

# موسسه بابان

انتشارات بابان و انتشارات راهیان ارشد

درس و کنکور ارشد

## مهندسی نرم افزار

(متدولوژی های چابک)

ویژهی داوطلبان کنکور کارشناسی ارشد مهندسی IT

براساس کتاب مرجع

راجر اس. پرسمن هشتم ۲۰۱۴

## ارسطو خلیلی فر

چابکی در مقابل لختی و سستی قرار دارد. در یک بیان ساده فرز و چابک بودن در انجام پروژه‌های نرم‌افزاری به معنی سریع‌بودن در انجام کارها است، اما نه به معنی نادیده گرفتن معیارهای موفق پروژه‌های نرم‌افزاری! (برآورده شدن نیازمندی‌های مشتری، مقرون به صرفه بودن و در زمان مورد انتظار آماده شدن)

پشت مجلل‌ترین اتومبیل هم که باشی، اما چابک نباشی، تصادف ممکن است، مرگبار باشد. چابکی یک هنر است، هنر استفاده از امکانات موجود به عالی‌ترین شیوهی ممکن، هنر انجام صحیح یک راه‌حل، اما به شیوه‌ای هوشمندانه و پُرسرعت، بدون آنکه چیزی از قلم بیفتد. مهندسان نرم‌افزار، ربات نیستند، آن‌ها تفاوت و شباهت‌هایی دارند، بعضی از آن‌ها نسبت به بعضی دیگر چابک‌تر هستند، و این نشأت گرفته از ماهیت انسانی بودن آن‌ها است، برخی باهوش‌تر و سریع‌تر هستند و این کاملاً مشهود است. در یک بیان ساده، چابکی، کاتالیزور فرآیند تولید نرم‌افزار است. بنابراین، چابکی، مقدمه می‌خواهد. یعنی باید مقدماتی فراهم گردد تا پروژه چابک پیش رود. برای مثال سازنده و مشتری هر دو چابک باشند.

$$\boxed{\text{سازنده چابک}} + \boxed{\text{مشتری چابک}} = \boxed{\text{پروژه چابک}}$$

به طور کلی شرایط چابکی و مقدمات چابکی عبارتند از:

**مدل فرآیند تولید چابک:** این مدل باید براساس رویکرد تکرار و تکامل و مبتنی بر مؤلفه شیء‌گرا باشد تا سازنده و مشتری روزانه و پیوسته با یکدیگر در ارتباط باشند و همچنین سبب شود تا رضایت مشتری از طریق تحویل نسخه‌های محصول به طور پیوسته و سریع با بالاترین اولویت برآورده گردد. همچنین مدل فرآیند تولید نرم‌افزار، شامل فعالیت‌های چارچوبی به عنوان یک نقشه‌ی حرکت، جهت هدایت و پیشرفت پروژه استفاده می‌گردد، و اینکه هر یک از

فعالیت‌های چارچوبی شامل مراحل ارتباط، برنامه‌ریزی، مدل‌سازی (تحلیل و طراحی)، ساخت (پیاده‌سازی و تست) و استقرار به چه شکل و با چه حدی از چابکی انجام گردد، در چابک بودن روند پیشرفت پروژه مؤثر است و نه ادغام و ترکیب فعالیت‌های چارچوبی همچون ادغام فعالیت‌های طراحی و ساخت.

**روش چابک:** روش شیء‌گرایی به دلیل استفاده از مفاهیمی همچون کلاس به عنوان یک مؤلفه قابل استفاده مجدد، وراثت به عنوان یک عامل بازدارنده از دوباره‌نویسی کد، چندریختی به عنوان یک عامل بالابرنده خوانایی در برنامه، می‌تواند شکل و تمایل چابکی را به خود بگیرد. در روش ساخت‌یافته، چابکی خیلی کم‌تر دیده می‌شود.

توجه کنید که شیء‌گرایی و مفاهیم کلاس باعث می‌شود تا بتوان مؤلفه‌هایی با طراحی ایده‌آل با همبستگی بالا (Cohesion) و پیوستگی یا اتصال پایین (Coupling) در فعالیت طراحی مربوط به فعالیت‌های چارچوبی ایجاد کرد. هرچه قدر در طراحی مؤلفه‌ها همبستگی بالا و پیوستگی پایین باشد، فرآیند توسعه نرم‌افزار سریع‌تر خواهد بود و این یعنی چابکی.

**ابزار چابک:** UML، Nassi-Shneiderman diagram، Warnier-Orr Diagram، Petri net و

SysML به دلیل مدل‌سازی خاص و منحصر به فرد خود می‌توانند شکل و شمایل چابکی را به خود بگیرد. مانند تولید کد سریع از مدل‌های طراحی ایجاد شده توسط UML به کمک رشتال رز. همانطور که پیش‌تر نیز گفتیم، متدولوژی شیء‌گرا یا مهندسی نرم‌افزار شیء‌گرا نظامی است یکپارچه شامل مدل فرآیند شیء‌گرا (مدرن)، روش شیء‌گرا (مبتنی بر مفاهیم کلاس، وراثت و چندریختی) و ابزارهای شیء‌گرا که منجر به ایجاد نرم‌افزاری در بازه‌ی زمانی از قبل برنامه‌ریزی شده، بودجه‌ای از قبل پیش‌بینی شده و دقیقاً مطابق با نیازمندی‌های واقعی مشتری می‌گردد.

**توجه:** متدولوژی RUP متداول‌ترین نمونه از متدولوژی شیء‌گرا براساس روش شیء‌گرا (مبتنی بر مفاهیم کلاس، وراثت و چندریختی)، مدل فرآیند مبتنی بر مؤلفه‌ی شیء‌گرا با رویکرد تکرار و تکامل و ابزارهای شیء‌گرا (مثل ابزار مدل‌سازی UML و زبان برنامه‌نویسی C++) می‌باشد.

**توجه:** نسبت نمونه متدولوژی شیء‌گرای RUP به متدولوژی شیء‌گرا، مثل نسبت سیستم عامل ویندوز مدرن به مفاهیم مدرن سیستم عامل است.

اما متدولوژی شیء‌گرای چابک یا مهندسی نرم‌افزار شیء‌گرای چابک نظامی است یکپارچه شامل مدل فرآیند شیء‌گرای چابک (مدرن‌تر و سریع‌تر)، روش شیء‌گرای چابک (مبتنی بر مفاهیم کلاس، وراثت و چندریختی) و ابزارهای شیء‌گرای چابک که منجر به ایجاد نرم‌افزاری در بازه‌ی زمانی از قبل برنامه‌ریزی شده، بودجه‌ای از قبل پیش‌بینی شده و دقیقاً مطابق با نیازمندی‌های واقعی مشتری به شکلی چابک می‌گردد. این تعریف، فرآیند تولید پروژه‌های نرم‌افزاری را به

سمت موفقیت و چابکی سوق می دهد.

**توجه:** متدولوژی های XP<sup>۱</sup>، Scrum، Crystal، ASD<sup>۲</sup>، DSDM<sup>۳</sup> و AUP<sup>۴</sup> متداول ترین نمونه از متدولوژی های شیء‌گرای چابک براساس روش شیء‌گرای چابک (مبتنی بر مفاهیم کلاس، وراثت و چندریختی)، مدل فرآیند مبتنی بر مؤلفه‌ی شیء‌گرای چابک با رویکرد تکرار و تکامل و ابزارهای شیء‌گرای چابک (مثل ابزار مدل‌سازی UML، Nassi-Shneiderman diagram، Warnier-Orr Diagram، Petri net، SysML و زبان برنامه‌نویسی C++ و C#) می باشد.

**توجه:** نسبت نمونه متدولوژی های شیء‌گرای چابک XP، Scrum، Crystal، ASD، DSDM و AUP به متدولوژی شیء‌گرای چابک، مثل نسبت سیستم عامل ویندوز مدرن تر و سریع تر به مفاهیم مدرن تر و سریع تر سیستم عامل است.

متدولوژی های چابک از طریق یک فرآیند تکرار شونده، به تحویل زود هنگام نرم افزار در قالب افزایش های مختلف و جلب حداکثری رضایت مشتری می پردازند. هر افزایش یک نسخه اجرایی از نرم افزار است که در طی فرآیند نرم افزار به شکل تکرار شونده به مشتری تحویل می شود. با گذشت زمان افزایش های نرم افزار، کامل و کامل تر شده تا به شکل نرم افزار نهایی تبدیل شود. در رویکرد مبتنی بر تکرار و تکامل، فرصت یادگیری و بهبود تدریجی در سرتاسر چرخه‌ی تولید فراهم است. بدین ترتیب، در طول پروژه، امکان تصحیح به موقع اشتباهات وجود خواهد داشت. در صورت بروز اشتباه در یک تکرار، امکان جبران آن در تکرار بعدی وجود دارد. رویکرد مبتنی بر تکرار و تکامل به برنامه‌ریزی مستمر و پویا در طول پروژه نیازمند است. رویکرد مبتنی بر تکرار و تکامل در هر تکرار نسخه‌هایی از نرم افزار را ارائه می دهد که هر یک از قبلی کامل تر است. به بیان دیگر در مدل های تکاملی در هر دور از تکرار نسخه‌ی کامل تری از نرم افزار تولید می شود. در متدولوژی های چابک، تحویل سریع تر نرم افزار ترجیح بیشتری به انجام مراحل تحلیل و طراحی دارد، اگرچه به هیچ وجه نباید از انجام مراحل تحلیل و طراحی صرف نظر کرد. علت اصلی استفاده از متدولوژی های چابک این است که نرم افزارهای جدید مدام در حال تغییر و تحول هستند، علت این تغییرات مکرر، نیاز نرم افزار جهت تطابق با قوانین و نیازهای محیط عملیاتی خودش و یا پیشرفت تکنولوژی است. متدولوژی های چابک ضمن ارائه‌ی زود هنگام نرم افزار، نسبت به تغییرات نرم افزار نیز به دلیل رویکرد تکرار و تکامل بسیار منعطف است. در متدولوژی های چابک فعالیت های توسعه‌ی نرم افزار با جزئیات کمتری نسبت به متدولوژی های قدیمی تر انجام می شود. در واقع تنها وظایف ضروری حاضر در فعالیت های اصلی یا framework activity مورد توجه قرار گرفته و از انجام دیگر وظایف غیر ضروری چشم پوشی

<sup>1</sup> Extreme Programming

<sup>2</sup> Adaptive Software Development

<sup>3</sup> Dynamic System Development Method

<sup>4</sup> Agile Unified Process

می شود. به همین دلیل نام دیگری که برای متدولوژی های چابک پیشنهاد می شود، مهندسی نرم افزار سبک یا *software engineering lite* است.

در بحث مدیریت منابع انسانی و غیرانسانی نیز، این مدیریت باید چابک باشد، به عبارت دیگر پروژه های نرم افزاری به واسطه فعالیت های چارچوبی و فعالیت های چتری ایجاد می گردند. به بیان دیگر مطابق آنچه پیش از این نیز گفتیم نه تنها فعالیت های چارچوبی (فرآیند تولید نرم افزار) باید چابک باشد، بلکه فعالیت های چتری نیز باید چابک باشد، برای مثال یکی از فعالیت های چتری، پشتیبان گیری از منابع انسانی (مانند مدیران رده بالا) و غیرانسانی (مانند اطلاعات پروژه) است که باید به شکلی چابک و سریع و تا دیر نشده است انجام گردد.

همچنین در مورد تیم های متدولوژی های چابک باید گفت که این تیم ها به صورت خود سازمانده (Self-Organizing) تشکیل شده و اعضای آن ها با اختیارات بیشتری نسبت به دیگر تیم های نرم افزاری عمل می کنند. که این خود سازماندهی سبب چابکی تیم های کاری می گردد. تیم خودسازمانده خودش کنترل کارهایش را برعهده دارد. این تیم خودش به تعهداتش عمل می کند و طرح هایی برای انجام آنها تعریف می کند. هیچ چیز به این اندازه انگیزه را در یک تیم از بین نمی برد که آدم دیگری برایش تعیین تکلیف کند و هیچ چیز به اندازه پذیرش مسئولیت برای تعیین وظایف و انجام تعهدات در تیم ایجاد انگیزه نمی کند.

باز هم تأکید می کنیم که چابکی، مقدمه می خواهد، مقدمات چابکی باید فراهم گردد، تا پیشرفت پروژه چابک باشد، در یک بیان ساده، طوری تیم ها و کارها را سازماندهی کنید که همه چیز چابک و سریع باشد و عاملی نتواند روند رو به جلوی پیشرفت پروژه را مختل نماید، به هر دلیلی.

در اقتصاد نوین، پیش بینی چگونگی تکامل یافتن یک سیستم کامپیوتری، برای مثال یک برنامه ی کاربردی مبتنی بر وب، در گذر زمان، غالباً کاری دشوار است. شرایط بازار به سرعت تغییر می کند، نیازهای کاربران نهایی تکامل می یابد و تهدیدهای رقابتی بدون هشدار قبلی ظهور می کند. در بسیاری شرایط، قادر به تعریف کامل خواسته ها قبل از شروع پروژه نخواهید بود. باید به قدر کافی چابک باشید تا بتوانید به یک محیط عملیاتی متغیر پاسخ دهید.

تغییر، هزینه بردار است. به ویژه اگر کنترل نشده باشد یا مدیریت ضعیفی بر آن اعمال شود. یکی از بارزترین ویژگی های متدولوژی های چابک، توانایی آن در کاهش دادن هزینه های ناشی از تغییر در سرتاسر فرآیند نرم افزار است.

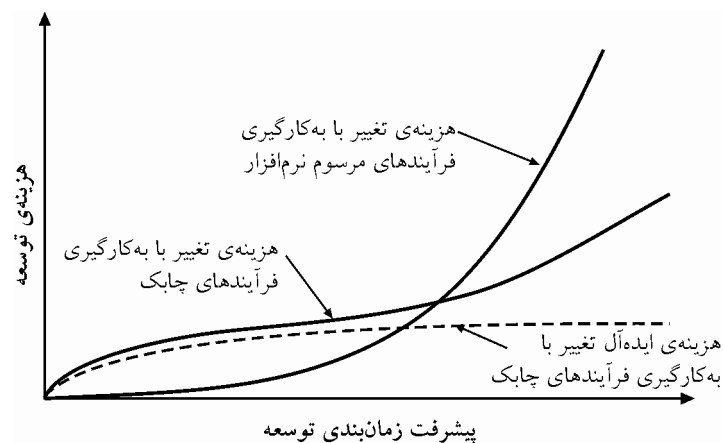
آیا این شرایط بدان معناست که شناخت چالش های پدید آمده از یک واقعیت جدید، باعث می شود که همه ی اصول، مفاهیم، روش ها و ابزارهای ارزشمند مهندسی نرم افزار را نادیده بگیرید؟ مطلقاً خیر! مهندسی نرم افزار نیز همانند کلیه رشته های مهندسی به تکامل خود ادامه می دهد و می توان آن را طوری تطبیق داد که چالش های ناشی از تقاضا برای سرعت را نیز پاسخگو باشد.

این روزها هنگام توصیف یک فرآیند نرم افزار مدرن تر، **چابکی** واژه ای است که به وفور به

گوش می‌رسد. تیم چابک تیمی فرزند چالاک است که قادر است به تغییرات پاسخ مناسب بدهد. تغییر چیزی است که در توسعه نرم‌افزار، بسیار با آن مواجه می‌شویم. تغییرات در نرم‌افزارهایی که در حال ساخته شدن هستند، تغییرات در اعضای تیم، تغییرات به دلیل تکنولوژی جدید، و یا هر نوع تغییری که ممکن است بر محصول در حال ساخت یا محصولی که محصول نهایی را ایجاد می‌کند، تاثیرگذار باشند. هرآنچه در نرم‌افزار انجام می‌دهیم باید خود حاوی ویژگی‌هایی برای پشتیبانی از تغییرات باشد، چیزی که قلب و روح نرم‌افزار است. تیم چابک می‌داند که نرم‌افزار توسط افرادی توسعه می‌یابد که در قالب تیمی کار می‌کنند و مهارت‌های این افراد و توانایی ایشان در همکاری، هسته اصلی موفقیت پروژه است. چابکی، بیشتر خاصیت خود را مرهون دانش نهفته در تیم است و نه در دانش نوشته شده روی کاغذ.

### چابکی و هزینه‌های تغییر

عقل سلیم در توسعه نرم‌افزار که چند دهه تجربه پشتیبان آن است، حکایت از آن دارد که هزینه تغییر به صورت غیرخطی با پیشرفت پروژه افزایش می‌یابد. پاسخگویی به تغییر در فعالیت ارتباطات یعنی هنگامی که تیم نرم‌افزاری مشغول جمع‌آوری خواسته‌هاست، نسبتاً آسان و با هزینه اندک است. و زمان مورد نیاز برای پاسخ به این نوع تغییرات تأثیری عمیق بر زمان‌بندی پروژه نخواهد داشت. اما اگر چند ماه جلوتر برویم، چطور؟ فرض کنید تیم در حال انجام فعالیت تست است، چیزی که نسبتاً در اواخر پروژه رخ می‌هد و ناگهان مشتری درخواست تغییری عمده در قابلیت‌های سیستم را دارد. این تغییر نیاز به اصلاح لیست نیازمندی‌ها، مدل تحلیل، مدل طراحی، پیاده‌سازی و تست دارد، برای مثال ساخت سه قطعه جدید و اصلاح و ویرایش پنج قطعه قدیم، و به تبع موردهای تست جدید نیز باید طراحی شوند. در این شرایط هزینه‌ها به سرعت بالا می‌رود و زمان لازم برای حصول اطمینان از اینکه تغییرات بدون برجای گذاشتن اثرات جانبی اعمال شود، قابل چشم پوشی نخواهد بود.



یک فرآیند چابک با رعایت اصول طراحی معماری ایده آل (پنهان‌سازی اطلاعات، افزایش انسجام، کاهش اتصال و استقلال عملیاتی) می‌تواند، هزینه تغییر را **تسطیح** کند. به این ترتیب، تیم نرم‌افزاری قادر به پاسخگویی به تغییرات در اواخر پروژه خواهد بود، بدون اینکه ضربه‌ای قابل ملاحظه از نظر زمان و هزینه به پروژه وارد آید. همانطور که پیش‌تر گفتیم، متدولوژی‌های چابک شامل تحویل افزایشی محصول می‌شود. هنگامی که تحویل افزایشی با سایر توصیه‌های چابک از قبیل تست واحد مستمر و برنامه‌نویسی زوجی (دو نفری بر روی یک کامپیوتر) تلفیق شود، زمان و هزینه اعمال تغییرات بیشتر هم کاهش می‌یابد.

### فرآیند چابک

فرآیند چابک باید از انطباق‌پذیری برخوردار باشد. ولی انطباق پیوسته و بدون پیشروی از موفقیت‌چندانی برخوردار نیست. بنابراین، در یک فرآیند نرم‌افزار چابک، باید روند انطباق به طور افزایشی انجام پذیرد. برای دستیابی به انطباق افزایشی و تدریجی، تیم نرم‌افزاری چابک نیاز به بازخورد از مشتری دارد، تا بتواند انطباق‌های لازم را انجام دهد. یک عامل شتاب‌دهنده به دریافت بازخورد مشتریان، نمایش نمونه‌ای اولیه از سیستم عملیاتی به مشتری است. از این رو، راهبرد توسعه افزایشی را باید نهادینه ساخت. نسخه‌های نرم‌افزار یعنی نمونه‌های اولیه قابل اجرا یا بخش‌هایی از سیستم عملیاتی باید در دوره‌های زمانی کوتاه مدت به مشتری تحویل داده شوند تا روند انطباق بتواند همگام با روند تغییرات ادامه یابد. مشتری به کمک این رویکرد مبتنی بر تکرار و تکامل می‌تواند، هر نسخه از نرم‌افزار را مرتباً ارزیابی کند، بازخوردهای لازم را برای تیم نرم‌افزاری فراهم سازد و بر انطباق‌های به عمل آمده جهت پاسخگویی به بازخوردها تأثیر بگذارد.

### اصول چابکی

در پیمان چابک (Agile Alliance) دوازده اصل برای کسانی که می‌خواهند به چابکی دست پیدا کنند، تعریف شده است.

- ۱- جلب رضایت مشتری از طریق تحویل زودهنگام و پیوسته نرم‌افزارهای ارزشمند، بیشترین اولویت را نزد ما دارد.
- ۲- پذیرا بودن تغییرات در خواسته‌ها حتی در اواخر فرآیند توسعه. که مزیتی رقابتی مابین شرکت‌های سازنده نرم‌افزار است.
- ۳- تحویل پیوسته‌ی نرم‌افزارهای کاری از دو هفته گرفته تا دو ماه، که بازه‌های زمانی کوتاه‌تر باید در اولویت قرار داده شوند.
- ۴- دست‌اندرکاران و افراد تجاری باید در سرتاسر پروژه هر روز باهم کار کنند.
- ۵- سپردن پروژه به افراد باانگیزه، فراهم‌سازی محیط و پشتیبانی مورد نیاز آنها و اطمینان‌کردن به آنها در انجام کارها.

- ۶- اثربخش ترین و موثرترین روش انتقال اطلاعات به درون و بیرون تیم توسعه، گفتگوی رو در رو است.
- ۷- نرم افزاری که به درستی کار کند، میزان اصلی در سنجش پیشرفت کاری است.
- ۸- فرآیندهای چابک، توسعه پایدار را ارتقا می بخشند. سازندگان و مشتریان باید قادر به حفظ سرعت ثابت در مسیر پیشرفت کار باشند.
- ۹- توجه پیوسته به اعتدالی فنی و طراحی معماری ایده آل (پنهان سازی اطلاعات، افزایش انسجام، کاهش اتصال و استقلال عملیاتی)، باعث افزایش چابکی می شود.
- ۱۰- ساده سازی ضروری است، یعنی هنر به انجام رساندن کارهای انجام ناپذیر.
- ۱۱- بهترین دستاورد از فعالیت های چارچوبی (ارتباطات، برنامه ریزی، مدل سازی، ساخت و استقرار) از تیم های خود سازمانده ظهور می کنند. به بیان دیگر بهترین تحلیل ها، طراحی ها، معماری ها و شناسایی نیازها از تیم های خود سازمانده منتج می شود.
- ۱۲- تیم در بازه های زمانی منظم، بازخوردی از میزان بهبود اثربخشی خود ارائه می دهد و سپس رفتار خود را مطابق این بازخورد تنظیم می کند.

این دوازده اصل در تمامی فرآیندهای چابک با وزن مساوی به کار برده نمی شوند و در برخی مدل ها از اهمیت یک یا چند اصل چشم پوشی می شود یا دست کم نقش آنها کمرنگ تر می شود. این اصول تعیین کننده هسته و جوهره چابکی هستند که در مدل های فرآیند چابک اغلب حفظ خواهند شد.

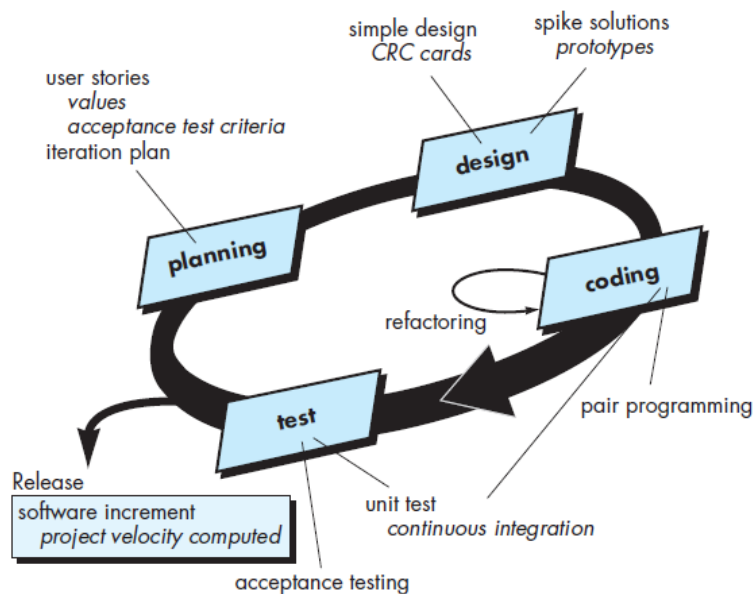
**توجه:** چابکی به معنی نادیده گرفتن مهندسی نرم افزار نیست. نباید بین چابکی و مهندسی نرم افزار یکی را انتخاب کرد، بلکه باید یک رویکرد مهندسی نرم افزار انتخاب شود که چابک هم باشد! اما روش شناسان سستی یک مُشت آدم های فاقد خلاقیت هستند که ترجیح می دهند مستندات بدون نقص تهیه کنند تا اینکه سیستمی کاری ارائه دهند که نیازهای مشتری را برآورده سازد. مکتب چابکی به معنی خروج از مکتب مهندسی نرم افزار نیست، بلکه اتفاقاً چابکی به معنی ورود و دقت بیشتر بر اصول اساسی و طراحی معماری ایده آل مهندسی نرم افزار است اما اینبار با سرعت و چابکی بیشتر. با در نظر گرفتن بهترین ایده ها از هر دو مکتب، بیشترین بهره عاید خواهد شد و چیزی از تخریب دیگری، به دست نخواهد آمد.

## متدولوژی XP

متدولوژی XP یا متدولوژی برنامه نویسی حدی (Extreme Programming) پرکاربردترین رویکرد در توسعه نرم افزار به شیوه چابک است. این متدولوژی با نام های XP و یا X-programming نیز شناخته می شود. علت نامگذاری این متدولوژی به برنامه نویسی حدی این است که نسبت به متدولوژی های دیگر، مرحله ی برنامه نویسی را با تاکید بیشتری انجام می دهد. متدولوژی XP از روش شیء گرا جهت توسعه برنامه استفاده می کند. در این متدولوژی فعالیت های



چارچوبی (framework activities) شامل چهار فعالیت برنامه‌ریزی، طراحی، برنامه‌نویسی و تست می‌باشد. به تازگی، شکل دیگری از XP موسوم به IXP (Industrial Extreme Programming) به معنی XP صنعتی پیشنهاد شده است. IXP پالایشی از XP است که فرآیند چابک را مشخصاً برای استفاده در سازمان‌های بزرگ هدف قرار داده است. شکل زیر گویای روند کارکرد متدولوژی XP است:



### برنامه‌ریزی (Planning)

فعالیت برنامه‌ریزی با گوش‌سپردن (listening) به خواسته‌های مشتری توسط سازنده طی جلساتی غیررسمی و شفاهی مابین سازنده و مشتری آغاز می‌شود. این جلسات منجر به بیان خواسته‌ها توسط مشتری و درک و جمع‌آوری خواسته‌ها توسط سازنده می‌شود. خواسته‌های شفاهی مشتری در قالب use case قرار می‌گیرند، سپس در ادامه برای هر use case (مورد کاربرد یا نیاز) سناریو یا شرح حال نوشته می‌شود.

سناریو بر دو طبقه اصلی و فرعی می‌باشد:

**سناریوی اصلی:** بیان روال حرکت قدم به قدم کارها داخل یک use case از نقطه شروع تا رسیدن به یک نتیجه مطلوب (موفق). (حرکت از خود موجود و رسیدن به خود مطلوب).

**سناریوی فرعی:** بیان روال حرکت قدم به قدم کارها داخل یک use case از نقطه شروع تا رسیدن به یک نتیجه نامطلوب (ناموفق). (حرکت از خود موجود و رسیدن به خود نامطلوب).

**توجه:** هر use case فقط و فقط یک سناریوی اصلی دارد و می تواند چندین سناریوی فرعی داشته باشد. سناریوها بر روی کارت شاخص (Index Card) نوشته می شوند.

در سناریوهای اصلی، خروجی ها و ورودی های نرم افزار، خصوصیات نرم افزار و کارکردهای نرم افزار مشخص می شود. مشتری باید برای هر سناریو اولویتی را مشخص کند. از آنجا که متدولوژی XP بر پایه رویکرد تکرار و تکامل است، این اولویت بندی سناریوها توسط مشتری، منجر به شناخت اولویت پیاده سازی سناریوها توسط سازنده در هر تکرار و افزایش می شود، ضمن اینکه در هر تکرار می توان سناریوهای جدیدی به پروژه اضافه نمود و یا سناریوهایی از پروژه حذف کرد. پس از درک سناریوهای مشتری توسط سازنده و اولویت بندی آنها توسط مشتری، سازنده هزینه و زمان لازم جهت پیاده سازی هر سناریو را بر حسب تعداد هفته های لازم مشخص می کند. اگر تعداد هفته های لازم جهت توسعه هر سناریو بیش از سه هفته بود، از مشتری خواسته می شود که سناریوی مورد نظر را به سناریوهای کوچکتر تقسیم کند و دوباره آنها را اولویت بندی کند. سازنده و مشتری با توافق یکدیگر سناریوهایی که در یک تکرار باید توسعه یابد را اولویت بندی و مشخص می کنند، اما ترتیب توسعه سناریوها در یک تکرار بر حسب شرایط پروژه بر اساس یکی از سه رویکرد زیر تعیین می شود:

- ۱- همه سناریوهای موجود در یک تکرار پیاده سازی شود.
- ۲- برخی از سناریوهای موجود در یک تکرار پیاده سازی شود، به این شکل که ابتدا سناریوهایی با اولویت بالاتر پیاده سازی شوند. بنابراین اگر یک تکرار یا افزایش تحت فشار زمان بندی قرار گرفت، این مزیت وجود دارد که سناریوهای پراولویت تر پیاده سازی شده اند.
- ۳- برخی از سناریوهای موجود در یک تکرار پیاده سازی شود، به این شکل که ابتدا سناریوهایی که ریسک بیشتری دارند پیاده سازی شوند. بنابراین اگر یک تکرار یا افزایش تحت فشار زمان بندی قرار گرفت، این مزیت وجود دارد که سناریوهای پرریسک تر پیاده سازی شده اند.

همانطور که گفتیم، متدولوژی XP بر اساس رویکرد تکرار و تکامل پیش می رود، بنابراین اندازه گیری های انجام شده بر روی سناریوهای انجام شده در تکرار و افزایش قبلی، مبنایی برای کار برنامه ریزی شامل فعالیت های «برآورد میزان کار»، «برآورد زمان لازم برای انجام کار»، «برآورد هزینه لازم برای انجام کار»، «مدیریت ریسک» و «زمان بندی» در افزایش و تکرار بعدی است. آینده نزدیک، بسیار شبیه به گذشته نزدیک است.

### طراحی (Design)

طراحی در متدولوژی XP برپایه ی سادگی بنا شده است، این اصل که سادگی را حفظ کن (keep it simple). در متدولوژی XP همواره یک طراحی ساده بر یک طراحی پیچیده برتری دارد.

در این متدولوژی صرفاً طراحی سناریوهایی انجام می‌شود که نیاز وضع موجود است و قرار به پیاده‌سازی قطعی آن است، بنابراین طراحی سناریوهایی که نیاز به آن در آینده محتمل است و قرار به پیاده‌سازی آن قطعی نیست، در این متدولوژی جایگاهی ندارد. متدولوژی XP در جهت حفظ سادگی در فعالیت طراحی خود فقط از کارت‌های CRC استفاده می‌کند. پس از شناسایی موارد کاربرد و سناریونویسی برای هر یک از موارد کاربرد، زمان تعریف کلاس‌ها، صفات، متدها و ارتباطات میان کلاس‌های همکار برای هر یک از موارد کاربرد می‌رسد. برای شناسایی کلاس‌ها، صفات، متدها و ارتباطات میان کلاس‌ها برای هر یک از موارد کاربرد، از تکنیکی موسوم به CRC که سرواژه‌ی عبارت Class – Responsibility Collaborator و به معنی «مدل همکاری مسئولیت‌های کلاس‌ها» می‌باشد، استفاده می‌شود. مدل‌سازی CRC روشی ساده جهت تعیین و سازماندهی کلاس‌های داخل هر مورد کاربرد است. اگر در بخشی از طراحی یک سناریو، سختی ایجاد شود، متدولوژی XP، ایجاد فوری یک نمونه اولیه عملیاتی (operational prototype) را برای آن بخش از طراحی توصیه می‌کند. استفاده از نمونه اولیه عملیاتی که موسوم به راهکار خیزشی (spike solution) نیز می‌باشد، دو پیامد دارد، اول اینکه، این راهکار منجر به کاهش ریسک فنی در پیاده‌سازی واقعی سناریوی مورد نظر می‌شود و دوم اینکه، این راهکار منجر به اعتبارسنجی برآوردهای اولیه یعنی «برآورد میزان کار»، «برآورد زمان لازم برای انجام کار»، «برآورد هزینه لازم برای انجام کار»، «مدیریت ریسک» و «زمان‌بندی» مربوط به سناریوی مورد نظر می‌شود.

### کدنویسی (Coding)

پس از کشف سناریوهای مشتری و انجام‌شدن کارهای طراحی مقدماتی، تیم توسعه به کدنویسی نمی‌پردازد، بلکه یک سری «آزمون واحد» تهیه می‌کند تا هر یک از سناریوهایی که قرار است در نسخه جاری یعنی تکرار فعلی گنجانده شود، مورد تست و بررسی قرار گیرد. هنگامی که آزمون واحد تهیه شد، سازنده بهتر می‌تواند توجه خود را به آن چیزی معطوف کند که باید پیاده‌سازی شود تا آزمون را با موفقیت پشت سر بگذارد. هنگامی که کدها کامل شدند، می‌توان تست واحدی را بلافاصله انجام داد و در نتیجه، بازخوردی فوری را حاصل کرد.

یکی از اصلی‌ترین مفاهیم متدولوژی XP، برنامه‌نویسی زوجی (pair programming) است که در مرحله کدنویسی قرار دارد. متدولوژی XP توصیه می‌کند که برای تولید کد هر سناریو، دو نفر از اعضای تیم توسعه بر روی یک کامپیوتر اقدام به کدنویسی نمایند. زیرا دو فکر غالباً بهتر از یک فکر است، مانند حضور مؤلف، هنگام تایپ یک کتاب، که به کشف زود هنگام خطاها کمک می‌کند. در واقع زمانی که برنامه توسط یکی از برنامه‌نویسان نوشته می‌شود، برنامه‌نویس دیگر در همان لحظه کد را با هدف کشف خطاهای آن بازبینی (review) می‌کند. که به آن تضمین کیفیت بی‌درنگ نیز گفته می‌شود.

به موازاتی که برنامه‌نویسان زوجی کار خود را کامل می‌کنند، کدی که می‌نویسند در کنار کار دیگران قرار داده می‌شود. در برخی موارد، این کار به صورت روزانه توسط تیم یکپارچه‌کننده (جامعیت) انجام می‌شود. در موارد دیگر، برنامه‌نویسان زوجی، مسئولیت جامعیت تدریجی (افزایشی) را نیز برعهده دارند. این راهبرد یعنی «جامعیت تدریجی پیوسته» به پرهیز از مشکلات سازگاری و ایجاد واسط کمک می‌کند و یک محیط تست دود (smoke testing) فراهم می‌سازد که به کشف زودهنگام خطاها کمک می‌کند.

همانطور که پیش‌تر گفتیم، طراحی در متدولوژی XP برپایه‌ی سادگی بنا شده است، بنابراین این دغدغه وجود دارد که گاهی طراحی‌های XP از کیفیت الگوریتمی (غیروظیفه‌مندی یا مزه) مناسبی برخوردار نباشد. متدولوژی XP برای مرتفع نمودن این دغدغه، روش فاکتورگیری مجدد یا بازآرایی (refactoring) را پیشنهاد می‌کند. بنابراین با وجود اینکه روش فاکتورگیری مجدد در مرحله کدنویسی انجام می‌شود اما ماموریت آن ارتقاء کیفیت الگوریتمی و خوشمزه‌سازی طراحی است. فاکتورگیری مجدد یا بازآرایی، فرآیند تغییر، بهبود و ارتقاء کیفیت کد است، بازآرایی، کد را خواناتر می‌کند، بازآرایی، کد را تغییرپذیرتر می‌کند، بازآرایی، ساختار داخلی کد را بهبود می‌دهد اما بدون آنکه بازآرایی، رفتار بیرونی کد را اصلاح کند، به عبارت دیگر بازآرایی، رفتار بیرونی کد را اصلاح نمی‌کند یعنی ورودی‌ها و خروجی‌های کد را تغییر نمی‌دهد.

بازآرایی، یک روش مناسب جهت اصلاح، ساده‌سازی و بهبود کد است. این کار احتمال وجود خطا در کد را کاهش می‌دهد. واضح است که عمل بازآرایی پس از تولید کد باید انجام شود. بازآرایی منجر به این می‌شود که علاوه بر بهبود ساختار داخلی کد، فعالیت طراحی نیز در طول فعالیت کدنویسی دستخوش تغییر، تحول و بهبود شود. یعنی هم قبل از کدنویسی و هم بعد از کدنویسی طراحی در حال انجام است. شعار طراحی قبل از کدنویسی «سادگی» است، اما شعار طراحی بعد از کدنویسی «کیفیت» است. و این کدنویسی است، که راهنمایی لازم برای چگونگی بهبود بخشیدن به طراحی را فراهم می‌سازد.

### تست (Testing)

شروع زودهنگام و غیررسمی آزمون واحد و آزمون جامعیت (یکپارچگی) در مرحله کدنویسی است، اما ادامه آن و شروع رسمی آزمون واحد، آزمون جامعیت (یکپارچگی) و آزمون اعتبارسنجی در مرحله تست است. آزمون واحد **صحت عملکرد یا روند اجرای** یک بخش، یک جزء یا یک مولفه از نرم‌افزار را مستقل از سایر مولفه‌های دیگر برنامه مورد تست و ارزیابی قرار می‌دهد. آزمون جامعیت یا یکپارچه‌سازی، اعتبارسنجی عملکرد یا روند اجرای چندین مولفه در کنار یکدیگر و یا کل مولفه‌های نرم‌افزار ایجاد شده در کنار یکدیگر به همراه **نقاط اتصال** مابین آن‌ها را مورد تست و ارزیابی قرار می‌دهد. در یک نرم‌افزار ممکن است هر یک از مؤلفه‌های برنامه به تنهایی درست عمل کنند اما ترکیب آنها با یکدیگر به درستی عمل نکند. زیرا ممکن است به دلیل

عدم تنظیم درست نقاط اتصال و ارتباطی مابین مولفه‌ها، این مولفه‌ها قادر نباشند در کنار یکدیگر، کارکرد مورد انتظار را نشان دهند. بنابراین آزمون جامعیت، جهت تست تدریجی یکپارچه‌شدن مولفه‌های برنامه در کنار یکدیگر مورد استفاده قرار می‌گیرد. منظور از نقاط اتصال مابین مولفه‌های مرتبط با هم، همان کانال یا محل تبادل داده مابین مولفه‌های مرتبط با هم هست، مانند فراخوانی با مقدار یا فراخوانی با ارجاع. اگر این نقاط اتصال مابین مولفه‌های مرتبط با هم درست تنظیم نشوند ممکن است داده‌ها در گذر از این نقاط اتصال از بین بروند مانند حالتی که یک متغیر دوبایتی به یک متغیر یک بایتی در روش فراخوانی با مقدار پاس داده می‌شود، در واقع در این حالت بخشی از داده‌ها از دست رفته است. این نقاط اتصال به واسطه مولفه نیز موسوم است. واضح است که، برطرف کردن مشکلات کوچک در هر چند ساعت یکبار، نسبت به برطرف کردن مشکلات بزرگ درست قبل از پایان مهلت، زمان کمتری می‌برد، بنابراین بهتر است آزمون جامعیت به شکل روزانه و به روش آزمون دود انجام شود. پس از انجام بازآرایی کد (refactoring) نیز می‌بایست آزمون جامعیت به روش آزمون رگرسیون انجام شود. آزمون اعتبارسنجی، اعتبارسنجی عملکرد یا روند اجرای کل مولفه‌های نرم‌افزار ایجاد شده در کنار یکدیگر را در شرایط **سهل و آسان** مورد تست و ارزیابی قرار می‌دهد. هنگامی که یک **نرم‌افزار سفارشی** تنها برای یک مشتری توسعه می‌یابد، «آزمون پذیرش»<sup>۱</sup> اجرا می‌شود تا مشتری را قادر به اعتبارسنجی کلیه خواسته‌ها کند. آزمون پذیرش، که به جای مهندس نرم‌افزار، توسط مشتری انجام می‌شود، می‌تواند از یک آزمون غیررسمی تا یک سری آزمون‌های برنامه‌ریزی شده سیستماتیک را شامل شود. در واقع، آزمون پذیرش را می‌توان در عرض یک هفته یا یک ماه انجام داد و لذا کشف خطاهایی که ممکن است سیستم را با گذشت زمان تنزل دهد، امکان‌پذیر می‌شود. در متدولوژی XP، آزمون اعتبارسنجی توسط آزمون پذیرش انجام می‌شود. مبنای آزمون پذیرش توسط مشتری، کنترل کارکرد همان سناریوهایی است که توسط مشتری در هر تکرار یا افزایش بیان شده‌است.

### ارزش‌های متدولوژی XP

در متدولوژی XP پنج ارزش تعریف شده است که مبنایی برای انجام صحیح همه کارهای موجود در متدولوژی XP است. هرکدام از این ارزش‌ها به عنوان محرکه‌ای برای فعالیت‌ها، کنش‌ها و وظایف XP به کار می‌رود، این ارزش‌ها به صورت زیر است:

- ۱- **ارتباطات (communication):** برقراری ارتباطات نزدیک به شکل رو در رو و غیررسمی مابین سازنده و مشتری جهت مشخص‌سازی خواسته‌های دقیق و واضح مشتری و پرهیز از برقراری ارتباط با مشتری برپایه تهیه مستندات پرحجم. هدف اصلی این ارتباطات شناخت نیازها و تعیین سناریوهای مشتری (customer stories) است.

<sup>1</sup> Acceptance Test

۲- **سادگی (Simplicity):** متدولوژی XP برای دستیابی به سادگی، به سازندگان توصیه می‌کند که فقط نیازهای فعلی بررسی و طراحی شوند و بررسی و طراحی نیازهای آینده به آینده واگذار شود و به حال آورده نشود. طراحی نیز باید ساده باشد و اگر نیاز به بهبود بود، بازآرایی می‌شود.

۳- **بازخورد (Feedback):** در متدولوژی XP تیم توسعه باید توجه ویژه‌ای به بازخورد محصول، مشتری و حتی خود تیم توسعه داشته باشد. با ایجاد هر کلاس مربوط به یک use case آزمون واحد بر روی هر کلاس مطابق قابلیت‌های مورد انتظار انجام می‌شود و به تدریج کارکرد کلاس‌های همکار درون یک use case در کنار هم تحت بررسی آزمون جامعیت (یکپارچگی) قرار می‌گیرد. در نهایت نیز آزمون اعتبارسنجی توسط آزمون پذیرش بر روی نحوه پیاده‌سازی سناریوهای موجود در use case‌های تحویلی در هر افزایش انجام می‌شود. همچنین در هنگام برنامه‌ریزی افزایش بعدی، بازخورد برنامه‌ریزی افزایش قبلی مورد استفاده قرار می‌گیرد.

۴- **شجاعت (Courage) یا انضباط (discipline):** پایبندی سفت و سخت به برخی جنبه‌های متدولوژی XP به جسارت و جرات نیاز دارد. یک واژه‌ی بهتر می‌تواند انضباط باشد. اغلب تمایل و فشار برای انجام طراحی نیازهای آینده وجود دارد. اکثر تیم‌های نرم‌افزاری هم سر تسلیم فرود می‌آورند، با این استدلال که طراحی برای آینده در بلندمدت منجر به صرفه‌جویی در زمان و تلاش می‌شود. اما تیم XP چابک باید انضباط و شجاعت لازم برای طراحی در امروز و زمان حال را داشته باشد. و در عین بداند که خواسته‌های آینده ممکن است به طرز چشمگیری تغییر کند که طراحی و پیاده‌سازی آنها در زمان حال منجر به دوباره‌کاری و اتلاف زمان و هزینه‌ها می‌گردد.

۵- **احترام (Respect):** تیم چابک، با محقق کردن هریک از ارزش‌های فوق، احترام متقابل میان سازنده و مشتری را برقرار می‌کند. با تحویل موفق هریک از نسخه‌های نرم‌افزار این احترام و اعتقاد بر موثر بودن متدولوژی چابک XP بیشتر و بیشتر نیز می‌شود.

## متدولوژی Scrum

نام متدولوژی Scrum از بازی راگبی گرفته شده است. راگبی ورزش گروهی است که در یک زمین چمن مستطیل شکل با دروازه‌های H شکل در دو سوی زمین و با یک توپ بیضی شکل بازی می‌شود. اسکرام در واقع یک شروع دوباره در بازی راگبی است به طوری که وقتی خطایی در این بازی روی می‌دهد یا اینکه توپ از زمین خارج می‌گردد از روش اسکرام برای آغاز دوباره بازی استفاده می‌شود. به این شکل که همه بازیکنان دور هم جمع می‌شوند، سرهایشان را پایین می‌گیرند و شروع مجدد می‌کنند، اسکرام در بازی راگبی بسیار اتفاق می‌افتد. اسکرام یک روش گروهی برای تولید و توسعه نرم‌افزار است. اینکه در هریک از فعالیت‌های

چارچوبی شامل ارتباط، برنامه‌ریزی، تحلیل، طراحی، پیاده‌سازی، تست و استقرار چه وظایفی باید انجام شود، توسط sprintها مشخص می‌شود که در ادامه به آن می‌پردازیم.

### نقش‌های اسکرام (scrum roles)

- **اسکرام مستر (scrum master)** : رهبر اسکرام وظیفه دارد تا تمامی اعضای تیم را هدایت و راهنمایی نماید تا هیچ یک از اعضای تیم از چارچوب و قوانین اسکرام خارج نشوند. رهبر اسکرام نقش مدیر را ندارد بلکه تنها وظیفه رهبری تیم را بر عهده دارد تا با رفع مشکلات و موانع پیش رو، در صورتی که اعضای تیم قادر به رفع موانع نباشند، اجرای اسکرام را بهبود بخشد.
- **صاحب محصول (product owner)** : صاحب محصول که نماینده ذینفعان (Stakeholders) پروژه و business است، با اعلام دقیق نیازمندی‌های خود به تیم تولید، با رهبر اسکرام و تیم تولید همکاری می‌نماید. صاحب محصول باید به سوالات تیم تولید پاسخ داده و همواره در دسترس باشد.
- **تیم تولید و توسعه نرم‌افزار (development team)** : افراد این تیم در چارچوب قوانین اسکرام، به تولید آن چه که صاحب محصول درخواست کرده است، می‌پردازند. تعداد اعضای تیم تولید نه باید آن قدر کم باشد که همکاری گروهی و کار تیمی بی‌معنا شود و نه آن قدر زیاد باشد که هماهنگی بین اعضای تیم تبدیل به امری دشوار و وقت‌گیر گردد. تعداد اعضای تیم تولید، بستگی به پروژه دارد اما معمولاً ۶ تا ۹ نفر اعضای این تیم را تشکیل می‌دهند.

### روند کارکرد اسکرام

هسته اصلی اسکرام را sprintها تشکیل می‌دهند. در متدولوژی‌های تکرارشونده (iterative) دوره‌های زمانی تکراری (iteration) وجود دارد که در این دوره‌ها به تدریج محصول کامل می‌گردد. بدین صورت که در تولید یک محصول، تعدادی تکرار در نظر گرفته می‌شود که در پایان دوره‌ی زمانی هر تکرار، یک محصول قابل ارائه وجود دارد. به این دوره‌های زمانی تکرارشونده در اسکرام sprint می‌گویند. در پایان هر sprint، محصول کامل‌تر شده و در نهایت محصول نهایی تولید می‌گردد. هر sprint دارای تعریفی است که در آن باید مشخص شده باشد که چه چیزی قرار است ساخته شود، نیازمندی‌ها، راهنمای ساخت و محصول خروجی نیز باید مشخص باشند. نتیجه اینکه مثل تمام متدولوژی‌های incremental و iterative در اسکرام نیز دوره‌های زمانی یا iteration داریم که در طی آنها محصول نهایی پروژه به تدریج تکمیل می‌شود.

در طی یک sprint که معمولاً یک دوره دو تا چهار هفته است یعنی حداکثر ۳۰ روز (طول دوره را تیم مشخص می‌کند) اعضا، یک محصول قابل ارائه و قابل استفاده را تدریجاً تولید

می کنند.

اصطلاح Product Backlog نامی است که به لیست نیازمندی های وظیفه مندی و غیروظیفه مندی کل یک پروژه اطلاق می شود و در حقیقت مجموعه ای اولویت بندی شده از نیازمندی های مشتری است که در نهایت بایستی تحویل داده شود.

مواردی از Product Backlog که در طی یک sprint بایستی انجام شود، در طول جلسه ی طراحی sprint یا Sprint Planning Meeting مشخص می شود. در طول این جلسه، صاحب محصول (Product Owner)، تیم تولید و توسعه نرم افزار (development team) را درباره ی مواردی از Product Backlog آگاه می کند. سپس اعضای تیم تولید مشخص می کنند که چه مقدار از موارد مشخص شده توسط Product Owner را می توانند در این sprint انجام دهند و چه میزان از آنرا در sprint های بعدی.

مواردی از Product Backlog که قرار است در یک Sprint انجام شود را اصطلاحاً Sprint Backlog می نامند. مفاد Sprint Backlog در واقع توافقی است بین اعضای تیم تولید و Product Owner، که بعد از تصویب شدن مفاد یک sprint، دیگر هیچ کس نمی تواند آنرا در طول sprint تغییر دهد. در طول دوره، نیازمندی های لحاظ شده در Sprint Backlog از Product Backlog حذف می شوند. اما امکان دارد به دلایلی که در ادامه می آید این نیازمندی ها دوباره به Product Backlog برگردد.

در ساده ترین روش معمولاً از نرم افزارهای صفحه گسترده (Spread Sheet) همچون Excel Microsoft برای ساختن و نگهداری Product Backlog و Sprint Backlog استفاده می شود، اما می توان از طیف وسیعی از ابزارهای نرم افزاری که برای استفاده در تیم های Agile نوشته شده اند نیز استفاده کرد.

مانند تمام متدولوژی های iterative، توسعه ی نرم افزار در اسکرام نیز Time Boxed است، به این معنی که sprint بایستی دقیقاً سر وقت تمام شود و اگر نیازمندی های تعیین شده در Sprint Backlog به هر علتی تکمیل نشده باشند آنها را کنار گذاشته و دوباره وارد Product Backlog می کنند.

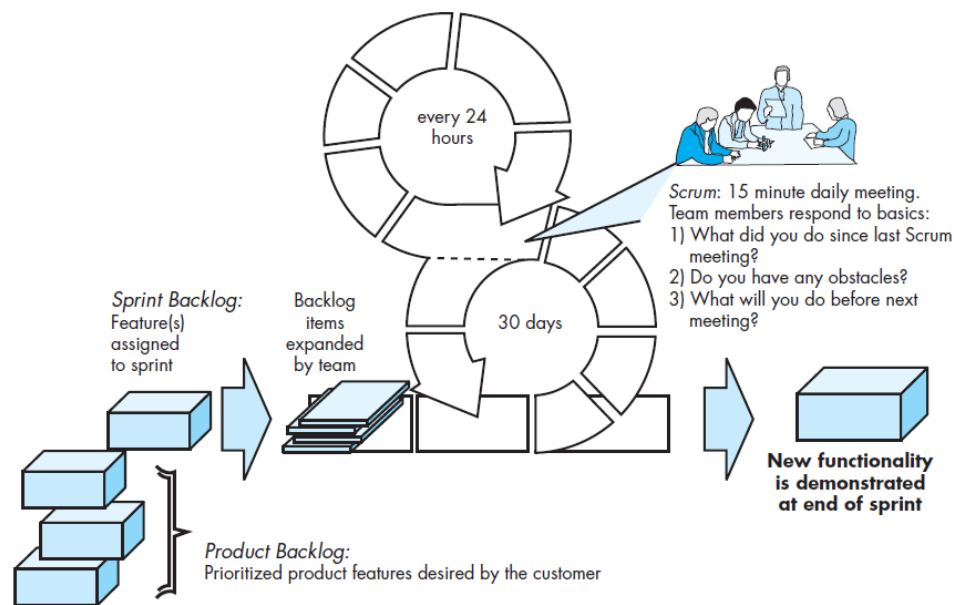
در طول اجرای هر sprint جلساتی روزانه (هر ۲۴ ساعت) و کوتاه حدود ۱۵ دقیقه با هدف کشف زود هنگام خطاها و بررسی پیشرفت پروژه توسط تیم اسکرام و با حضور اعضای تیم (تیم تولید و ذینفعان) برگزار می شود که به آن جلسات اسکرام (scrum meeting) گفته می شود.

در این جلسات باید به سه پرسش زیر پاسخ داده شود:

- از جلسه قبل تا این جلسه چه کارهایی انجام شده است؟
- از جلسه قبل تا این جلسه با چه موانعی مواجه شده اید؟
- از این جلسه تا جلسه بعدی چه کارهایی قرار است انجام شود؟



اگرچه جلسات اسکرام به نسبت کوتاه است، اما تاثیر جلسات بر روند بهبود فرآیند قابل چشم‌پوشی نیست. کشف زودهنگام خطاهای بالقوه از مهم‌ترین مزایای این جلسات است. این موضوع منجر به این می‌شود که تیم پروژه زمانی خطا را مرتفع کند که هزینه و زمان مورد نیاز برای رفع آن هنوز مقداری قابل قبول است. مزیت دیگر این جلسات آن است که دانش عمومی تیم، نسبت به کار یکدیگر و کل کاری که در sprint جاری انجام می‌شود، بالا رود. این موضوع باعث استحکام کار تیمی خصوصا در تیم‌های خودسازمانده (self-organizing) می‌شود. جلسات اسکرام توسط رهبر اسکرام (scrum master) هدایت می‌شود. رهبر اسکرام ضمن هدایت جلسات، مسئولیت ارزیابی پاسخ‌های اعضا در این جلسات را نیز برعهده دارد. شکل زیر گویای روند کارکرد متدولوژی اسکرام است:



بعد از خاتمه یک sprint، اعضای تیم طی جلسه‌ای به Product Owner و سایر ذینفعان پروژه نشان می‌دهند که چکار کرده‌اند و چطور از نسخه‌ی جاری نرم‌افزار می‌شود استفاده کرد. در واقع پس از اتمام هر sprint کارکردهای جدید که در طول این sprint ساخته شده‌اند، در قالب یک افزایش جدید از نرم‌افزار به صاحب محصول و ذینفعان ارائه می‌شود. همانطور که پیش‌تر نیز گفتیم، کارکردهای جدید هر افزایش، دقیقا برابر تمام کارکردهایی که در ابتدای sprint برنامه‌ریزی شده بود، نیست. در واقع گاهی محدودیت زمانی sprint باعث می‌شود در مواقعی که sprint از زمان‌بندی عقب است، برخی از کارکردها در آن افزایش پیاده‌سازی نشوند. در این شرایط پیاده‌سازی این کارکردها به افزایش‌های بعدی سپرده می‌شود. به مرحله‌ی ارائه‌ی کارکردهای جدید

نرم‌افزار در انتهای هر sprint یک demo گفته می‌شود. پس از تحویل هر افزایش یا sprint، افزایش یا sprint بعدی برنامه‌ریزی می‌شود. همچنین در هنگام برنامه‌ریزی sprint بعدی، بازخورد برنامه‌ریزی sprint قبلی مورد استفاده قرار می‌گیرد.

نتیجه اینکه از مجموع جلسات اسکرام یک Sprint بوجود می‌آید و از مجموع این sprintها هم کل فرآیند توسعه‌ی نرم‌افزار ایجاد می‌گردد. متدولوژی اسکرام برای انجام پروژه‌های پراهمیت که تحت فشار زمان‌بندی بوده و همواره در حال تغییر هستند، مناسب و کارآمد است.

## تست‌های فصل یازدهم

۱- کدام مورد به عنوان یک متدولوژی شناخته می‌شود؟  
(مهندسی IT - دولتی ۸۸)

ORACLE (۱)      4GT (۲)      RAD (۳)      X-Programming (۴)

۲- کدام یک از گزینه‌های زیر از اصول رسیدن به چابکی نیست؟  
(مهندسی IT - دولتی ۹۳)

(۱) شرط چابکی ادغام طراحی و ساخت است.  
(۲) تولیدکنندگان و مشتریان باید روزانه و پیوسته با یکدیگر همکاری کنند.  
(۳) بهترین طراحی‌ها، معماری‌ها و نیازها از تیم‌های خود سازمانده منتج می‌شود.  
(۴) رضایت مشتری از طریق تحویل نسخ محصول بطور پیوسته و سریع بالاترین اولویت را دارد.

۳- کدام عبارت در مورد بازآرایی کد (Refactoring) در XP، هیچگاه درست نیست؟  
(مهندسی IT - دولتی ۹۶)

- (۱) بازآرایی، کد را خواناتر می‌کند.  
(۲) بازآرایی، کد را تغییرپذیرتر می‌کند.  
(۳) بازآرایی رفتار بیرونی کد را اصلاح می‌کند.  
(۴) بازآرایی ساختار داخلی کد را بهبود می‌دهد.

## پاسخ تست‌های فصل یازدهم

۱- گزینه (۴) صحیح است.

ORACLE یک DBMS است، 4GL زبان‌های نسل چهارم را گویند و RAD یک مدل توسعه سریع می‌باشد. X-Programming یک متدولوژی چابک است.

۲- گزینه (۱) صحیح است.

طراحی در متدولوژی XP بر پایه‌ی سادگی بنا شده است، این اصل که سادگی را حفظ کن (keep it simple). در متدولوژی XP همواره یک طراحی ساده بر یک طراحی پیچیده برتری دارد. اما این سادگی در طراحی به این معنی نیست که شرط چابکی حذف طراحی و یا ادغام طراحی و ساخت است.

۳- گزینه (۳) صحیح است.

صورت سوال به این شکل است:

کدام عبارت در مورد بازآرایی کد (Refactoring) در XP، هیچگاه درست نیست؟

(۱) بازآرایی، کد را خواناتر می‌کند.

گزینه اول پاسخ سوال نیست.

(۲) بازآرایی، کد را تغییرپذیرتر می‌کند.

گزینه دوم پاسخ سوال نیست.

(۳) بازآرایی رفتار بیرونی کد را اصلاح می‌کند.

گزینه سوم پاسخ سوال است، زیرا طراحی در متدولوژی XP بر پایه‌ی سادگی بنا شده است، بنابراین این دغدغه وجود دارد که گاهی طراحی‌های XP، از کیفیت الگوریتمی (غیروظیفه‌مندی یا مزه) مناسبی برخوردار نباشد. متدولوژی XP برای مرتفع نمودن این دغدغه، روش فاکتورگیری مجدد یا بازآرایی (refactoring) را پیشنهاد می‌کند. بنابراین با وجود اینکه روش فاکتورگیری مجدد در مرحله کدنویسی انجام می‌شود اما ماموریت آن ارتقاء کیفیت الگوریتمی و خوشمزه‌سازی طراحی است. فاکتورگیری مجدد یا بازآرایی، فرآیند تغییر، بهبود و ارتقاء کیفیت کد است، بازآرایی، کد را خواناتر می‌کند، بازآرایی، کد را تغییرپذیرتر می‌کند، بازآرایی، ساختار داخلی کد را بهبود می‌دهد اما بدون آنکه بازآرایی، رفتار بیرونی کد را اصلاح کند، به عبارت دیگر بازآرایی، رفتار بیرونی کد را اصلاح نمی‌کند یعنی ورودی‌ها و خروجی‌های کد را تغییر نمی‌دهد.

(۴) بازآرایی ساختار داخلی کد را بهبود می‌دهد.

گزینه چهارم پاسخ سوال نیست.

## مشاوره تخصصی رشته کامپیوتر و IT

در راستای رسالت مؤسسه فرهنگی و انتشاراتی بابان مبنی بر ارتقای سطح علم و دانش کشور و کمک همه جانبه به دانشجویان و داوطلبان گرامی، در جهت قبولی در کنکور کارشناسی ارشد و دکتری مهندسی کامپیوتر و IT دو طرح زیر را پایه‌ریزی کرده‌ایم:

(۱) ارائه مشاوره تخصصی حضوری و غیر حضوری (تلفنی و آنلاین)

(۲) برگزاری کلاس‌های حضوری و غیر حضوری (فیلم آموزشی و کلاس آنلاین) بدین منظور برای تماس با شما و تعیین وقت قبلی عدد ۲ را به شماره ۵۰۰۲۰۶۰۴۵ پیامک نمایید.

برای آشنایی بیشتر با خدمات ارائه شده توسط موسسه بابان به وب سایت [khalilifar.ir](http://khalilifar.ir) یا کانال تلگرام [@arastookhalilifar](https://t.me/arastookhalilifar) مراجعه فرمایید.

تلفن دفتر مرکزی موسسه بابان: ۰۲۱-۷۷۹۷۲۸۶۸

پایگاه اطلاع رسانی موسسه بابان: [www.baban.ir](http://www.baban.ir)

تلفن دفتر فروشگاه انتشارات بابان: ۰۲۱-۷۷۹۷۳۳۸۶

فروشگاه اینترنتی انتشارات بابان: [shop.baban.ir](http://shop.baban.ir)