. این روش به جای نمونه‌های مشهورتر برنامه‌ریزی-طراحی-ساخت[۷] و یا فرآیند تکراری برنامه‌ریزی-ساخت-اصلاح[۸] استفاده می‌شود. فرض ASD بر این است که تمام جنبه‌های تولید سیستم شدیدا تغییر پذیرند؛ فرآیند تولید نرم‌افزار که فرآیندی

[1] Standup Meeting
[2] Collective Code Ownership
[3] System-wide Verification and Validation
[4] Development Engine
[5] Post-mortem Activities
[6] Speculate-Collaborate-Learn
[7] Plan-Design-Build
[8] Plan-Build-Revise

شدیدا پیچیده است تنها در صورت تسهیل همکاری انسان‌ها ممکن می‌شود؛ عدم قطعیتی که در این وضعیت به وجود می‌آید تنها با کوتاه گرفتن زمان چرخه‌ها ممکن است. عملا در یک چرخه‌ی ۱ تا ۸ هفته‌ای تمرکز خود را در زیرمجموعه‌ای از سیستم حفظ می‌شود و تغییرات افراد تیم را دچار مشکل نمی‌کنند. برنامه‌ها در این وضعیت تخمین‌هایی هستند که بعد از هر چرخه بازبینی می‌شوند و ممکن است تغییر کنند. در این متدولوژی ممکن است به دلیل عدم‌قطعیتی که بحث شد مجبور شویم وسط یک چرخه و یک کار همه‌چیز را رها کنیم و به چیز دیگری بپردازیم. در شکل ۴ فرآیند کلی این متدولوژی را مشاهده می‌کنیم.



شکل ۴. فرآیند متدولوژی ASD [1]

سه فاز اساسی فرآیند ASD را تشکیل می‌دهند: شروع پروژه[۱]، تولید تکراری[۲] و تضمین کیفیت نهایی و ترخیص[۳]. تولید تکراری خود شامل سه مرحله‌ی برنامه‌ریزی منعطف چرخه[۴] (حدس)، مهندسی هم‌زمان اجزا[۵] (ساخت) و مرور کیفیت[۶] (یادگیری) است. شایان ذکر است که این متدولوژی فازی تحت عنوان نگهداری ندارد.

در شروع پروژه به تعیین اهداف پروژه و معیار موفقیت آن می‌پردازیم و به همراه کاربران به تهیه‌ی مصنوع[۷] مشخصات خواسته‌ها می‌پردازیم؛ این مستند شامل دید کلی[۸] از کاری است که قرار است انجام شود و کارهایی که باید انجام شوند به همراه محدوده‌ی آن‌ها مشخص می‌شود. عملا کار تحلیل در این فاز انجام می‌شود و در نهایت

[1] Project Initiation
[2] Iterative Development
[3] Final Q/A and Release
[4] Adaptive Cycle Planning
[5] Concurrent Component Engineering
[6] Quality Review
[7] Artifact
[8] Vision/Charter

یک مستند تک صفحه‌ای به نام صفحه‌ی اطلاعات پروژه[1] تهیه می‌شود که تمام اطلاعات را در خود دارد. در نهایت این مرحله ما به کسب تایید از کاربر و (یا) پشتیبان[2] اقدام می‌کنیم و ارزش‌های موجود در پروژه را از طریق بحث در مورد اهداف کیفی و معیارهای ارزیابی به اشتراک می‌گذاریم.

در برنامه‌ریزی منعطف چرخه به تعیین محدوده‌ی زمانی پروژه و هرکدام از چرخه‌ها می‌پردازیم، هدف چرخه‌ای که می‌خواهد شروع شود و مولفه‌های محصول که در آن تولید خواهند شد تعیین می‌شوند. مولفه‌های تعیین شده به سه دسته‌ی خصوصیات[3]، پشتیبانی[4] و تکنیکی[5] تقسیم می‌شوند. مولفه‌ی خصوصیات سعی دارند قسمتی از کاری که در حوزه‌ی مساله است را انجام دهند. مولفه‌ی پشتیبانی سعی می‌کند مسائلی که برای سرپا نگهداشتن سیستم لازم است را تولید کند (سیستم مدیریت نرم‌افزار) و مولفه‌ی تکنیکی مشکلات تکنولوژیک بر سر راه تولید را رفع می‌کند (لایه‌ی دسترسی به داده).

اختصاص این مولفه‌ها به چرخه بر اساس ریسک تولید آن‌هاست. برنامه‌ریزی پروژه به صورتی است که جای باز برای تطابق با تغییرات و مشکلات را دارد[6]. در این مرحله ابزارهایی که برای تسهیل ارتباط و همکاری تیم لازم هستند برپا می‌شوند و لیست کارهای پروژه به روز می‌شود. این لیست شامل کارها و مولفه‌هایی است که باقی مانده‌اند.

در این متدولوژی می‌توان تیم‌هایی داشت که به صورت موازی با هم کار می‌کنند. بنابراین فاز تولید این فرآیند را مهندسی همزمان اجرا نامیده‌اند. در این مرحله مولفه‌های تخصیص داده شده به چرخه به صورت ساخت‌های[7] روزانه و یا هفتگی تحویل داده می‌شوند. کنترل و مدیریت پروژه به صورت مستمر اقدام به هماهنگی و مراقبت از همکاری درون تیمی و بین تیمی می‌کند. در ادامه برای بررسی کیفیت و کنترل کیفیت نهایی آماده می‌شویم، نمونه آزمون‌ها تهیه می‌شوند، برنامه‌ی آزمون‌ها ریخته می‌شود و برای جلسات در آن فازها آماده می‌شویم.

در فاز مرور کیفیت به مرور چرخه‌ی انجام شده و ارائه‌ی نتایج به مشتری و اخذ نظرات او می‌پردازیم. بر اساس نتایج این فاز قدم بعدی را تعیین می‌کنیم که یا ادامه‌ی تولید است و یا ترخیص نهایی می‌باشد. در نهایت هم مرحله‌ی پس از مرگ چرخه را داریم که کارایی تیم و روش استفاده شده را مورد بررسی قرار داده و از آن یاد می‌گیریم و درس‌های آموخته شده را در تکرارهای بعدی مورد استفاده قرار می‌دهیم.

در تضمین کیفیت نهایی و ترخیص نهایی با اجرای آزمون‌ها به اعتبار سنجی سطح سیستم می‌پردازیم، نتایج آزمون‌ها را ارزیابی و مشکلات را اصلاح می‌کنیم. در این مرحله بر اساس نتیجه‌ی آزمون تصمیم می‌گیریم که تکرار چرخه‌ی تولید را داشته باشیم یا به سمت ترخیص رویم. در ترخیص، به انتقال سیستم به حالت کارکردن و

[1] Project Data Sheet
[2] Sponsor
[3] Feature Component
[4] Support Component
[5] Technical Component
[6] Buffered Schedule
[7] Builds

ملزومات مستقر سازی آن مانند آموزش، تبدیلات لازم و مستند سازی می‌پردازیم. در نهایت نیز پروژه بسته می‌شود و مراسم پس از مرگ پروژه برگزار می‌شود و درس‌های آموخته شده ثبت می‌شوند.

۱ - ۵. dX

dX سعی می‌کند RUP را به صورت چابک ارائه کند. مدل‌گریزی[1] برخلاف متدولوژی‌های چابک دیگر در این متدولوژی وجود ندارد ولی عملا به دلیل کوچک بودن زمان هر تکرار که کم‌تر از یک هفته است فرصت مدل‌سازی نداریم. اصولا نام این متدولوژی از همین کوچکی بیش از حد تکرارهایش الهام گرفته است (البته اگر چرخش XP را الهام حساب نکنیم!).

dX از چهار فاز تشکیل شده که همانند فازهای RUP هستند. این فازها عبارتند از آغاز[2]، جزئی‌شدن[3]، ساخت[4] و انتقال[5]. البته دو فاز جزئی‌شدن و ساخت عملا یک فاز هستند و تنها برای وفاداری به RUP مجزا شده‌اند. تکرارهای فاز جزئی‌شدن و ساخت خود مراحلی دارند: برنامه‌ریزی تکرار[6]، طراحی، برنامه‌نویسی[7]، بررسی پس از تکرار[8] و انتقال به ساخت[9].

در فاز آغاز نیازمندی‌های سطح بالا تعیین می‌شوند به این صورت که نماینده‌ی مشتری توضیحاتی ساده را بر روی کارت نمایه[10] می‌نویسد و بر اساس آن یک پروتوتایپ دور ریختنی ساخته می‌شود تا کارایی برنامه‌نویس و موارد کاربرد مشخص شوند. بر اساس نتیجه‌ی پروتوتایپ برنامه‌ریزی اولیه انجام می‌شود و یک روش پیشنهادی[11] معماری انتخاب می‌شود.

فاز جزئی‌شدن و ساخت همان‌طوری که گفته شد یکی هستند و رابطه‌ی آن‌ها به این صورت است که از یک مرحله به بعد عملا از فاز جزئی‌شدن وارد فاز ساخت می‌شویم. در قسمت جزئی‌تر شدن موارد کاربرد مشخص شده و آن‌هایی که ریسک بیش‌تری دارند پیاده‌سازی می‌شوند؛ این پیاده‌سازی البته به صورت تکراری انجام می‌شود. با حذف ریسک‌های بالاتر معماری و برنامه‌ریزی‌ها ثابت می‌شوند و با ایجاد برنامه‌ی ترخیص وارد مرحله‌ی ساخت می‌شویم. در این مرحله همان کارهای جزئی‌شدن انجام می‌شود با این تفاوت که بر اساس برنامه‌ی ترخیص عمل می‌کنیم و بقیه‌ی موارد کاربرد را می‌سازیم. در این فاز نماینده‌ی کاربر لیست موارد کاربرد را نگهداری می‌کند، میزان کار لازم برای هر مورد کاربرد بر روی کارت نمایه نوشته می‌شود، موارد کاربرد بر

[1] Model Phobia
[2] Inception
[3] Elaboration
[4] Construction
[5] Transition
[6] Iteration Planning
[7] Coding
[8] Post-iteration Revision
[9] Transition to Construction
[10] Index Card
[11] Alternative

اساس ریسک اولویت بندی می‌شوند. تولید، طراحی و پیاده‌سازی در تکرارهای کوچک انجام می‌پذیرند و یکپارچگی[1] مستمر را داریم.

در برنامه‌ریزی تکرار موارد کاربرد با اولویت بالا توسط مشتری به تکرار تخصیص داده می‌شوند. در طراحی موارد کاربرد در جلسات طراحی به گونه‌ای طراحی می‌شوند که به معماری بخورند. در برنامه‌نویسی کد نوشته می‌شود و از بسیاری از اصول XP تبعیت می‌شود. در بررسی پس از تکرار بر مبنای درس‌های آموخته شده، معماری و برنامه تغییر داده می‌شوند و در نهایت انتقال به ساخت به محض ثابت شدن برنامه‌ریزی و معماری اتفاق می‌افتد.

در فاز انتقال معرفی نرم‌افزار تولید شده به محیط کاربر انجام می‌شود. این ترخیص‌ها به صورت مکرر و موازی با ساخت انجام می‌شوند و هرچه زودتر باشند بهتر است. از آن‌جا که ترخیص‌های اولیه مشکلاتی دارند سیستم‌های قدیمی را موازی سیستم جدید نگهداری می‌کنیم. عملا این فاز هم در دل فاز ساخت و جزئی شدن است و تنها برای وفادار ماندن به RUP است که جدا شده است.

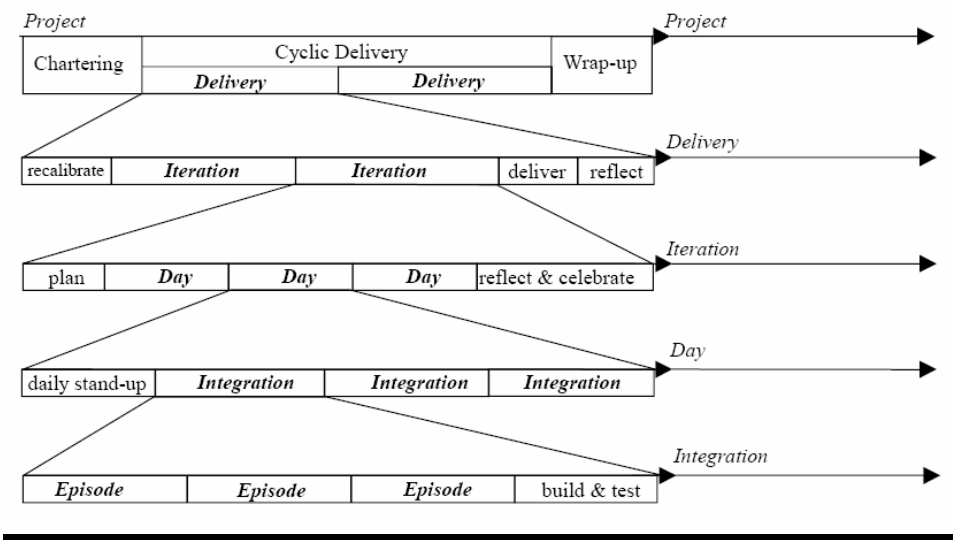
۱ - ۶. کریستال

کریستال در حقیقت ارائه دهنده‌ی خانواده‌ای از متدولوژی‌هاست. اعضای این خانواده با رنگ‌ها معرفی می‌شوند و رنگ تیره‌تر نشان‌دهنده‌ی متدولوژی پیچیده‌تر است. پروژه‌ها را بر اساس اندازه (تعداد افراد درگیر) و بحرانی بودن [2] طبقه‌بندی می‌کنند. این متدولوژی به ما اجازه می‌دهد در داخل آن از متدولوژی‌های دیگر نیز استفاده کنیم. ابتدا یک متدولوژی پایه انتخاب می‌شود و جلسات تامل[3] برای بررسی و بهبود فرآیند برگزار می‌شوند. فرض این متدولوژی این است که تولید کنندگان باید نزدیک به هم (در یک ساختمان) باشند و ارتباطات فراوانی داشته باشند. در این جا ما به بررسی متدولوژی کریستال شفاف به عنوان عضوی از خانواده می‌پردازیم.

فرآیند کریستال دارای سه مرحله‌ی برقرار کردن[4]، تحویل چرخه‌ای[5] و خاتمه دادن[6] است. تحویل چرخه‌ای نیز خود دارای مراحلی است: برنامه‌ریزی تکرار[7]، برنامه‌نویسی-آزمون-یکپارچگی چرخشی[8] و جشن اتمام تکرار[9]. در برنامه‌نویسی-آزمون-یکپارچگی چرخشی، چرخه‌های روزانه داریم که شامل یک جلسه‌ی ایستاده و چندین چرخه‌ی یکپارچه‌سازی است. در کریستال شفاف زمان تکرارها از یک هفته تا سه ماه می‌باشد. این‌که این فرآیند

[1] Integration
[2] Criticality
[3] Reflection
[4] Chartering
[5] Cyclic Delivery
[6] Wrap-up
[7] Iteration Planning
[8] Cyclic Program-Test-Integrate
[9] Iteration Completion Ritual

نگهداری ندارد و فرض بر این است که تغییرات مستلزم شروع مجدد چرخه‌ی ذکر شده هستند. در شکل ۵ نمای کلی فرآیند این متدولوژی را مشاهده می‌کنیم.



شکل ۵. فرآیند متدولوژی کریستال شفاف [1]

در فاز برقرار کردن به تشکیل تیم شامل مدیر اجرایی که متخصص دامنه[1] هم هست، سر طراح، نماینده‌ی کاربر[2] و اعضای تیم تولید (تحلیل‌گر، طراح-برنامه‌نویس، آزمون‌گر، متخصص دامنه، هماهنگ کننده، متن‌نویس و ...) می‌پردازیم. سپس به امکان سنجی اولیه و بررسی سطح بالای پروژه می‌پردازیم؛ ارزش تجاری سیستم، نیازمندی‌های سطح بالا، مدل دامنه[3]، انتخاب‌های تکنولوژی، برنامه‌ها، محدودیت‌ها، منابع و جزئیات متدولوژی همه مشخص می‌شوند. جزئیات متدولوژی همان‌طور که قبلا هم گفته شد در جلسات تامل برای بررسی و بهبود فرآیند تعیین می‌شوند و فرآیند را خاص پروژه تغییر می‌دهند؛ ولی در این ابتدا حداقل قوانینی که برای شروع پروژه توافق بر سر آن‌ها لازم است تعیین می‌شوند. در نهایت هم برنامه‌ی اولیه ایجاد می‌شود؛ نقشه‌ی پروژه بر اساس کارها و روابط نیازمندی‌ها به دست می‌آید و اولویت‌ها و کارها مشخص می‌شوند.

در تحویل چرخه‌ای که باید در دو چرخه‌ی تحویل یا بیش‌تر انجام شود چهار کار اساسی انجام می‌شوند: تنظیم دوباره‌ی برنامه‌ی ترخیص[4] بر اساس درس‌های گرفته شده را داریم. تولید توسط تکرارها[5] می‌باشد. تحویل محصول به کاربران حقیقی[6] که زیر مجموعه‌ی اندکی از کاربران هستند. این مجموعه‌ی اندک تاثیر بالایی بر

[1] Domain Expert
[2] Ambassador User
[3] Domain Model
[4] Recalibrate the Release Plan
[5] Develop in Iterations
[6] Deliver to Real Users

پذیرش سیستم دارند و در این مرحله نظرات آن‌ها را اعمال می‌کنیم. تامل بر محصول[1] از طریق بررسی متدولوژی و کیفیت محصول و برنامه‌ها صورت می‌گیرد. انجام تکرارهای تولید نیز برای خود مراحلی دارد که شامل سه مرحله‌ی اساسی است: برنامه‌ریزی تکرار که مشخص می‌کند چه کارهایی باید انجام شود، تکرار روزانه‌ی چرخه‌ی برنامه‌نویسی-آزمون-یکپارچگی، با برگزاری جلسات ایستاده و چندین چرخه‌ی یکپارچه سازی، در نهایت نیز جشن اتمام تکرار برگزار شده و جلسه‌ای برای بررسی درس‌های یاد گرفته شده در تکرار برگزار می‌شود. در چرخه‌های یکپارچه‌سازی دو مرحله داریم: ابتدا با طراحی و برنامه‌نویسی جزئی به سیستم اضافه می‌شود که مورد آزمون قرار گرفته سپس عملیات یکپارچه سازی با محصول و آزمون‌های یکپارچگی[2] انجام می‌شوند.

در نهایت خاتمه دادن سیستم با قراردادن آن در محیط کاربر و انجام آزمون‌های مقبولیت را داریم. در این مرحله محصول نهایی برای محیط کاربر آماده می‌شود، تبدیلات سیستم[3] انجام شده و دروس یاد گرفته شده ثبت می‌گردند.

FDD ۱ - ۷.

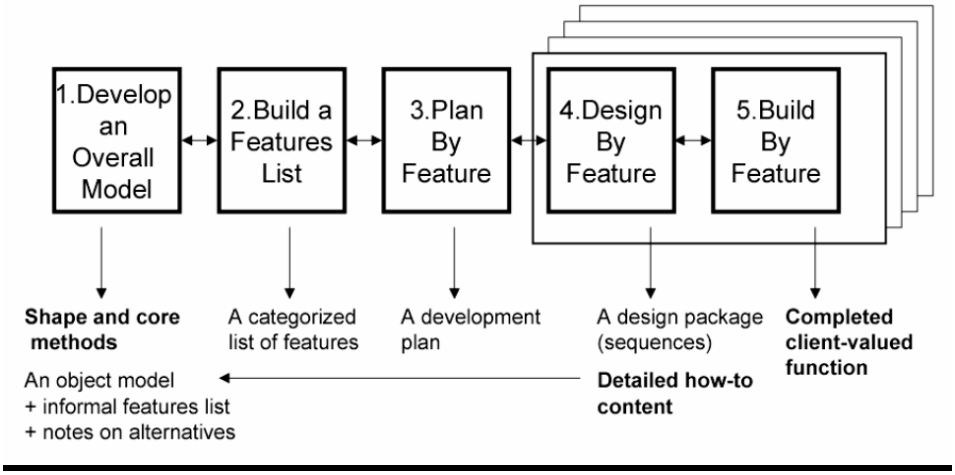
FDD به عنوان یک هسته‌ی قابل افزایش عمل می‌کند و فرآیند آن کل زمان پروژه را در بر نمی‌گیرد؛ در این فرآیند فعالیت‌های پس از پروژه و امکان‌سنجی‌های اولیه وجود ندارند. پایه‌ی همه‌ی کارها در این متدولوژی خصیصه‌ها[4] هستند. خصیصه‌ها در سه سطح انتزاع مطرح می‌شوند. ناحیه‌ها[5] که نام دیگر آن‌ها مجموعه خصوصیات عمده[6] است حاصل تقسیم‌بندی بر روی حوزه‌ی مساله[7] هستند. ناحیه‌ها به چندین فعالیت[8] تقسیم می‌شوند که نام دیگرشان مجموعه خصوصیات[9] است و سطح انتزاع کم‌تری نسبت به ناحیه‌ها دارند. در نهایت در دل یک فعالیت چندین قدم[10] که همان خصیصه است وجود دارد.

از جنبه‌ی معماری نیز می‌توان سیستم را تقسیم‌بندی کرد. معماری شامل چهار لایه‌ی واسط کاربر[11]، حوزه‌ی مساله، مدیریت داده‌ها[12] و ارتباط سیستم‌ها[13] است. در FDD مشکلی با تعدد تیم‌های پیاده‌سازی نداریم ولی تیم مدل‌سازی بهتر است یک تیم باشد. در این متدولوژی نیز مانند ASD از زمان‌بندی با قابلیت تطابق با مشکلات با قرار دادن زمان اضافه استفاده می‌کنیم. در این متدولوژی بر خلاف اکثر متدولوژی‌های چابک فرآیند و

1 Reflect on the Delivery
2 Integration Test
3 System Conversion
4 Features
5 Area
6 Major Feature Set
7 Problem Domain
8 Activity
9 Feature Set
10 Step
11 User Interface
12 Data Management
13 System Interaction

همین‌طور برنامه‌ها بررسی مجدد نمی‌شوند؛ شاید فکر می‌کنند که عمل‌کرد متدولوژی این‌قدر خوب است که نیازی به بررسی مجدد نیست!

FDD دو دسته فاز دارد؛ این فازها که در شکل ۶ دیده می‌شوند شامل سه فاز پشت سرهم با نام‌های تهیه‌ی یک مدل عمومی[۱]، تهیه‌ی لیست خصیصه‌ها[۲]، برنامه‌ریزی با خصیصه[۳] و سپس دو فاز تکرارِی طراحی با خصیصه[۴] و ساخت با خصیصه[۵] هستند. زمان تکرارها در این متدولوژی کم‌تر از دو هفته است.



شکل ۶. فرآیند متدولوژی FDD [1]

در تهیه‌ی مدل عمومی تیم مدل‌کننده تشکیل می‌شود که شامل نماینده‌ی کاربر، سربرنامه‌نویس‌ها[۶]، متخصصان دامنه می‌باشد و تحت سرپرستی سر معمار[۷] است. سپس در قالب یک چرخه مدل‌سازی انجام می‌شود و مدل موجودیت‌ها[۸] به دست می‌آید. در این چرخه‌ی تکراری ابتدا متخصص دامنه یک مرور کلی بر روی حوزه‌ی مساله ارائه می‌دهد سپس مدل‌سازی ناحیه‌ها (که حاصل شکسته شدن حوزه‌ی مساله هستند) به صورت تکراری و با اجرای عملیات‌هایی انجام می‌شود.

عملیات به این صورتند که یک شناخت کلی از دامنه به دست می‌آید، سپس چند زیر گروه از تیم برای مدل کردن یک ناحیه منصوب می‌شوند، مدل‌های ارائه شده توسط آن‌ها ادغام شده و مدل دامنه و مدل کلی اشیا

[1] Develop an Overall Model
[2] Build a Feature List
[3] Plan By Feature (PBF)
[4] Design By Feature (DBF)
[5] Build By Feature (BBF)
[6] Chief Programmer
[7] Chief Architect
[8] Object Model

بدست آمده با هم ادغام می‌شوند. در نهایت انتخاب‌های انجام شده و دلایل آن‌ها (اصولا هر چیزی را که در مستندات ذکر شده نمی‌توان آورد ولی باید ثبت شوند) را به عنوان یادداشت‌های مدل ثبت می‌کنیم.

در تهیه‌ی لیست خصیصه‌ها، تیمی با حضور مدیر پروژه، نماینده‌ی کاربر و سربرنامه‌نویس‌ها تشکیل می‌شود. این تیم اقدام به تهیه‌ی لیست مذکور در سه سطح ناحیه، فعالیت و قدم می‌کند؛ این کار به صورت بالا (ناحیه) به پایین (قدم) انجام می‌شود و انواع تکنیک‌ها را استفاده می‌کنیم که سطوح انتزاع از سه سطح بیش‌تر نشود.

در برنامه‌ریزی با خصیصه ابتدا تیمی متشکل از مدیر پروژه، سربرنامه‌نویس‌ها و مدیر ایجاد[1] می‌سازیم. این تیم زمان تکمیل هرکدام از فعالیت‌ها و در نتیجه زمان تکمیل ناحیه‌ها را مشخص می‌کند و براساس آن برنامه‌ریزی را انجام می‌دهد. مجموعه خصوصیات به سربرنامه‌نویس‌ها داده می‌شوند و هر سربرنامه‌نویس صاحب یک مجموعه خصوصیت می‌شود. کلاس‌ها نیز به برنامه‌نویس‌ها منصوب می‌شوند و بنابراین آن‌ها را صاحب کلاس[2] می‌نامیم.

در طراحی با خصیصه ابتدا تیم خصیصه تشکیل می‌شود. کار به این صورت است که سربرنامه‌نویسی که مسئول یک مجموعه خصیصه است صاحبان کلاس‌هایی که لازم دارد را دور هم جمع می‌کند و در صورت لزوم در شناخت محیط به آن‌ها کمک می‌کند. در این مرحله برای پیاده‌سازی خصیصه‌ها نمودار توالی ایجاد می‌شود و مشخص می‌شود که اشیا چگونه باید تعامل کنند تا یک خصیصه پیاده‌سازی شود. ممکن است لازم باشد مدل اشیا که در قالب یک نمودار کلاس‌هاست برای انطباق با نمودارهای توالی تغییر کند. در نهایت ساختار درونی[3] کلاس‌ها و متدها تهیه می‌شود، طراحی بررسی می‌شود و یک بسته‌ی طراحی ایجاد می‌شود. همان‌طوری که مشخص است رابطه‌ی صاحبان کلاس‌ها و سربرنامه‌نویس‌ها چند به چند است و محدودیتی ندارد.

در نهایت در مرحله‌ی ساخت با خصیصه‌ها متدها به همراه آزمون‌هایشان و بر اساس توصیف سطح طراحی پیاده‌سازی می‌شوند. کدها بررسی می‌شوند تا مطمئن شویم درست هستند و با استانداردها مطابقت دارند. در سطوح مختلف آزمون را انجام می‌دهیم تا مطمئن شویم که نیازمندی‌ها پیاده‌سازی شده‌اند و در نهایت در صورت عدم وجود مشکل حاصل کار را به مجموعه‌ی ساخت اضافه می‌کنیم.

۲. چرخه‌ی حیات متدولوژی‌های چابک

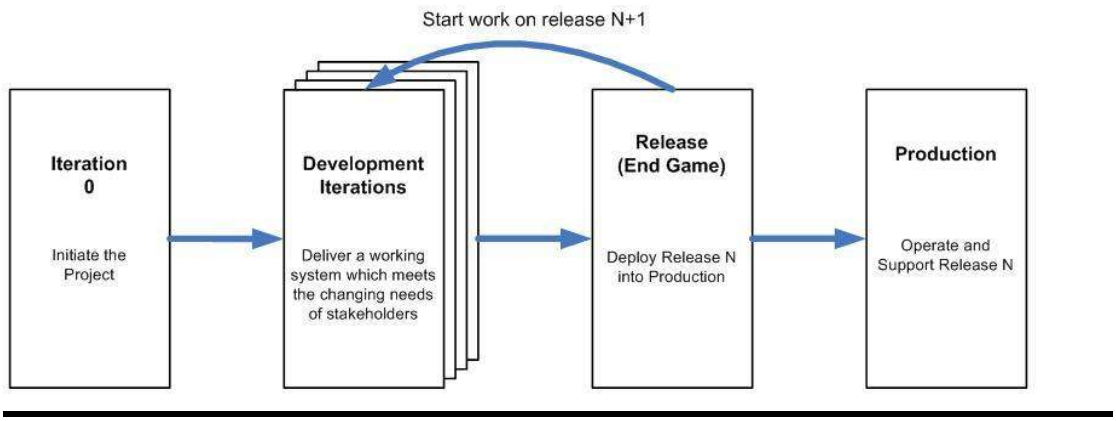
چرخه‌ی حیات تولید سیستم متدولوژی‌های چابک توسط آقای امبلر ارائه شده است و هدف آن دادن دید کلی نسبت به فرآیند موجود در این نوع متدولوژی‌هاست. البته بگذریم از این‌که بعضی متدولوژی‌های چابک ادعا می‌کنند که فرآیند ندارند و به جای کلمه‌ی فاز از توالی[4] استفاده می‌کنند! خصوصیتی که متدولوژی‌های چابک

[1] Development Manager
[2] Class Owner
[3] Prologue
[4] Track

دارند این است که در آن‌ها تکرار، افزایشی بودن و همکاری تاکید می‌شوند. افرادی که در این متدولوژی‌ها کار می‌کنند معمولا در همه‌ی چرخه‌ی تولید نرم‌افزار توانایی دارند.

فرآیند ارائه شده شامل چهار فاز است. در ابتدا شروع پروژه[۱] در تکرار صفر[۲] انجام می‌شود، سپس تکرارهای ایجاد[۳] را داریم. در ادامه تکرارهای ترخیص[۴] می‌آیند و به اصطلاح اتمام بازی[۵] رخ می‌دهد، در پایان نیز فرآوری[۶] را داریم. فازهای این چرخه‌ی عمومی به همراه ارتباطات بین فازها را در شکل ۷ مشاهده می‌کنیم.

همان‌طور که گفتیم در تکرار صفر پروژه آغاز می‌شود. برای این کار باید حمایت و بودجه‌ی لازم را تامین کرد و در نتیجه باید مشخص کنیم که چه چیزی کسب خواهد شد، چقدر هزینه دارد و چقدر طول می‌کشد، در این مرحله لازم است کار خود را با یک امکان‌سنجی اثبات کنیم. پس از اطمینان از حمایت پروژه باید نیازمندی‌های سطح بالا به کمک ذینفعان[۷] پروژه به دست آید. برای این کار از ابزارهای لازم مانند کارت‌های نمایه و تخته‌ی وایت‌برد استفاده می‌شود. هدف فهم نیازمندی‌هاست نه مستند کردن آن، مدل‌سازی دقیق نیازمندی‌ها هر موقع که لازم باشد[۸] و در جلسات مدل‌سازی[۹] انجام می‌شود.



شکل ۷. چرخه‌ی حیات ایجاد سیستم با متدولوژی چابک [2]

در این مرحله است که اقدام به ساخت تیم می‌کنیم؛ مد نظر داریم که تیم در طول زمان تغییر خواهد کرد و در این مرحله اعضای کلیدی تیم را تعیین می‌کنیم. سپس به ارائه‌ی یک معماری اولیه برای سیستم می‌پردازیم. در این معماری حداقل باید ایده‌ای کلی از این‌که چگونه می‌خواهیم سیستم را بسازیم داشته باشیم. این معماری

[1] Project Initiation
[2] Iteration 0
[3] Development Iterations
[4] Release Iterations
[5] End Game
[6] Production
[7] Stakeholder
[8] Just In Time (JIT)
[9] Model Storming

در طول زمان تغییر می‌کند. در ارائه‌ی معماری جزئی نمی‌شویم و هدفمان تعیین یک تدبیر برای معماری[1] است نه تهیه‌ی یک مستند عظیم. در نهایت نیز قبل از شروع تولید به آماده‌سازی محیط کار می‌پردازیم.

در تکرارهای ایجاد به صورت افزایشی سیستم را می‌سازیم. در این تکرارها همکاری از یک طرف بین تولید کننده و دیگر تولید کنندگان و از طرف دیگر بین او و ذینفعان وجود دارد؛ این کار باعث می‌شود چرخه‌ی بازخورد اطلاعات تنگ‌تر شود و همکاری و ارتباطات بهتر شوند. در پیاده‌سازی نیازمندی‌ها بر اساس اولویت‌بندی مد نظر داریم که نیازمندی‌ها و اولویت‌بندی آن‌ها در هر زمان می‌توانند توسط ذینفعان عوض شود و اصولا ذینفعان کنترل تام بر روی حوزه، بودجه و زمان‌بندی دارند.

همان‌طور که گفته شد طراحی‌ها هر موقع که لازم باشد و در جلسات مدل‌سازی انجام می‌شوند؛ مدل‌سازی این جلسات نیز از حد رسم یک سری مدل با دست فراتر نمی‌رود. پیاده‌سازی به همکاری شدید[2] و استفاده از روش تست مبنا[3] نیاز دارد. برای نیازمندی‌های پیچیده که نیاز به بررسی بیش‌تر و طراحی عمیق‌تری دارند کمی زودتر مدل‌سازی می‌کنیم[4] تا وقت تولید کنندگان بیهوده تلف نشود.

برای تضمین کیفیت در ایجاد مکانیزم‌هایی مانند قالب و سبک کد نویسی و مدل‌سازی وجود دارد. امکان ریفکتور کردن[5] به ما اجازه می‌دهد هر وقت که بخواهیم به سمت بهترین طراحی برویم. نرم‌افزارهای تولید شده نیز به طور منظم، به صورت نا تمام در حالی که کار می‌کنند برای کاربران نمایش داده می‌شود و به آن‌ها تحویل می‌شود. هرچه تحویل‌ها زودتر و بیش‌تر باشند آزمون یکپارچگی را بهتر می‌توان انجام داد. در تولید همیشه آزمون را مد نظر داریم این آزمون‌ها سطوح مختلف از آزمون‌های برنامه‌نویسان تا آزمون‌های مقبولیت را می‌پوشانند.

در تکرارهای ترخیص آزمون نهایی سیستم انجام می‌شود. در صورت لزوم و بر مبنای مشکلاتی که پیدا می‌شود کارهایی انجام می‌شود و برخی از مشکلات رفع می‌شوند. بسته به اندازه‌ی سیستم و مشکلات موجود این کار ممکن است چندین تکرار را بطلبد. مستندات نیز مانند هر نیازمندی دیگری باید بسته شوند، اولویت بندی شوند و تنها در صورتی که ذینفعان بخواهند ایجاد شوند. در این مرحله مستنداتی که باید تحویل شوند بسته می‌شوند. در نهایت کاربران را آموزش داده و سیستم را مستقر[6] می‌کنیم.

در فاز فرآوری سیستمی که در محیط کاربر مستقر شده را پشتیبانی می‌کنیم و نمی‌گذاریم از مفید بودن خارج شود. اتمام این فاز فقط در شرایطی است که سیستم می‌خواهد بازنشسته شود یا اصولا نسخه‌ی جدیدی از آن آمده است و باید نسخه‌ی جدید را فرآوری کرد و به هر حال سرمایه‌گذاری و حمایت از سیستم قبلی دیگر انجام نمی‌شود. این مرحله معمولا یک تکراری است چون تنها یک ترخیص از نرم‌افزار را در بر می‌گیرد.

---

[1] Architecture Strategy
[2] Highly Collaborative
[3] Test Driven Development (TDD)
[4] Model just a bit ahead
[5] Refactoring
[6] Deploy

۳. انطباق متدولوژی‌ها ارائه شده با چرخه‌ی حیات

در این مرحله می‌خواهیم متدولوژی‌های چابک معرفی شده را در قالب چرخه‌ی حیات ارائه شده با هم انطباق دهیم و در مواقع لزوم چرخه‌ی ارائه شده را گسترش بخشیم. به این منظور هرکدام از فازهای ارائه شده در این چرخه‌ی حیات را تک به تک با متدولوژی‌های چابک انطباق می‌دهیم. در این میان فازهایی که به چرخه اضافه شده‌اند نیز ارائه می‌شوند.

روش انطباق یک فاز از چرخه‌ی حیات با متدولوژی‌ها به این صورت است که ابتدا فعالیت‌های بیان شده برای آن فاز در چرخه‌ی حیات را در دست می‌گیریم؛ سپس فازهایی از متدولوژی مورد بررسی که بر این فاز انطباق دارند را می‌سنجیم و حضور فعالیت‌های ذکر شده در آن‌ها را بررسی می‌کنیم؛ در این میان ممکن است فعالیت‌هایی در متدولوژی‌های مورد بررسی وجود داشته باشند که مجموعه فعالیت‌های اولیه اضافه شوند و آن را گسترش دهند.

برای بیان نتیجه‌ی این بررسی ابتدا تعریف مختصری از فاز و فعالیت‌های موجود در آن ارائه می‌کنیم. نتایج بررسی را در دو جدول ارائه می‌کنیم. در جدول دوم فعالیت‌هایی از چرخه که هر متدولوژی انجام می‌دهد مشخص می‌شوند. در مواردی لازم است توضیحاتی به جدول اضافه شوند که پس از جدول می‌آیند.

در جدول اول فازهایی از متدولوژی‌های چابک که توسط این فاز از چرخه‌ی عمومی پوشانده می‌شود بیان می‌شوند. فازهای متدولوژی‌ها همان‌طور که دیدیم در سطوح مختلف ارائه شده‌اند. اگر کل یک فاز سطح بالا توسط این فاز از چرخه پوشانده شده باشد در جدول نام همان فاز را می‌آوریم اما اگر قسمتی از آن پوشانده شده باشد نام فازهای سطح پایین‌تری را می‌آوریم (زیر مجموعه‌ی آن هستند) که توسط چرخه پوشانده شده‌اند. در مورد فازهایی که بیش‌تر شکسته نمی‌شوند و تنها بخشی از آن‌ها پوشانده شده به ذکر این مطلب بسنده می‌کنیم.

در بررسی متدولوژی چابک سعی می‌کنیم تنها به چیزهایی که مستقیما توسط خود متدولوژی بیان شده بپردازیم. درست است که یک متدولوژی ممکن است قابلیت انجام کاری را داشته باشد اما اگر به آن اشاره‌ی مستقیم نکرده باشد آن را در نظر نمی‌گیریم. شاید کمبود فعالیتی در یک متدولوژی از کمبود دانش ما در مورد آن متدولوژی باشد. اما بررسی بیش‌تر متدولوژی‌های ذکر شده از حوزه‌ی این مستند خارج است.

انطباقی که بین چرخه و متدولوژی‌ها برقرار می‌کنیم قطعا قسمت‌هایی از متدولوژی‌ها را نمی‌پوشاند. به طور مثال متدولوژی ASD امکان این را می‌دهد که ناگهان همه چیز را رها کنیم و به سراغ کاری مهم‌تر رویم. این امکان حتی در فرآیند خود متدولوژی هم در نظر گرفته نشده است. یا مثلا در FDD طراحی داریم که این موضوع در هیچ‌کدام از دیگر متدولوژی‌ها این نمود را ندارد. در انتها نیز در ارائه‌ی یک چرخه‌ی کامل بر اساس فازهای ارائه شده این مشکل برقرار خواهد بود که در همان‌جا به مشکل اشاره می‌کنیم.

از طرفی سطح درشت دانگی کارها نیز در متدولوژی‌ها متفاوت است. مثلا تحلیل نیازمندی‌ها می‌تواند در سطوح مختلفی انجام شود. در این‌جا نیز ما وقتی که کلیتی از دو متدولوژی مانند هم هستند آن‌ها را مشترک می‌گیریم هرچند که سطح عمل آن‌ها متفاوت باشد. به طور مثال سطح تحلیل اولیه‌ی نیازمندی‌ها در FDD

خیلی وسیع‌تر از این کار در XP است؛ اما چون هر دو به کلیتی از یک مفهوم توجه کرده‌اند ما هر دو را دارای این فعالیت در نظر می‌گیریم.

اگر این فرض‌های ساده کننده را انجام نمی‌دادیم، انطباق متدولوژی‌ها ممکن نبود چون هر متدولوژی دقیقا شرایطی را به ما دیکته می‌کرد که در انطباق به خودش برسیم. این فرض‌های ساده کننده کمی شرایط را راحت‌تر می‌کنند.

۳ - ۱. تکرار صفر

در این مرحله پروژه را شروع می‌کنیم. بررسی‌های لازم، تخمین‌ها و برنامه‌ریزی‌های اولیه را انجام می‌دهیم و حوزه‌ی مساله را مشخص می‌کنیم. در این مرحله فعالیت‌های زیر وجود دارند:

- کسب حمایت و بودجه: باید مشخص شود که پشتیبانی لازم از پروژه و شروع آن به عمل خواهد آمد.
- تخمین و برنامه ریزی: در این فعالیت دید کلی از مدت زمان و هزینه‌ی کار به دست می‌آید. این فعالیت در چرخه مستقیما و به عنوان برنامه‌ریزی بیان نشده است.
- امکان سنجی: این موضوع در خیلی از متدولوژی‌ها به این استدلال که قبلا نمونه‌ی پروژه انجام شده است وجود ندارد.
- استخراج نیازمندی‌های سطح بالا: معمولا به زبان کاربر هستند.
- کمک ذینفعان: در این مرحله از پروژه معمولا کاربران را نیز درگیر می‌کنیم.
- ساخت تیم: معمولا شالوده‌ی اولیه تیم در این مرحله ریخته می‌شود و جذب افراد انجام می‌شود.
- معماری اولیه: ایده‌ی کلی از چگونگی ساخت سیستم را به ما می‌دهد.
- آماده‌سازی محیط و منابع: باید قبل از شروع به کار این‌جور نیازها را رفع کنیم.
- استفاده از پروتوتایپ: برای شناخت بهتر نیازمندی‌ها گاهی از پروتوتایپ‌ها نیز استفاده می‌کنیم. این فعالیت توسط خودمان به چرخه اضافه شده است.
- اولویت بندی نیازمندی‌ها: باید مشخص شود که چه نیازمندی‌هایی مهم‌تر هستند تا اول با آن‌ها شروع کنیم. این فعالیت در چرخه‌ی اولیه وجود ندارد.

در جدول ۱ و ۲ متدولوژی‌های موجود با این قسمت از چرخه‌ی عمومی انطباق داده شده‌اند.

	فازهایی که پوشانده می‌شوند
DSDM	قبل پروژه، امکان سنجی، بررسی کسب و کار، مدل‌کردن عملیات
اسکرام	قبل بازی
XP	اکتشاف، برنامه ریزی (برای ترخیص)
ASD	شروع پروژه، بخشی از برنامه‌ریزی منعطف چرخه
dX	آغاز
کریستال	بر قرار کردن

	فازهایی که پوشانده می‌شوند
FDD	تهیه‌ی یک مدل عمومی، تهیه‌ی لیست خصیصه‌ها، برنامه‌ریزی با خصیصه

جدول ۱. فازهای پوشانده شده از متدولوژی‌های چابک توسط تکرار صفر از چرخه

	DSDM	اسکرام	XP	ASD	dX	کریستال	FDD
حمایت				✓		✓	
تخمین	✓	✓	✓	✓	✓	✓	✓
امکان سنجی	✓					✓	
نیازمندی‌ها	✓	✓	✓	✓	✓	✓	✓
کمک ذینفعان	✓	✓	✓	✓	✓	✓	✓
ساخت تیم		✓	✓	✓		✓	✓
معماری	✓	✓	✓		✓		
محیط و منابع	✓					✓	
پروتوتایپ	✓	✓	✓		✓		
اولویت بندی	✓	✓	✓	✓	✓	✓	

جدول ۲. انطباق فعالیت‌های تکرار صفر چرخه با متدولوژی‌های چابک

در انجام فعالیت برنامه‌ریزی برخی از متدولوژی‌ها سعی می‌کنند تا انتها را ببینند ولی برخی دیگر مانند XP فقط تا اولین ترخیص را بررسی می‌کنند.

dX اولویت بندی نیازمندی‌ها را مستقلا و به عنوان یک فعالیت معرفی نمی‌کند ولی بالاخره باید در این فاز آن را انجام دهد به این دلیل که باید در فاز بعدی بر اساس آن معماری را ایجاد کند.

۳ - ۲. ایجاد

در استخراج فعالیت‌های موجود در تکرارهای ایجاد به مجموعه‌ای از فعالیت‌ها رسیدیم. برخی متدولوژی‌ها مانند XP در این مرحله خیلی فعال بودند و ما نتوانستیم تمام اصول (برنامه نویسی دو نفره و ...) آن‌ها را پوشش دهیم اما تقریبا توانستیم اجزای کلی فرآیند آن‌ها را در این مرحله بپوشانیم. در این مرحله فعالیت مربوط به برنامه‌ریزی را در نظر نگرفتیم و حتی برنامه ریزی‌های مربوط به هر تکرار را به تکرار صفر فرستادیم.

برخی متدولوژی‌ها هم ایده‌های جالبی در سطح فرآیند داشتند که البته نمی‌توانستیم در قالب چرخه اعمالشان کنیم چون هیچ‌کدام از متدولوژهای دیگر از آن پشتیبانی نمی‌کردند. مثلا چند سطحی بودن فرآیند در مرحله‌ی تولید در حد زمان‌های بسیار کوچک در کریستال ایده‌ی جالبی بود این چند سطحی بودن در متدولوژی‌های دیگر حداکثر تا دو سطح پیش‌رفته بود. شروع کردن کریستال با حداقل چارچوب و تکامل آن نیز

ایده‌ی جالبی بود. در نهایت بحث طراحی و حضور آن در FDD نیز موضوع جالبی به نظر می‌رسید. در ادامه فعالیت‌های مشترک استخراج شده را می‌بینیم:

- تولید افزایشی: به وسیله‌ی تکرارها و به تدریج سیستم را کامل می‌کنیم.
- همکاری تولید کنندگان: تولید کنندگان شدیدا با هم ارتباط دارند و تبادل اطلاعات انجام می‌دهند.
- کنترل تام ذینفعان بر برنامه: کاربران بر تمام قسمت‌های برنامه و فرآیند دسترسی دارند و می‌توانند آن را تغییر دهند.
- طراحی برنامه: این کار که معمولا در جلساتی انجام می‌شود درک کلی از چگونگی پیاده‌سازی یک قسمت از سیستم به ما می‌دهد.
- آزمون مبنا: از آزمون‌ها در سطوح مختلف بسیار استفاده می‌شود و بر اساس آن‌ها پیش می‌رویم.
- مدل‌سازی کمی‌زودتر یا دیدن کل: قبل از این‌که وارد تک‌تک اجزا شویم دیدی کلی نسبت به آن‌ها به دست می‌آوریم و ارتباطات آن‌ها را در نظر می‌گیریم و یک طراحی سطح بالا انجام می‌دهیم. این کار می‌تواند حتی در قالب یک معماری دور ریختنی باشد.
- قالب و سبک کد نویسی مشترک: این کار برای مالکیت عمومی کد لازم است.
- ریفکتور کردن: در آینده (پس از تولید) می‌نشینیم و مشکلات و کاستی‌ها را با تغییر کنترل شده در سیستم و طراحی آن رفع می‌کنیم. با این کار طراحی سیستم را بهبود می‌بخشیم.
- نمایش برای کاربران: این نمایش اولیه‌ی درون سازمانی به منظور گرفتن نظرات کاربران و اعمال آن‌ها در ترخیص استفاده می‌شود. در حقیقت از همکاری کاربران استفاده می‌شود تا سیستم بهتر و بی‌نقص‌تر تولید شود.
- یکپارچه سازی: عملیات انجام شده در قالب یک کل سازماندهی می‌شوند و کارهای جدید به آن اضافه می‌شوند؛ همیشه یک سیستم داریم که آخرین فعالیت‌ها در آن موجود است. این فعالیت توسط خودمان به چرخه اضافه شده است.
- جلسات روزانه و ایستاده: در این جلسات مشکلات از جلسه‌ی قبل تا کنون مطرح می‌شود و همگان سعی در بررسی و کمک دارند. همه‌ی افراد تیم در این جلسه حضور دارند و زمان آن کوتاه است. در نهایت نیز برنامه‌ی هر شخص تا جلسه‌ی بعد اعلام می‌شود. این فعالیت نیز توسط خودمان اضافه شده است.
- بررسی پیشرفت کار و فراگیری دروس: هر چند وقت یک بار اتفاقاتی رخ داده را مرور می‌کنیم و از آن‌ها درس می‌گیریم. چیزهایی که یاد می‌گیریم را در آینده و در سطوح مختلف از خود فرآیند گرفته تا برنامه‌ها اعمال می‌کنیم. این فعالیت که توسط خودمان اضافه شده است تنها به وجود عبرت گیری می‌پردازد و به سطح اعمال آن کاری ندارد.
- تعیین قدم بعدی: در انتهای هر تکرار می‌توانیم تعیین کنیم در ادامه به کدام سمت برویم و مرحله‌ی بعدی که باید پیش گرفته شود را انتخاب می‌کنیم.

در جدول ۳ و ۴ متدولوژی‌های موجود با تکرارهای ایجاد انطباق داده شده‌اند.

	فازهایی که پوشانده می‌شوند
DSDM	طراحی و ساخت
اسکرام	تولید (بازی)
XP	تکرارها برای اولین ترخیص
ASD	بخشی از برنامه‌ریزی منعطف چرخه، مهندسی هم‌زمان اجزا، مرور کیفیت
dX	جزئی‌شدن، ساخت
کریستال	تحویل چرخه‌ای
FDD	طراحی با خصیصه، ساخت با خصیصه

جدول ۳. فازهای پوشانده شده از متدولوژی‌های چابک توسط تکرارهای ایجاد از چرخه

	DSDM	اسکرام	XP	ASD	dX	کریستال	FDD
تولید افزایشی	✓	✓	✓	✓	✓	✓	✓
همکاری		✓	✓	✓	✓		✓
کنترل ذینفعان			✓		✓		
طراحی برنامه	✓		✓	✓	✓	✓	✓
آزمون مبنایی	✓		✓	✓	✓	✓	✓
دید سطح بالا			✓	✓			✓
قالب و سبک			✓		✓		
ریفکتور کردن			✓	✓	✓		
ارائه به کاربران	✓	✓	✓	✓		✓	
یکپارچه سازی	✓		✓	✓	✓	✓	✓
جلسات روزانه		✓	✓			✓	
درس آموختن		✓	✓	✓	✓	✓	
تعیین قدم بعدی	✓	✓		✓			

جدول ۴. انطباق فعالیت‌های تکرارهای ایجاد چرخه با متدولوژی‌های چابک

طراحی برنامه در XP توسط خود برنامه نویس و با استفاده از کارت‌های CRC انجام می‌شود و در جلسه نیست.

۳ - ۳. ترخیص

فعالیت‌های انجام شده در این مرحله در برخی از متدولوژی‌ها به خوبی پشتیبانی نمی‌شود. در این ترخیص هدف ما معرفی محصول به محیط کاربر است. فعالیت‌های این مرحله عبارتند از:

- آزمون نهایی سیستم: سیستم مورد آزمون نهایی قرار می‌گیرد و مطمئن می‌شویم که مورد قبول کاربر است. در این مرحله ممکن است از آزمون بتا[1] نیز استفاده کنیم.
- رفع کردن مشکلات یافته شده: مشکلاتی که در آزمون پیدا شده‌اند باید رفع شوند.
- بستن مستندات: مستنداتی که کاربر از ما انتظار دارد باید در این مرحله نهایی شوند.
- آموزش کاربران: کاربران آموزش داده می‌شوند که چگونه باید با سیستم کار کنند.
- استقرار سیستم: سیستم در محل مشتری مستقر می‌شود و تمام کارهای مرتبط مثل تبدیلات لازم و کارهای مشابه برای عملیاتی کردن سیستم انجام می‌شوند.

در جدول ۵ و ۶ انطباق متدولوژی‌های چابک با تکرار ترخیص را مشاهده می‌کنیم.

	فازهایی که پوشانده می‌شوند
DSDM	پیاده‌سازی
اسکرام	بعد بازی
XP	محصول سازی
ASD	بخشی از تضمین کیفیت نهایی و ترخیص
dX	انتقال
کریستال	بخشی از خاتمه دادن
FDD	بسیار محدود در ساخت با خصیصه

جدول ۵. فازهای پوشانده شده از متدولوژی‌های چابک توسط تکرار ترخیص چرخه

	DSDM	اسکرام	XP	ASD	dX	کریستال	FDD
آزمون نهایی	✓	✓	✓	✓	✓	✓	✓
رفع مشکلات			✓	✓			
بستن مستندات	✓	✓	✓	✓			
آموزش کاربران	✓	✓	✓	✓	✓		
استقرار سیستم	✓	✓	✓	✓	✓	✓	

جدول ۶. انطباق فعالیت‌های تکرار ترخیص چرخه با متدولوژی‌های چابک

همان‌طور که اشاره شد، برخورد متدولوژی‌ها با این فاز بسیار متفاوت است. اصولا FDD هیچ توجهی به این فاز نداشته است، کریستال و ASD نیز به گونه‌ای دچار مرگ زودرس درست پس از این فاز می‌شوند. ASD حتی

[1] Beta Test

در این مرحله بین ترخیص نهایی و مرگ سیستم تصمیم می‌گیرد. dX نیز سیستم قدیمی را برای جلوگیری از بروز مشکلات در کنار سیستم جدید نگهداری می‌کند.

مشکلاتی که در این فاز و با اجرای آزمون‌ها یافته می‌شوند یا باید در همین فاز حل شوند، یا باید توسط تکرار چرخه‌ی اصلی رفع شوند.

ارتباط نزدیکی بین این فاز و سطح تاثیر آن با دو فاز بعدی و همچنین فاز قبلی وجود دارد؛ وقتی دو فاز نهایی (نگهداری و مرگ) به صورت کامل پیاده‌سازی نمی‌شوند (FDD، ASD و کریستال)، ارائه‌های به کاربران در فاز ایجاد تنها به صورت محدود و به زیر مجموعه‌ای از کاربران خواهد بود و در تکرار ترخیص برای اولین بار سیستم را وارد محیط کاربر می‌کنیم و حالت موازی بودن آن با تکرار ایجاد از دست می‌رود.

با این دید تکرارهای ایجاد و ترخیص به صورت تکراری انجام نمی‌شوند و پشت سر هم خواهند بود. البته می‌توان آن ارائه‌های محدود را در فاز ترخیص گرفت ولی در این صورت عنصر مهمی از تکرار ترخیص که در محیط کاربر بودن است از دست می‌رود. در بخش ۳-۶ که به ارتباطات فازها می‌پردازیم این موضوع را بازتر می‌کنیم.

۳ - ۴. نگهداری

در این مرحله تنها به فعالیت پشتیبانی سیستم و نگهداری آن می‌پردازیم. البته این فعالیت، یک فعالیت بزرگ محسوب می‌شود. این فاز در چرخه‌ی اصلی فرآوری نامیده شده است ولی ما آن را نگهداری نامیدیم. برخورد با این فاز در متدولوژی‌های مختلف با هم بسیار تفاوت دارد. بعضی از متدولوژی‌ها اصولا هیچ اشاره‌ای به این فاز نکرده‌اند، برخی اشاره کرده‌اند ولی آن را داخل چرخه‌ی حیات خود نیاورده‌اند و برخی نیز آن را مستقیما و در داخل خود مطرح کرده‌اند. در بین متدولوژی‌هایی به این فاز فکر کرده‌اند (چه خارج از چرخه‌ی خود و چه داخل آن) در بین همه فرض بر این است که در این فاز موتور ایجاد دوباره به راه می‌افتد. در این مفهوم برخی آن را جدی‌تر گرفته‌اند (XP) و برخی نیز آن را خارج از چرخه‌ی خود و با تکرار چرخه در نظر گرفته‌اند (کریستال).
تک فعالیت این فاز عبارت است از:

- پشتیبانی و نگهداری: برخی کارها وجود دارند که پس از مستقر شدن سیستم لازم هستند. این کارها می‌توانند از اصلاح مشکلی از سیستم، تا اضافه کردن یک تکه‌ی عملیاتی جدید بر اساس نیاز جدیدی از مشتری باشند.

در جدول‌های ۷ و ۸ سعی کرده‌ایم فاز نگهداری را با متدولوژی‌های معرفی شده تطبیق دهیم.

	فازهایی که پوشانده می‌شوند
DSDM	پس از پروژه
اسکرام	خارج از چرخه
XP	نگهداری
ASD	ندارد

	فازهایی که پوشانده می‌شوند
dX	ندارد
کریستال	خارج از چرخه
FDD	ندارد

جدول ۷. فازهای پوشانده شده از متدولوژی‌های چابک توسط تکرار نگهداری چرخه

	DSDM	اسکرام	XP	ASD	dX	کریستال	FDD
پشتیبانی و نگهداری	✓		✓				

جدول ۸. انطباق فعالیت‌های تکرار نگهداری چرخه با متدولوژی‌های چابک

بعضی از متدولوژی‌ها از روی این فاز پریده‌اند و به فاز آخر که به آن اشاره خواهیم کرد پریده‌اند. نگهداری یکی از مهم‌ترین قسمت‌های تولید نرم‌افزار است و متدولوژی‌هایی که اصلا این فاز را ندارند قطعا آن را مدنظر داشته‌اند ولی چون آن را خارج از چرخه‌ی خود (با تکرار چرخه‌ی خود) می‌دانسته‌اند، اشاره‌ای به آن نکرده‌اند. البته برخی هم آن‌قدر خودشان موتور ایجاد را تکرار می‌کنند که دیگر چیزی برای نگهداری نماند (ASD) اما مستقیما به نگهداری اشاره‌ای نمی‌کنند.

۳ - ۵. مرگ

وقتی پروژه و سیستم از دست تیم خارج می‌شود وارد این مرحله می‌شویم. کارهای نهایی با سیستم انجام می‌شود و آن را به عنوان یک کار تمام شده ثبت می‌کنیم. فعالیت‌های این فاز به این قرارند:

- بستن پروژه: پروژه خاتمه یافته اعلام می‌شود و رسما بسته می‌شود.
- مستند سازی پس از مرگ: تجربیات کسب شده در تولید سیستم ثبت می‌شوند.

	فازهایی که پوشانده می‌شوند
DSDM	ندارد
اسکرام	ندارد
XP	مرگ
ASD	قسمتی از تضمین کیفیت نهایی و ترخیص
dX	ندارد
کریستال	قسمتی از خاتمه دادن
FDD	ندارد

جدول ۹. فازهای پوشانده شده از متدولوژی‌های چابک توسط فاز مرگ چرخه

جدول‌های ۹ و ۱۰ نشان دهنده‌ی انطباق متدولوژی‌های چابک با فاز مرگ می‌باشند.

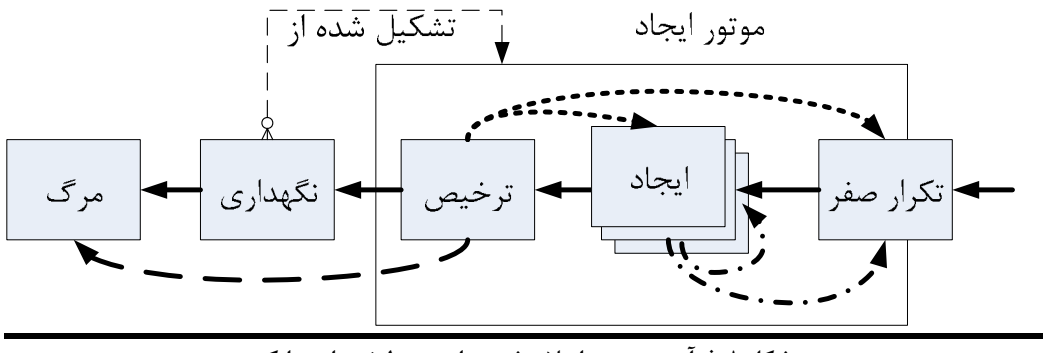
	DSDM	اسکرام	XP	ASD	dX	کریستال	FDD
بستن پروژه			✓	✓			
مستندسازی بَعدِ مرگ			✓	✓		✓	

جدول ۱۰. انطباق فعالیت‌های فاز مرگ چرخه با متدولوژی‌های چابک

۳ - ۶. فرآیند عمومی اصلاح شده برای متدولوژی‌های چابک

تا کنون به هرچه پرداختیم تک فازهای فرآیندهای چابک بودند. در این قسمت به چگونگی ارتباط این فازها و حرکت روی آن‌ها می‌پردازیم. در بخش‌های قبلی کمی این مفهوم و برخی شبهات در مورد آن را مطرح کردیم. ترتیب بین فازها که با گذشت زمان در طول پروژه معین می‌شود کاملا مشخص است؛ ابتدا تکرار صفر انجام می‌شود سپس ایجاد و بعد از آن ترخیص کردن حاصل فاز ایجاد را داریم. در نهایت هم به نگهداری و پس از آن مرگ پروژه می‌رسیم. این ارتباطات در شکل ۸ با استفاده از خطوط جهت دار بین فازها نمایش داده شده‌اند.

همان‌طور که در اکثر متدولوژی‌های چابک مشاهده کردیم بحث بازگشت به مراحل قبل و تکرار یک فاز موضوع بسیار مطرحی می‌باشد و به دفعات دیده می‌شود. به وسیله‌ی این بازگشت‌هاست که فرآیند تکراری-افزایشی می‌شود. دو جور بازگشت در بین فرآیندهای چابک بررسی شده وجود دارد. این دو دید را در انتهای بخش ۳-۳ که تکرار ترخیص را بررسی نمودیم به اجمال معرفی کردیم. تفاوت این دو دید از آن‌جا ناشی می‌شود که در یکی ترخیص‌ها به محیط کاربر منتقل می‌شوند، سیستم عملیاتی می‌شود و تمام کاربران آن را می‌بینند. در دیگری فقط به صورت محدود و داخل سازمان تولید کننده ترخیص انجام می‌شود و سیستم تنها به زیرمجموعه‌ای از کاربران نمایش داده می‌شود؛ ترخیص با این محدودیت از دید ما در همان تکرارهای ایجاد انجام می‌شود. وقتی به فاز ترخیص می‌رویم باید سیستم در محیط کاربر عملیاتی شود. پس نوع محدود آن را در فاز ترخیص در نظر نمی‌گیریم.



شکل ۸. فرآیند عمومی اصلاح شده برای متدولوژی‌های چابک

بازگشت‌های نوع اول را در شکل ۸ با خطوط جهت‌دار نقطه‌چین (بالای شکل) نمایش دادیم. بازگشت‌های نوع دوم توسط خطوط جهت‌دار نقطه-خط‌چین (پایین شکل) نمایش داده شدند. بدیهی است که یکی از این دو مجموعه بازگشت‌ها برای یک متدولوژی انتخاب می‌شوند.

بحث مطرح دیگر در این چرخه موضوع نگهداری است. خیلی از متدولوژی‌ها نگهداری را در چرخه‌ی خود نمی‌بینند و آن را خارج از چرخه می‌دانند. در این صورت فاز مرگ مطرح شده برای آن‌ها بسیار محدود و تنها کمی فراتر از کارهای پس از تکرار می‌شود. فاز مرگ با تمامی فوایدی که دارد تنها پس از اتمام کل پروژه است که می‌تواند برقرار شود.

اگر متدولوژی‌هایی که ادعا می‌کنند نگهداری را خارج چرخه و با تکرار آن انجام می‌دهند مدل خود را ارائه دهند باز هم به همان مدل قبلی می‌رسیم. یعنی مرگ آن‌ها در کارهای پس از چرخه انجام می‌شود و نگهداری که آن را خارج از خود می‌دانستند در فاز نگهداری انجام می‌شود. در نهایت هم بالاخره سیستم روزی می‌میرد و باید فاز مرگ را برای آن داشت.

اما ما برای پایبند بودن به میثاق خود که رعایت دقیق همان چیزی است که خود متدولوژی گفته است، این دید را کنار گذاشتیم و با خط جهت‌داری که به صورت خط‌چین نمایش داده شده و در گوشه‌ی پایین و سمت چپ شکل ۸ حضور دارد امکان پرش از فاز نگهداری را دادیم. بدیهی است که یا باید فاز نگهداری را در نظر گرفت و این ارتباط را کنار گذاشت یا این ارتباط را حفظ کرد و فاز نگهداری و تمام ورودی و خروجی‌های آن را حذف کرد.

موتور تولید همان‌طور که در شکل ۸ نیز مشاهده می‌شود فازهای ایجاد، ترخیص و قسمتی از تکرار صفر را در بر می‌گیرد. این‌که چرا ایجاد و ترخیص جزئی از آن هستند بدیهی به نظر می‌رسد. اما کامل نپوشاندن تکرار صفر شاید بدیهی به نظر نرسد؛ برخی کارها که در تکرار صفر انجام می‌شوند غیر قابل تکرار و اصلاح هستند. به طور مثال در تامین منابع و خرید ملزوماتی مانند صندلی دیگر قابلیت بازگشت وجود ندارد! دلیل بازگشت به این فاز در موتور تولید می‌تواند اصلاح برنامه‌ها، تغییر نیازمندی‌ها و ... باشد ولی اصلاحات تمام فعالیت‌های فاز را در بر نمی‌گیرد. پس ما نیز موتور ایجاد را تنها شامل قسمتی از تکرار صفر در نظر گرفتیم. موتور ایجاد یال خروج از ترخیص را نیز در بر نمی‌گیرد و وقتی در فاز نگهداری تکرارش می‌کنیم پس از آخرین ترخیص به مرگ خواهیم رفت.

فاز نگهداری متشکل از سه فاز پیشین (موتور ایجاد) است. اگر به دید علوم کامپیوتر بررسی کنیم شاید ماشین حالتی که با تکرار موتور ایجاد در فاز نگهداری به دست می‌آید قابل ساده کردن باشد. متدولوژی‌ها مختلف دید متفاوتی را در این زمینه دارند و معیار آن‌ها برای انتقال به نگهداری متفاوت است؛ برخی مانند XP این کار را زودتر و برخی دیرتر انجام می‌دهند. بنابر موضوع مطرح شده مشخص می‌شود که واقعا این استفاده‌ی مجدد لازم است و قابل ساده‌تر شدن نیست. از طرفی حضور موتور تولید در نگهداری تفاوت‌هایی با حالت عادی دارد؛ مثلا ممکن است زمان تکرارها در نگهداری با حالت عادی فرق کند.

۴. الگوهای فرآیند متدولوژی‌های چابک

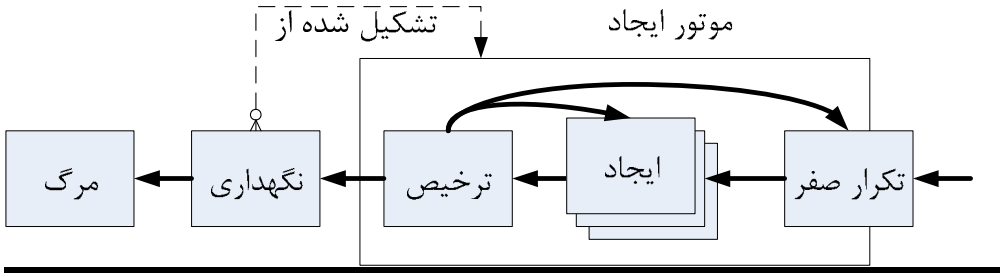
در بخش‌های قبل به بررسی شباهت فازهای متدولوژی‌های چابک پرداختیم. در این راه فعالیت‌هایی شناسایی شدند که هرکدام قسمتی از کارها را انجام می‌دادند. پایه‌ی الگوهای فرآیندی که در این بخش ارائه می‌کنیم نیز همین فعالیت‌ها هستند.

در این بخش ابتدا چهار فرآیند کلی متصور برای متدولوژی‌های چابک را ارائه می‌کنیم. سپس به ارائه‌ی الگوهای فرآیند فاز[1] و مرحله[2] می‌پردازیم و تک‌تک فازهای فرآیند عمومی مورد بررسی قرار می‌گیرند. در قالب یک شکل برای هر کدام الگوهای فاز را ارائه می‌دهیم و سپس اجزای تشکیل دهنده‌ی الگوهای مرحله‌ی را نام می‌بریم. به دلیل محدودیت زمان و فضا برای الگوهای مرحله شکل ارائه نمی‌دهیم. در نهایت نیز پاد الگوها را می‌بینیم.

فعالیت‌های تشکیل دهنده‌ی الگوهای فرآیند مرحله به صورت مختصر در بخش ۳ که برهم‌نهی متدولوژی‌ها را انجام دادیم توضیح داده شدند؛ به همین دلیل در این بخش به توضیح مختصر از آن‌ها بسنده می‌کنیم. این فعالیت‌ها می‌توانند به صورت الگوهای فرآیند کار[3] معرفی شده و شرح داده شوند؛ در این الگوها به چگونگی انجام عملیات پرداخته می‌شود؛ به طور مثال برنامه ریزی به صورت‌های مختلفی قابل انجام است و می‌توان محدودیت را بر روی زمان و یا نیازمندی‌ها قرار داد. اما الگوهای فرآیند کار از حوزه‌ی این نوشتار خارج هستند پس ما به آن‌ها نمی‌پردازیم.

۴ - ۱. فرآیند کلی

در شکل ۸ فرآیند عمومی حاکم بر متدولوژی‌های چابک را مشاهده کردیم. در این‌جا این فرآیند عمومی را بازتر کرده و چهار انتخاب ممکن فرآیند که می‌توان از آن استخراج کرد را ارائه می‌کنیم.

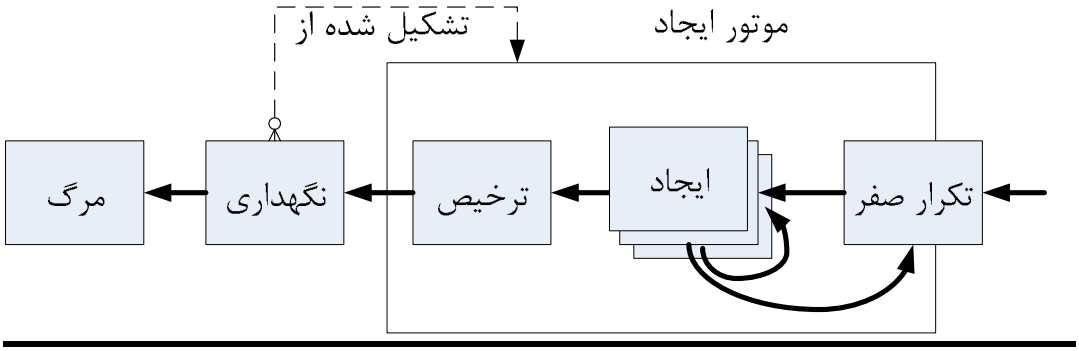


شکل۹. بازارایی اول فرآیند عمومی

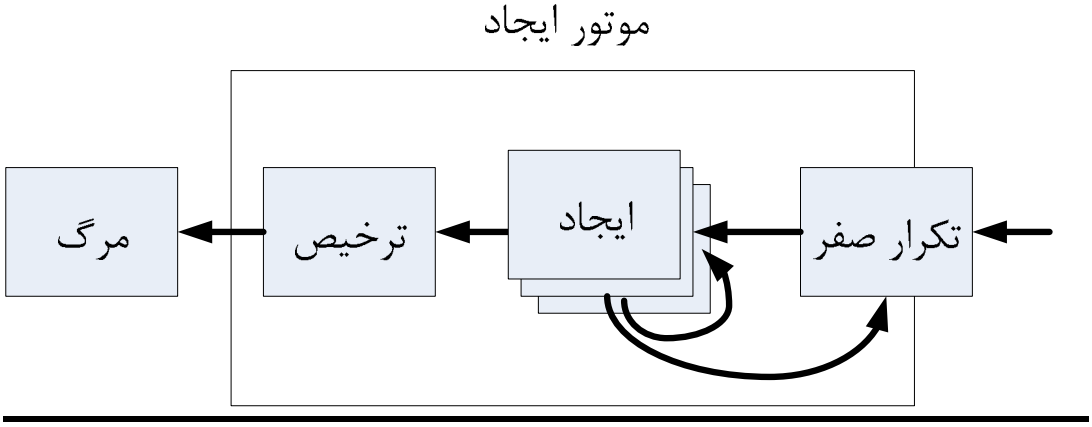
[1] Phase Process Patterns
[2] Stage Process Patterns
[3] Task Process Patterns

در شکل ۹ حالتی از فرآیند را داریم که مرحله‌ی ترخیص آن به صورت کامل انجام می‌شود و مرحله‌ی نگهداری هم حضور دارد. در این فرآیند در انتهای هر تکرار محصول وارد محیط کاربر می‌شود. و پس از ترخیص در صورت نیاز به تغییر برنامه، زمان بندی یا مجموعه‌ی نیازمندی‌ها به تکرار صفر بر می‌گردیم و گرنه حلقه‌ی ایجاد و ترخیص را ادامه می‌دهیم. در انتها نیز با تکرار موتور تولید به نگهداری سیستم می‌پردازیم. تکرار صفر و مرگ نیز همان طور هستند که قبلا توضیح داده شده است.

نوع دیگری از فرآیند را در شکل ۱۰ می‌بینیم. در این حالت ترخیص کامل در محیط مشتری و در برخی موارد حتی یکپارچه سازی محصول در داخل سازمان تولید کننده وجود ندارد. معمولا در فازهای ایجاد و به صورت محدود محصول را به زیرمجموعه‌ای از کاربران نمایش می‌دهند و نظرات آن‌ها را اعمال می‌کنند. در نهایت هم در یک فاز تک تکراری به ترخیص محصول می‌پردازند. نگهداری این نوع از فرآیندها معمولا از حالتی که در شکل ۹ دیدیم محدودتر است؛ همان طور که در تکرار ایجاد و ترخیص انعطاف کم‌تری از این دسته دیدیم در نگهداری هم انعطاف کم‌تر دارند و معمولا نگهداری آن‌ها به رفع مشکلات اندکی محدود می‌شود. بنابراین می‌توان این حالت را مشابه حالتی گرفت که در شکل ۱۱ دیده می‌شود.

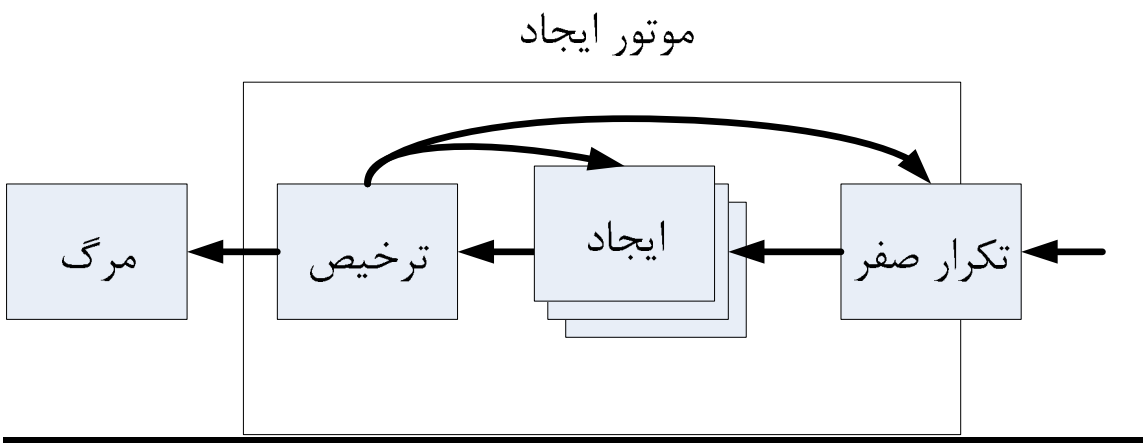


شکل ۱۰. بازارایی دوم فرآیند عمومی



شکل ۱۱. بازارایی سوم فرآیند عمومی

دید یک متدولوژی در فازهای مختلف آن تاثیر می‌گذارد. وقتی انعطاف آن کم است و ترخیص‌ها عمومی کم‌تری دارد یا اصولا ترخیص عمومی میانی ندارد، به همان دلیل در نگهداری هم کم فعالیت‌تر هستند. برخی از این متدولوژی‌ها سعی می‌کنند در تکرارهای موتور تولید تا می‌توانند کار را تمام کنند و کار کم‌تری برای نگهداری بگذارند. با این دید عملا شکل ۱۰ و ۱۱ خیلی به هم نزدیک می‌شوند. در عمل هم وقتی ما یک ترخیص عمومی داریم، در فاز نگهداری کار کم‌تری خواهد بود. با این دید احتمالا تعداد پروژه‌های سازمان بیش‌تر می‌شود چون برای نگهداری و اضافه کردن چیزهایی به سیستم باید یک پروژه‌ی جدید تعریف شود.



شکل ۱۲. بازارایی چهارم فرآیند عمومی

حالت نهایی که می‌توان متصور بود نیز در شکل ۱۲ ارائه شده است. در این حالت ترخیص‌های مکرر داریم ولی نگهداری نداریم. این روش در هیچ کدام از متدولوژی‌های مورد بررسی استفاده نشده است.

در این جا وابستگی سه فاز آخر که در بخش‌های ۳-۳ و ۳-۶ (زمانی که فاز ترخیص و فرآیند عمومی ارائه می‌شدند) به آن اشاره شد، کاملا مشخص می‌شود. دید متدولوژی مشخص می‌کند که این سه فاز چگونه‌اند. ترخیص‌های محدود، همراه با نگهداریِ کم حجم است (در عوض به دلیل تعریف پروژه‌ی جدید برای هر نگهداری تعدد پروژه‌های مشابه بالا می‌رود. پروژه‌ی نگهداری ممکن است به سازمانی دیگر واگذار شود). ترخیص‌های زیاد نیز با فاز نگهداری عمیق‌تر و طولانی‌تر همراه خواهد بود.

۴ - ۲. الگوی فرآیند فاز تکرار صفر

الگوی فاز تکرار صفر به همراه الگوهای مرحله‌ی تشکیل دهنده در شکل ۱۳ نشان داده شده است. این الگو شامل چهار الگوی فرآیند مرحله می‌باشد.

۴ - ۲ - ۱. برقرار کردن پروژه

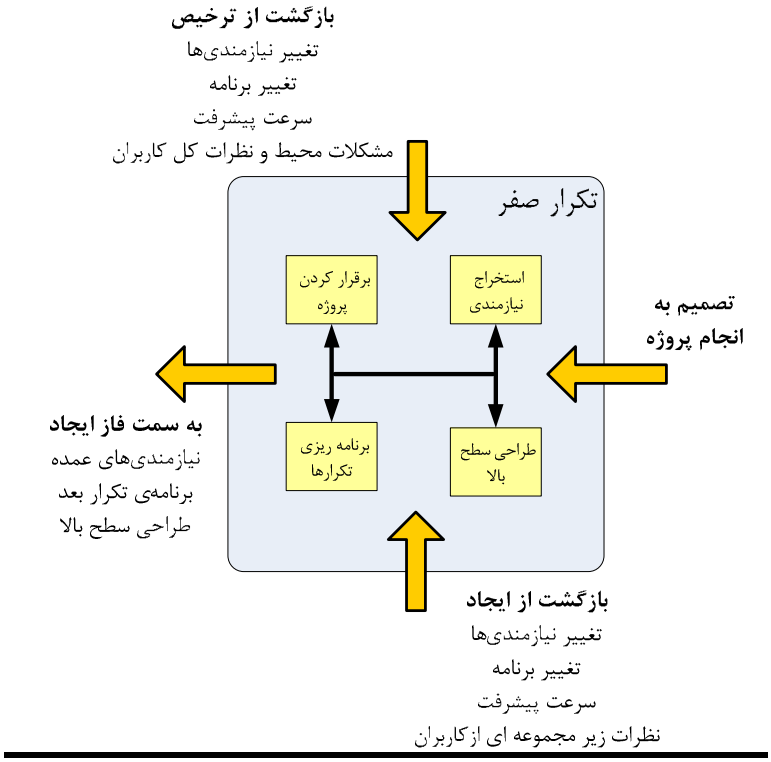
در برقرار کردن پروژه سعی ما این است که ملزومات اولیه‌ی انجام پروژه را فراهم کنیم. این الگوی فرآیند مرحله از فعالیت‌های زیر تشکیل شده است:

- کسب حمایت و تامین بودجه برای شروع به کار
- امکان سنجی انجام پروژه
- آماده‌سازی محیط و منابع برای شروع به کار تیم
- تشکیل شالوده‌ی تیم

۴ - ۲ - ۲. استخراج نیازمندی

استخراج نیازمندی با استفاده از خود ذینفعان در این مرحله اتفاق می‌افتد. فعالیت‌های زیر در این مرحله رخ می‌دهند:

- استخراج نیازمندی‌های سطح بالا معمولا به زبان کاربر
- کمک ذینفعان



شکل ۱۳. الگوی فرآیند فاز – تکرار صفر

۴ - ۲ - ۳. برنامه ریزی تکرارها

برنامه ریزی و تخمین تولید سیستم و تکرارها در این مرحله رخ می‌دهد. قسمتی از برنامه ریزی انجام شده برای کل پروژه است و بقیه‌ی برنامه ریزی‌ها برای هر کدام از تکرارها و یا چندین تکرار پشت سر هم صورت می‌گیرد. این مرحله شامل فعالیت‌های زیر است:

- تخمین و برنامه ریزی

- استفاده از پروتوتایپ برای شناخت بهتر نیازمندی‌ها
- اولویت بندی نیازمندی‌ها

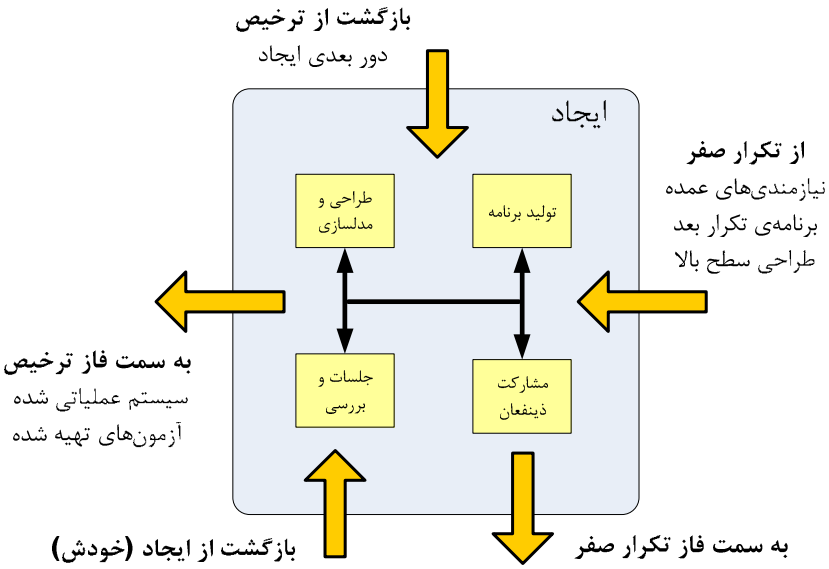
۴ - ۲ - ۴. طراحی سطح بالا

طراحی سطح بالا که به ما دید کلی از چگونگی به کار گیری سیستم می‌دهد شامل فعالیت‌هایی است:

- به دست آمدن معماری اولیه
- استفاده از پروتوتایپ برای بررسی انتخاب‌های مختلف

۴ - ۳. الگوی فرآیند فاز ایجاد

در الگوی فاز ایجاد ما به ساختن سیستم می‌پردازیم. این الگوی فاز شامل چهار الگوی مرحله است که در شکل ۱۴ قابل مشاهده هستند.



شکل ۱۴. الگوی فرآیند فاز - ایجاد

۴ - ۳ - ۱. تولید برنامه

در این مرحله به ساخت سیستم می‌پردازیم. این کار معمولا به صورت مشارکتی و با همکاری ایجاد کنندگان انجام می‌گیرد. فعالیت‌هایی زیر در این مرحله حضور دارند:

- تولید افزایشی و تدریجی به وسیله‌ی تکرارها
- همکاری تولید کنندگان

- قالب و سبک کد نویسی مشترک
- ریفکتور کردن و رفع کاستی‌ها
- آزمون مبنایی
- یکپارچه سازی در قالب یک کل

۴ - ۳ - ۲. مشارکت ذینفعان

ذینفعان در فرآیندهای چابک نقش بزرگی در تولید سیستم دارند. حضور آن‌ها در زمان تولید می‌تواند گره گشای ما باشد. تاثیر ذینفعان شامل موارد زیر است:

- کنترل تام ذینفعان بر برنامه‌ها و فرآیند
- نمایش برای کاربران درون سازمانی و گرفتن نظرات آن‌ها

۴ - ۳ - ۳. طراحی و مدل‌سازی

در متدولوژی‌های چابک مدل گریزی وجود دارد ولی در موارد لازم به تهیه‌ی مدل هرچند دور ریختنی مبادرت می‌ورزند. کارهای زیر مرتبط با این مرحله هستند:

- طراحی برنامه معمولا در جلسات
- مدل‌سازی کمی‌زودتر و دیدن کل

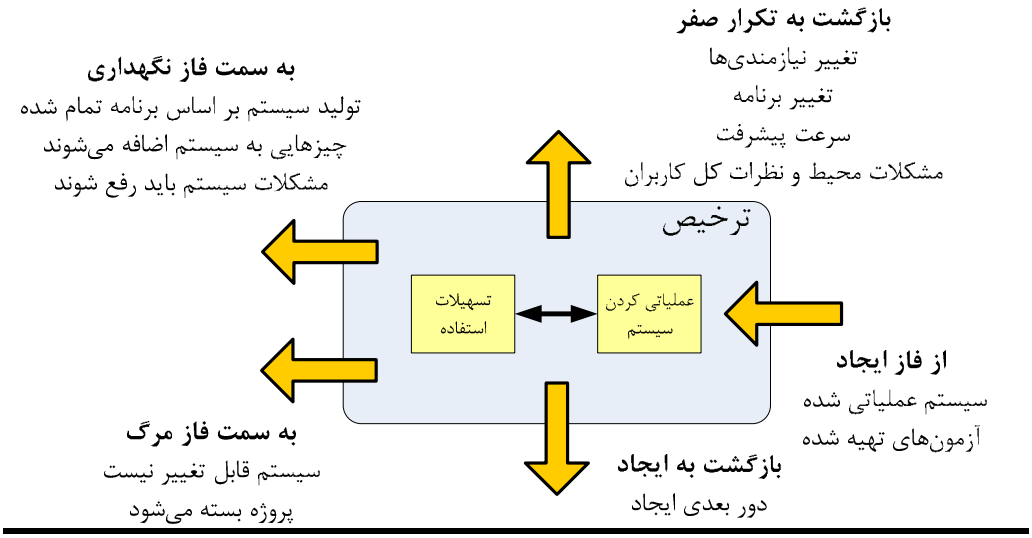
۴ - ۳ - ۴. جلسات و بررسی

تصمیم گیری ها در متدولوژی‌های چابک غالبا در جلسات و به صورت مشترک اخذ می‌شود. برخی هماهنگی‌ها و ارتباطات انسانی نیز در جلسات انجام می‌گیرند. در زیر کارهایی که در این جهت هستند را می‌بینیم:

- جلسات روزانه و ایستاده
- بررسی پیشرفت کار و فراگیری دروس
- تعیین قدم بعدی

۴ - ۴. الگوی فرآیند فاز ترخیص

در این فاز سیستم وارد محیط مشتری می‌شود و در عمل شروع به استفاده از آن می‌کنیم. در این فاز چیزهایی که مورد نیاز کاربر است تا بتواند بهتر با سیستم کار کند فراهم می‌شوند. این فاز همان طوری که در شکل ۱۵ دیده می‌شود شامل دو الگوی فرآیند مرحله است.



شکل ۱۵. الگوی فرآیند فاز - ترخیص

۴ - ۴ - ۱. عملیاتی کردن سیستم

سیستمی که در سازمان تولید کننده ایجاد شده باید وارد محیط واقعی شود. برای وارد کار شدن این سیستم باید عملیات‌هایی را انجام داد. در این مرحله سیستم آماده شده را برای کار کردن در محیط کاربر تنظیم می‌کنیم. فالیت‌های این مرحله عبارتند از:

- آزمون نهایی سیستم
- رفع کردن مشکلات یافته شده
- استقرار سیستم

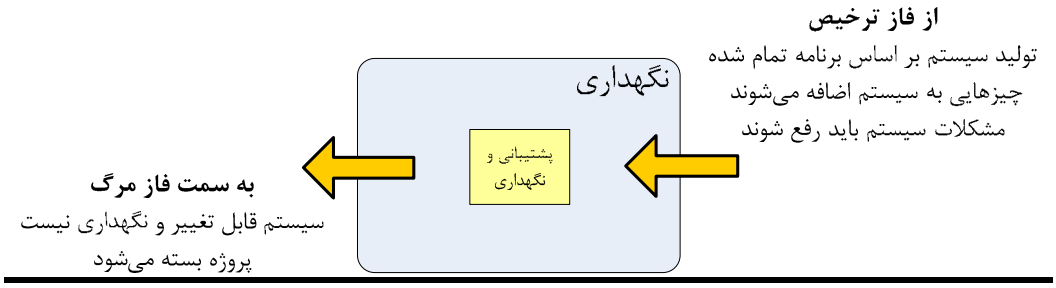
۴ - ۴ - ۲. تسهیلات استفاده‌ی بهتر

برای استفاده‌ی بهینه از سیستم باید در کنار آن کمک‌هایی را ارائه دهیم تا منحنی یادگیری کاربر سرعت بیش‌تری به خو بگیرد. در این مرحله فعالیت‌های زیر حضور دارند:

- بستن و نهایی کردن مستندات
- آموزش کاربران

۴ - ۵. الگوی فرآیند فاز نگهداری

به نظر تنها فعالیت و تنها الگوی مرحله‌ی این فاز پشتیبانی و نگهداری است. این فعالیت که در متدولوژی‌ها چابک محجور واقع شده و معمولا به خارج چرخه و تکرار آن رانده می‌شود، خود شامل فعالیت‌های زیادی است. از آن‌جا که هدف ما بیرون کشیدن مشترکات بین متدولوژی‌های چابک است و آن‌ها خود در مورد این فاز کم مطلب گفته‌اند، نمی‌توانیم از این بیش‌تر فاز نگهداری را غنی کنیم. شکل ۱۶ نشان دهنده‌ی این فاز است.



شکل ۱۶. الگوی فرآیند فاز - نگهداری

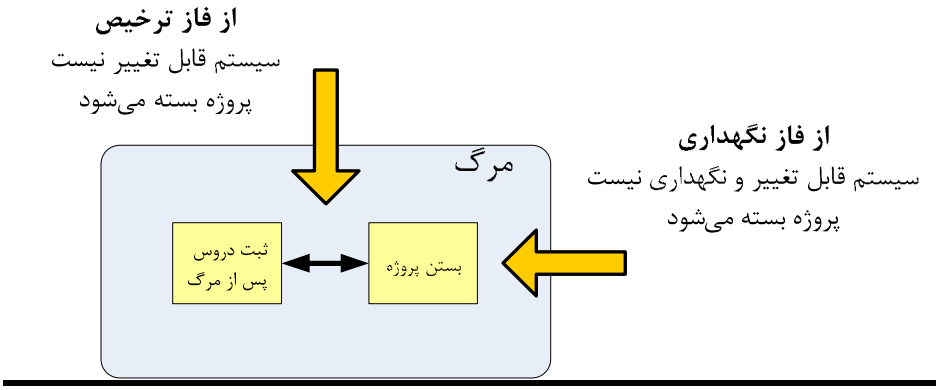
۴ - ۵ - ۱. پشتیبانی و نگهداری

خیلی از فعالیت‌هایی که در موتور تولید انجام می‌شوند عملا در این فاز نیز انجام خواهند شد و قابل تکرار در این جا هستند. تک فعالیت این مرحله همان‌طور که اشاره شد عبارت است از:

- پشتیبانی و نگهداری با استفاده از تکرار موتور تولید

۴ - ۶. الگوی فرآیند فاز مرگ

در این فاز به خداحافظی با سیستم و کارهای لازم می‌پردازیم. دو الگوی مرحله‌ی این فاز در شکل ۱۷ قابل مشاهده‌اند.



شکل ۱۷. الگوی فرآیند فاز - مرگ

۴ - ۶ - ۱. بستن پروژه

در این مرحله پروژه بسته می‌شود و رسما از سازمان خارج می‌شود. حالا می‌توانیم به جشن‌هایی که می‌توان گرفت فکر کنیم! این الگو تک فعالیتی است:

- بستن پروژه

۴ - ۶ - ۲. ثبت دروس پس از مرگ

در این مرحله به ثبت کلیه‌ی تجربیاتی که در پروژه کسب کردیم به صورت مدون می‌پردازیم. این اطلاعات در پروژه‌های بعدی و به فراخور نیاز استفاده خواهند شد. این مرحله نیز تک فعالیتی است:

- مستند سازی پس از مرگ

۴ - ۷. پاد الگوها

در متدولوژی‌های چابک فرضیات غلط و اشتباهی نیز مشاهده می‌شوند. این روش‌های اشتباه در متدولوژی‌های مختلف پیش گرفته شده‌اند. این روش‌های غلط تکرار شونده را پاد الگو[1] می‌نامیم. پاد الگوهای ارتباطات انسانی، مدل گریزی، نداشتن طراحی و ابزار محور بودن در مورد این متدولوژی‌ها شناخته شده است.

متدولوژی‌های چابک بر ارتباطات انسانی بسیار تکیه می‌کنند. این موضوع در این متدولوژی‌ها تعیین کننده است. در برخی موارد انتظاری که از روابط انسانی و انتقال اطلاعات با استفاده از آن می‌رود دچار زیاده روی می‌شود. بنابراین پایه‌ای بودن آن در متدولوژی نیز می‌تواند باعث بروز مشکلات در پروژه و عدم مقیاس پذیری آن شود.

تعداد زیادی از این متدولوژی‌ها از ترس افتادن در دام مدل سازی و فلج شدن در آنالیز[2] کلا این فعالیت‌ها را کنار گذاشته‌اند. این متدولوژی‌ها کمبود نداشتن یک درک مشترک در قالب طراحی و مدل‌ها را با هزینه‌های فراوانی که برای مدیریت و هماهنگ نگه داشتن تیم مصرف می‌کنند جبران می‌کنند.

این متدولوژی‌ها به دلیل نداشتن طراحی، مقیاس پذیری و بی‌درز بودن خود را از دست داده‌اند. منظور ما از طراحی، طراحی سطح برنامه نیست. طراحی میان تحلیل و پیاده‌سازی می‌نشیند و مدل‌های تحلیل را برای پیاده‌سازی آماده می‌کند و به گونه‌ای تبدیل یکنواخت این مدل‌ها به سطح بعدی (پیاده‌سازی) را ممکن می‌سازد.

ابزار محور بودن یکی دیگر از فرضیات این متدولوژی‌هاست. این فرض به خودی خود مشکلی ندارد اما وقتی افراطی شود وابستگی به ابزارها را زیاد می‌کند.

نتیجه گیری

متدولوژی‌های چابک ارائه شده مشترکات زیادی دارند. در این مستند توانستیم یک چرخه‌ی عمومی برای زیر مجموعه‌ای از این متدولوژی‌ها ارائه دهیم و مجموعه‌ای الگوی فرآیند از آن‌ها بیرون بکشیم. همان طور که از بخش‌های مستند حاکی است، تمرکز فعالیت‌های این متدولوژی‌ها در بخش‌های مربوط به ایجاد سیستم و موتور ایجاد بیش‌تر است و هرچه به انتهای پروژه نزدیک می‌شویم از تعدد فعالیت‌ها کاسته می‌شود. این امر منطقی به نظر می‌رسد؛ مدت زمانی که در فازهای مرتبط با موتور تولید سپری می‌شوند به مراتب بیش‌تر از زمانی است که

[1] Anti Pattern
[2] Analysis Paralysis

در فازهای دیگر فعالیت می‌کنیم و طبیعی است که تعداد فعالیت‌های فازهای وابسته به آن نیز بیش‌تر باشد. حتی گاهی در این متدولوژی‌ها فعالیت‌های انتهایی نیز وارد موتور تولید می‌شوند و توسط آن انجام می‌گیرند.

ما تمرکز خود را بر روی چرخه‌ی ایجاد سیستم گذاشتیم و فعالیت‌های درون فازهای این چرخه را بررسی کردیم. برخی فعالیت‌ها که در همه جای چرخه‌ی ایجاد نرم‌افزار قابل استفاده هستند در این نوشتار بررسی نشدند. نمونه‌ی این فعالیت‌ها را می‌توان مرور دانست که در هر سطحی از چرخه قابل استفاده است. این فعالیت‌ها از آن‌جا که در همه جا قابل استفاده هستند در تطابق متدولوژی‌ها کمکی به ما نمی‌کردند و اطلاعاتی نیز نمی‌افزودند.

در میان متدولوژی‌های بررسی شده به نظر می‌رسد متدولوژی XP با توجه به جدول‌های ۱ تا ۱۰ بیش‌ترین پوشش را بر فرآیند عمومی متدولوژی‌های چابک می‌دهد. مشاهده می‌کنیم این متدولوژی که ادعا می‌شد فرآیند ندارد بیش‌ترین تطابق را با فرآیند مطرح شده دارد. شاید به همین دلیل است که متدولوژی‌های چابک را با این متدولوژی می‌شناسند.

در این نوشتار تا حد الگوهای فرآیند مرحله پیش رفتیم و الگوهای فرآیند کار را بررسی نکردیم. این الگوها نیز در مورد فرآیندهای چابک قابل بررسی بودند. برای بررسی الگوهای فرآیند کار به بررسی دقیق‌تر متدولوژی‌های ذکر شده و با جزئیات بیش‌تری نیاز داریم. در ضمن زمانی که برای بررسی و ارائه‌ی این الگوها نیاز است بسیار بیش‌تر از زمان لازم برای الگوهای فرآیند مرحله خواهد بود.

منابع

[1] R. Ramsin, Object Oriented Methodologies course, lecture 8 to 15, Computer Engineering Department, Sharif University of Technology, http://sharif.edu/~ramsin/index_files/Page501.htm, last visited: 2007-01-07

[2] Scott W. Ambler, The Agile System Development Lifecycle (SDLC), http://www.ambysoft.com/essays/agileLifecycle.html, last visited: 2007-01-07

معادل انگلیسی عبارت‌های فارسی

اتمام بازی	End Game
ارتباط سیستم‌ها	System Interaction
اسپایک	Spike
استادِ اسکرام	Scrum Master
اسکرام	Scrum
اسکرام‌هایی از اسکرام‌ها	Scrums of Scrums
اعتبارسنجی	Continuous Validation
افزودنی	Increment
اکتشاف	Exploration
الگوهای فرآیند فاز	Phase Process Patterns
الگوهای فرآیند کار	Task Process Patterns
الگوهای فرآیند مرحله	Stage Process Patterns
امکان سنجی	Feasibility Study
انتقال	Transition
انتقال به ساخت	Transition to Construction
انعطاف‌پذیر	Flexible
ایجادِ تست مبنا	Test Driven Development (TDD)
آزمون بتا	Beta Test
آزمون مستمر	Continuous Test
آزمون مقبولیت	Acceptance Test
آزمون‌های یکپارچگی	Integration Test
آغاز	Inception
بحرانی بودن	Criticality
بر مبنای پردازش	Process-Oriented
بر مبنای کسب و کار	Business-Oriented
بررسی اعتبار و درستی سیستم	System-wide Verification and Validation
بررسی پس از تکرار	Post-iteration Revision

Business Study	بررسی کسب و کار
Chartering	برقرارکردن
Planning (Release Planning)	برنامه ریزی (برای ترخیص)
Planning	برنامه‌ریزی
Plan-Build-Revise	برنامه‌ریزی-ساخت-اصلاح
Plan-Design-Build	برنامه‌ریزی-طراحی-ساخت
Sprint Planning	برنامه‌ریزی اسپرینت
Plan By Feature (PBF)	برنامه‌ریزی با خصیصه
Iteration Planning	برنامه‌ریزی تکرار
Iteration Planning	برنامه‌ریزی تکرار
Adaptive Cycle Planning	برنامه‌ریزی منعطف چرخه
Coding	برنامه‌نویسی
Cyclic Program-Test-Integrate	برنامه‌نویسی-آزمون-یکپارچگی چرخشی
Post-Game	بَعدِ بازی
Product Backlog	بک‌لاگ محصول
Anti Pattern	پاد الگو
Prototyping	پروتوتایپ
Post-Project	پس از پروژه
Sponsor	پشتیبان
Implementation	پیاده‌سازی
Predictive	پیش‌بینی کننده
Reflect on the Delivery	تامل بر محصول
System Conversion	تبدیلات سیستم
Cyclic Delivery	تحویل چرخه‌ای
Deliver to Real Users	تحویل محصول به کاربران حقیقی
Release	ترخیص
System-wide Test	تست سطح سیستم
Final Q/A and Release	تضمین کیفیت نهایی و ترخیص
Evolutionary	تکاملی

Iteration 0	تکرار صفر
Iterations to first Release	تکرارها برای اولین ترخیص
Development Iterations	تکرارهای ایجاد
Release Iterations	تکرارهای ترخیص
Iterative Incremental	تکراری افزایشی
Recalibrate the Release Plan	تنظیم دوباره‌ی برنامه‌ی ترخیص
Build a Feature List	تهیه‌ی لیست خصیصه‌ها
Develop an Overall Model	تهیه‌ی یک مدل عمومی
Track	توالی
Development (Game)	تولید (بازی)
Sprint Development	تولید اسپرینت
Iterative Development	تولید تکراری
Develop in Iterations	تولید توسط تکرارها
Elaboration	جزئی‌شدن
Iteration Completion Ritual	جشن اتمام تکرار
Standup Meeting	جلسات ایستاده
Reflection	جلسات تامل
Model Storming	جلسات مدل‌سازی
Agile	چابک
Agile System Development Lifecycle	چرخه‌ی عمومی ایجاد توسط متدولوژی‌های چابک
Speculate-Collaborate-Learn	حدس-همکاری-یادگیری
Problem Domain	حوزه‌ی مساله
Wrap-up	خاتمه دادن
Feature	خصیصه
Project-Proper	خود پروژه
User Story	داستان کاربر
Vision/Charter	دید کلی
Stakeholders	ذینفعان
Alternative	روش پیشنهادی

Refactoring	ریفکتور کردن
Builds	ساخت
Construction	ساخت
Build By Feature (BBF)	ساخت با خصیصه
Prologue	ساختار درونی
Chief Programmer	سربرنامه‌نویس
Chief Architect	سرمعمار
Project Initiation	شروع پروژه
Project Initiation	شروع پروژه
Class Owner	صاحب کلاس
Project Data Sheet	صفحه‌ی اطلاعات پروژه
Design By Feature (DBF)	طراحی با خصیصه
Program Design	طراحی برنامه
Design and Build Iteration	طراحی و ساخت
Production	فرآوری
Activity	فعالیت
Analysis Paralysis	فلج شدن در آنالیز
Pre-Game	قبلِ بازی
Pre-Project	قبل پروژه
Step	قدم
User Story Index Card	کارت داستان کاربر
Index Card	کارت نمایه
Post-mortem Activities	کارهای پس از مرگ سیستم
Crystal	کریستال
Spiral	مارپیچی
Collective Code Ownership	مالکیت عمومی کد
Metaphor	متافور
Domain Expert	متخصص دامنه
Feature Set	مجموعه خصوصیات

Major Feature Set	مجموعه خصوصیات عمده
Time-Box	محدوده‌ی زمانی
Product ionizing	محصول سازی
Domain Model	مدل دامنه
Object Model	مدل موجودیت‌ها
Functional Model Iteration	مدل‌کردن عملیات
Model Phobia	مدل‌گریزی
Development Manager	مدیر ایجاد
Data Management	مدیریت داده‌ها
Death	مرگ
Sprint Review	مرور اسپرینت
Quality Review	مرور کیفیت
Deploy	مستقر کردن
Artifact	مصنوع
Architecture/High-level Design	معماری و طراحی سطح بالا
Tangible	ملموس‌تر
Concurrent Component Engineering	مهندسی هم‌زمان اجزا
Development Engine	موتور ایجاد
Support Component	مولفه‌های پشتیبانی
Technical Component	مولفه‌های تکنیکی
Feature Component	مولفه‌های خصوصیتی
Area	ناحیه
Maintenance	نگهداری
Ambassador User	نماینده‌ی کاربر
Just In Time (JIT)	هر موقع که لازم باشد
Highly Collaborative	همکاری شدید
User Interface	واسط کاربر
Architecture Strategy	یک تدبیر برای معماری
Integration	یکپارچگی

معادل فارسی عبارت‌های انگلیسی

Acceptance Test	آزمون مقبولیت
Activity	فعالیت
Adaptive Cycle Planning	برنامه‌ریزی منعطف چرخه
Agile	چابک
Agile System Development Lifecycle	چرخه‌ی عمومی ایجاد توسط متدولوژی‌های چابک
Alternative	روش پیشنهادی
Ambassador User	نماینده‌ی کاربر
Analysis Paralysis	فلج شدن در آنالیز
Anti Pattern	پاد الگو
Architecture Strategy	یک تدبیر برای معماری
Architecture/High-level Design	معماری و طراحی سطح بالا
Area	ناحیه
Artifact	مصنوع
Beta Test	آزمون بتا
Build a Feature List	تهیه‌ی لیست خصیصه‌ها
Build By Feature (BBF)	ساخت با خصیصه
Builds	ساخت
Business Study	بررسی کسب و کار
Business-Oriented	بر مبنای کسب و کار
Chartering	برقرارکردن
Chief Architect	سرمعمار
Chief Programmer	سربرنامه‌نویس
Class Owner	صاحب کلاس
Coding	برنامه‌نویسی
Collective Code Ownership	مالکیت عمومی کد
Concurrent Component Engineering	مهندسی هم‌زمان اجزا
Construction	ساخت
Continuous Test	آزمون مستمر

Continuous Validation	اعتبارسنجی
Criticality	بحرانی بودن
Crystal	کریستال
Cyclic Delivery	تحویل چرخه‌ای
Cyclic Program-Test-Integrate	برنامه‌نویسی-آزمون-یکپارچگی چرخشی
Data Management	مدیریت داده‌ها
Death	مرگ
Deliver to Real Users	تحویل محصول به کاربران حقیقی
Deploy	مستقر کردن
Design and Build Iteration	طراحی و ساخت
Design By Feature (DBF)	طراحی با خصیصه
Develop an Overall Model	تهیه‌ی یک مدل عمومی
Develop in Iterations	تولید توسط تکرارها
Development (Game)	تولید (بازی)
Development Engine	موتور ایجاد
Development Iterations	تکرارهای ایجاد
Development Manager	مدیر ایجاد
Domain Expert	متخصص دامنه
Domain Model	مدل دامنه
Elaboration	جزئی‌شدن
End Game	اتمام بازی
Evolutionary	تکاملی
Exploration	اکتشاف
Feasibility Study	امکان سنجی
Feature	خصیصه
Feature Component	مولفه‌های خصوصیتی
Feature Set	مجموعه خصوصیات
Final Q/A and Release	تضمین کیفیت نهایی و ترخیص
Flexible	انعطاف‌پذیر

Functional Model Iteration	مدل‌کردن عملیات
Highly Collaborative	همکاری شدید
Implementation	پیاده‌سازی
Inception	آغاز
Increment	افزودنی
Index Card	کارت نمایه
Integration	یکپارچگی
Integration Test	آزمون‌های یکپارچگی
Iteration 0	تکرار صفر
Iteration Completion Ritual	جشن اتمام تکرار
Iteration Planning	برنامه‌ریزی تکرار
Iteration Planning	برنامه‌ریزی تکرار
Iterations to first Release	تکرارها برای اولین ترخیص
Iterative Development	تولید تکراری
Iterative Incremental	تکراری افزایشی
Just In Time (JIT)	هر موقع که لازم باشد
Maintenance	نگهداری
Major Feature Set	مجموعه خصوصیات عمده
Metaphor	متافور
Model Phobia	مدل‌گریزی
Model Storming	جلسات مدل‌سازی
Object Model	مدل موجودیت‌ها
Phase Process Patterns	الگوهای فرآیند فاز
Plan By Feature (PBF)	برنامه‌ریزی با خصیصه
Plan-Build-Revise	برنامه‌ریزی-ساخت-اصلاح
Plan-Design-Build	برنامه‌ریزی-طراحی-ساخت
Planning	برنامه‌ریزی
Planning (Release Planning)	برنامه ریزی (برای ترخیص)
Post-Game	بَعدِ بازی

Post-iteration Revision	بررسی پس از تکرار
Post-mortem Activities	کارهای پس از مرگ سیستم
Post-Project	پس از پروژه
Predictive	پیش‌بینی کننده
Pre-Game	قبلِ بازی
Pre-Project	قبل پروژه
Problem Domain	حوزه‌ی مساله
Process-Oriented	بر مبنای پردازش
Product Backlog	بک‌لاگ محصول
Product ionizing	محصول سازی
Production	فرآوری
Program Design	طراحی برنامه
Project Data Sheet	صفحه‌ی اطلاعات پروژه
Project Initiation	شروع پروژه
Project Initiation	شروع پروژه
Project-Proper	خود پروژه
Prologue	ساختار درونی
Prototyping	پروتوتایپ
Quality Review	مرور کیفیت
Recalibrate the Release Plan	تنظیم دوباره‌ی برنامه‌ی ترخیص
Refactoring	ریفکتور کردن
Reflect on the Delivery	تامل بر محصول
Reflection	جلسات تامل
Release	ترخیص
Release Iterations	تکرارهای ترخیص
Scrum	اسکرام
Scrum Master	استادِ اسکرام
Scrums of Scrums	اسکرام‌هایی از اسکرام‌ها
Speculate-Collaborate-Learn	حدس-همکاری-یادگیری

Spike	اسپایک
Spiral	مارپیچی
Sponsor	پشتیبان
Sprint Development	تولید اسپرینت
Sprint Planning	برنامه‌ریزی اسپرینت
Sprint Review	مرور اسپرینت
Stage Process Patterns	الگوهای فرآیند مرحله
Stakeholders	ذینفعان
Standup Meeting	جلسات ایستاده
Step	قدم
Support Component	مولفه‌های پشتیبانی
System Conversion	تبدیلات سیستم
System Interaction	ارتباط سیستم‌ها
System-wide Test	تست سطح سیستم
System-wide Verification and Validation	بررسی اعتبار و درستی سیستم
Tangible	ملموس‌تر
Task Process Patterns	الگوهای فرآیند کار
Technical Component	مولفه‌های تکنیکی
Test Driven Development (TDD)	ایجادِ تست مبنا
Time-Box	محدوده‌ی زمانی
Track	توالی
Transition	انتقال
Transition to Construction	انتقال به ساخت
User Interface	واسط کاربر
User Story	داستان کاربر
User Story Index Card	کارت داستان کاربر
Vision/Charter	دید کلی
Wrap-up	خاتمه دادن