Certified Professional for Requirements Engineering

RE@Agile Primer

سرفصل و راهنمای یادگیری

Lars Baumann, Peter Hruschka,
Kim Lauenroth, Markus Meuten,
Sacha Reis, Gareth Rogers,
Francois Salazar,
Hans-Jörg Steffe, Thorsten Weyer

v1.4.0 | ۲۰۲۴، ژوئن

شرایط استفاده

۱. افراد و آموزش‌دهندگان ممکن است از این برنامه درسی و راهنمای یادگیری به‌عنوان مبنایی در سمینارهای خود استفاده کنند، مشروط بر اینکه حق چاپ به اثبات رسیده و در مطالب سمینار گنجانده شود. هر شخصی که قصد استفاده تبلیغاتی از این سرفصل و راهنمای یادگیری را داشته باشد، به اجازه کتبی از بورد بین‌المللی مهندسی نیازمندی‌ها احتیاج دارد.

۲. فرد یا گروهی از افراد می‌توانند از این برنامه درسی و راهنمای یادگیری به‌عنوان پایه‌ای برای مقالات، کتاب‌ها یا سایر نشریات مشتق شده استفاده کنند، به شرطی که داشتن حق چاپ برای نویسندگان و IREB e.V به‌عنوان منبع و صاحب این سند در چنین نشریاتی تأیید شده باشد.

قدردانی

این سیلابس و راهنمای آموزشی توسط: Lars Baumann, Peter Hruschka, Kim Lauenroth, Markus Meuten, Sacha Reis, Gareth Rogers, Francois Salazar, Hans-Jörg Steffe, Thorsten Weyer نوشته شده است.

نظرات مروری توسط: Bernd Aschauer, Thomas Emmerich, Dirk Fritsch, Rainer Grau, Andrea Hermann, Krystian Kaczor, Niko Kaintantzis, Elisabeth Larson, Ladislau Szilagyi, Daniel Tobler, Erik van Veenendaal, Arun Vetrivel, Sven van der Zee ارائه شده‌اند.

نقد و بررسی انگلیسی توسط Joy Beatty, Gareth Rogers and Candase Hokanson انجام شده است. ترجمه انگلیسی‌به‌فارسی این متن توسط آیدین ضیاپور سُهی، آراز ساعی ارسی، حسن حقیقی و محمود نشاطی انجام شده است.

ما از تمامی این افراد بابت مشارکتشان تشکر می‌کنیم.

در تاریخ ۲ مارچ ۲۰۱۷ به توصیه Xavier Franch توسط شورای IREB تایید شده است.

هدف از این مستند

این سرفصل و راهنمای یادگیری به تعریف سطح مبانی اولیه گواهینامه حرفه‌ای RE@Agile که توسط بورد بین‌المللی مهندسی نیازمندی‌ها (IREB) معرفی شده است، می‌پردازد. این سرفصل و راهنمای یادگیری، مقدمات طراحی و ایجاد مطالب درسی را برای آموزش‌دهندگان فراهم می‌کند. دانشجویان می‌توانند از این سرفصل و راهنمای یادگیری برای آمادگی شرکت در آزمون استفاده کنند.

محتوای سرفصل و راهنمای یادگیری

سطح مقدماتی نیازهای کلیه افراد درگیر در موضوع مهندسی نیازمندی‌ها و چابک را برطرف می‌کند. این سطح شامل افرادی است که در نقش‌هایی نظیر مدیریت پروژه یا مدیریت فناوری اطلاعات، کارشناسان دامنه، تحلیلگران سیستم و توسعه‌دهندگان نرم‌افزار و همچنین اسکرام مسترها، مالکان محصول و افرادی که بخشی از سازمان‌های چابک فعالیت می‌کنند.

محدوده محتوا

RE@Agile از دیدگاه IREB در مورد ارزش‌های چابک و همچنین از دیدگاه چابک در مورد ارزش‌های مهندسی نیازمندی‌ها الهام گرفته شده است. محتوای آن شامل طبقه‌بندی و ارزیابی محصولات کاری و تکنیک‌های مهندسی نیازمندی‌ها در زمینه چابک، محصولات کاری و تکنیک‌های چابک در زمینه مهندسی نیازمندی‌ها و عناصر اساسی فرایند در توسعه محصول چابک است. RE@AGILE به انگیزه استفاده از چابک در یک فرایند توسعه اشاره می‌کند.

یک موضوع بسیار، مهم هم‌افزایی بین مهندسی نیازمندی‌ها و چابک است: اصول چابک در مورد مهندسی نیازمندی‌ها و طرز فکر چابک دررابطه‌با ارزش‌های اصلی مهندسی نیازمندی‌ها.

مدارک RE@Agile بورد بین‌المللی مهندسی نیازمندی‌ها، حمایت از گروه‌های زیر را به‌عنوان هدف خود قرار داده است:

- مهندسین نیازمندی‌هایی که می‌خواهند در توسعه چابک نقش داشته باشند و قصد دارند تکنیک‌های خود را با موفقیت در این محیط به کار گیرند.
- مهندسین نیازمندی‌هایی که می‌خواهند مفاهیم و تکنیک‌های تثبیت شده از رویکردهای چابک را برای بهبود فرایندهای مهندسی نیازمندی¬ها خود استفاده کنند.
- متخصصان چابکی که می‌خواهند ارزش و مزایای دیسیپلین مهندسی نیازمندی‌ها را در پروژه‌های چابک درک کنند.
- متخصصان چابکی که می‌خواهند با استفاده از تکنیک‌ها و روش‌های اثبات شده مهندسی نیازمندی‌ها، توسعه چابک را بهبود بخشند.

- افرادی از رشته‌های مرتبط - مدیران فناوری اطلاعات، آزمون گران (Testers)، توسعه‌دهندگان، معماران و سایر نمایندگان مشاغل درگیر در توسعه (عمدتاً، اما نه‌تنها توسعه نرم‌افزار) - که می‌خواهند بدانند چگونه می‌توان رویکردهای مهندسی نیازمندی‌ها و چابک را در فرایندهای توسعه با موفقیت ترکیب کرد.

سطح جزئیات

سطح جزئیات این سرفصل و راهنمای یادگیری موجب سازگاری تدریس و آزمون در سطح بین‌المللی می‌شود. در راستای رسیدن به این هدف، این سرفصل و راهنمای یادگیری از موارد زیر تشکیل شده است:

- اهداف آموزشی عمومی،
- محتوا به همراه شرح اهداف آموزشی و
- ارجاع به متون بیشتر (در صورت لزوم)

اهداف آموزشی / سطوح دانش شناختی

یک سطح شناختی به هر ماژول برنامه درسی و راهنمای مطالعه اختصاص‌داده‌شده است. سطوح بالاتر شامل سطوح پایین‌تر می‌شوند. فرموله‌کردن اهداف آموزشی با استفاده از افعال "دانستن" برای سطح L۱ و "تسلط و استفاده" برای سطح L۲ بیان می‌شود. این دو فعل به افعال زیر اشاره دارند:

کلیه واحدها و اهداف آموزشی در این سرفصل به یک سطح شناختی اختصاص داده شده‌اند. سطوح زیر در این سرفصل مورداستفاده قرار گرفته‌اند:

- **L۱: دانستن** (توصیف، برشمردن، مشخص‌کردن، شناسایی، نام‌گذاری، به‌یادآوردن و...) - یادآوری یا بازیابی مطالب قبلاً آموخته شده.
- **L۲: درک** (تشریح‌کردن، تفسیرکردن، تکمیل‌کردن، خلاصه‌کردن، توجیه‌کردن، طبقه‌بندی، مقایسه‌کردن و...) - درک / ساختن معنی از مواد یا موقعیت‌های داده شده.
- **L۳: اعمال** (مشخص‌کردن، نوشتن، طراحی، توسعه‌دادن، پیاده‌سازی و...) - اعمال دانش و مهارت در موقعیت‌های معین.

توجه داشته باشید که یک هدف یادگیری در سطح دانش شناختی Ln حاوی عناصر تمام سطوح شناختی زیر آن (L۱ تا Ln-۱) است.

مثال:

یک هدف یادگیری از نوع "به‌کارگیری تکنیک مهندسی نیازمندی‌های xyz" در سطح دانش شناختی (L۳) است. بااین‌حال، توانایی اعمال مستلزم آن است که فراگیران تکنیک مهندسی نیازمندی‌های xyz سطح (L۱) را بدانند و بدانند که این تکنیک برای چیست (L۲).

تمامی اصطلاحاتی که در واژه‌نامه به‌عنوان اصطلاحات پایه معرفی شده‌اند، باید مورد یادگیری قرار گیرند (L۱)، حتی اگر به طور صریح در اهداف آموزشی ذکر نشده باشد. واژه‌نامه در صفحه خانگی IREB در https://www.ireb.org/en/downloads/#re-agile-glossary برای دانلود در دسترس است.

این سرفصل و راهنمای یادگیری از اصطلاح "RE" برای مهندسی نیازمندی‌ها استفاده می‌کند.

ساختار سرفصل و راهنمای یادگیری

این سرفصل و راهنمای یادگیری از ٤ بخش اصلی تشکیل شده است. هر بخش یک واحد درسی (EU) را پوشش می‌دهد. عنوان فصل اصلی شامل سطح شناختی فصل‌های آن ها است که شامل بالاترین سطح زیر فصل‌های آن‌ها می‌شود. علاوه بر این، مدت زمان پیشنهاد شده، حداقل زمان تدریسی است که یک دوره باید برای آن فصل در نظر بگیرد. اصطلاحات مهم فصل، که در واژه نامه تعریف شده اند، در ابتدای فصل ذکر شده اند.

مثال: **EU۲ مبانی و اصول L۱ RE@AGILE**

مدت‌زمان: ۱ ٤/۱ ساعت

اصطلاحات: مالک محصول، بک‌لاگ محصول، بک‌لاگ اسپرینت، اپیک ها، داستان‌های کاربر، نقشه‌های داستان

این مثال نشان می‌دهد که فصل ۲ شامل اهداف آموزشی در سطح L۱ است و ۷۵ دقیقه برای آموزش مطالب در این فصل در نظر گرفته شده است.

هر فصل می‌تواند شامل زیرفصل‌ها باشد. همچنین عناوین آن‌ها شامل سطح شناختی محتوای آن‌ها است.

اهداف آموزشی (EO) قبل از متن اصلی ذکر شده است. شماره‌گذاری آن‌ها نشان می‌دهد که به کدام زیر فصل تعلق دارند.

مثال: **EO ۳٫۱٫۲**

این مثال نشان‌دهنده این است که هدف آموزشی EO ۳٫۱٫۲ در زیر فصل ۳٫۱ تشریح شده است.

آزمون

این سرفصل و راهنمای یادگیری، مبنای آزمون گواهینامه RE@Agile Primer است. دو نوع آزمون مختلف در دسترس قرار دارند:

- آزمون نظارتی چندگزینه‌ای به همراه مدرک رسمی RE@AGILE Primer، شبیه آزمون‌های چندگزینه‌ای سطح مبانی CPRE و ماژول RE@AGILE Practitioner، اما با مدت‌زمان ٤۰ دقیقه.

- آزمون خودارزیابی چندگزینه‌ای آنلاین به همراه گواهی شرکت در آزمون.

امتحانات نظارتی می‌توانند بلافاصله پس از یک دوره آموزشی یا به‌صورت مستقل از دوره‌ها (به‌عنوان‌مثال، در یک مرکز امتحان) برگزار شوند. لیستی از ارائه‌دهندگان شناخته شده آزمون را می‌توان در صفحه اصلی IREB پیدا کرد.

آزمون خودارزیابی از طریق صفحه اصلی IREB در دسترس خواهد بود: http://www.ireb.org

یک سؤال در آزمون می‌تواند مطالب مربوط به چندین فصل از سرفصل و راهنمای یادگیری را پوشش دهد. تمامی فصل‌های موجود در سرفصل و راهنمای یادگیری (EU ١ تا EU ٤) می‌توانند مورد آزمون قرار گیرند.

تاریخچه نسخه

نظر	تاریخ	نسخه
• اولین نسخه فارسی.	۱ سپتامبر ۲۰۲۱	1.0.0
• رفع باگ‌ها. • همسویی با راهنمای اسکرام ۲۰۲۰. • همسویی با نسخه ۳ سطح مقدماتی CPRE و نسخه ۲ سطح پیشرفته CPRE RE@Agile. • سطوح دانش شناختی باتوجه‌به تعریف جدید IREB اعمال شده است. • همسویی اهداف آموزشی با سطوح دانش شناختی جدید. • زمان پیشنهادی تدریس واحدهای آموزشی متناسب با سطوح دانش شناختی جدید اصلاح شده است.	ژانویه ۲۰۲۳	1.2.0
• نسخه‌ای منتشر نشده است.	-	1.3.0
• رفرنس‌های دارای مشکل اصلاح شدند. • طراحی سازمانی جدید اعمال شد.	ژوئن ۲۰۲٤	1.4.0

محتوا

انگیزه و زمینه

کیفیت نیازمندی‌ها، فارغ از متدولوژی توسعه مورداستفاده، موفقیت یا عدم موفقیت کل فرایند توسعه را تعیین می‌کند. برخلاف تصور عموم، تکنیک‌ها و روش‌های دیسیپلین مهندسی نیازمندی‌ها (Requiremets Engineering discipline) در استفاده از آن‌ها در روش‌های خاص توسعه (مانند مدل آبشاری یا اسکرام) بسیار مفید هستند بااین‌حال، مهندسی نیازمندی‌ها اغلب به‌عنوان یک دیسیپلین غیرچابک توسعه درک شده است که منجر به این باور غلط می‌شود که مجموعه دانش مهندسی نیازمندی‌ها، هیچ ارتباطی در موفقیت فرایندهای توسعه چابک ندارد.

در بسیاری از موارد، رویکردهای مهندسی نیازمندی‌ها و چابک به‌جای اینکه با هم درنظر گرفته شوند، جدای از هم در نظر گرفته می‌شوند. با اینکه در فرایندهای مرسوم توسعه، مهندسی نیازمندی‌ها با نقش‌های اختصاصی به‌عنوان یک دیسیپلین جداگانه در چرخه حیات یک سیستم ایجاد می‌شود، در توسعه چابک اهمیت این موضوع اغلب دست‌کم گرفته می‌شود.

رویکردهای چابک مبتنی بر ارتباط مستقیم، سادگی راهکارها و بازخورد هستند. یکی از ارزش‌های اصلی آن‌ها واکنش سریع به تغییرات است؛ بنابراین، تغییر در نیازمندی‌ها و اولویت‌های آن‌ها مفهوم ذاتی تمام رویکردهای چابک را نشان می‌دهد. در حقیقت، اهمیت قائل‌شدن به صلاحیت مهندسی نیازمندی‌ها (Requirmenets Engineering competence) در فرایندهای توسعه چابک، ضمن افزایش پایدار کیفیت سیستم‌ها و محصولات پیشرفته، می‌تواند در موفقیت پروژه‌های چابک تأثیرگذار باشد. از سوی دیگر، روش مهندسی نیازمندی‌ها می‌تواند به طور قابل‌توجهی از برخی اصول و تکنیک‌های بسیار مفید چابک، مستقل از روش خاص توسعه استفاده شده، بهره‌مند شود.

در حال حاضر در بسیاری از موارد، افراد یا در مهندسی نیازمندی‌ها یا در استفاده از برخی رویکردهای چابک متخصص هستند. در نتیجه، افراد از هر دو سمت باید راه خود را پیدا کنند تا از مزایای استفاده از اصول و فنون حوزه صلاحیتی دیگر استفاده کنند. در حال حاضر گواهینامه‌هایی با تمرکز بر ادغام هر دو زمینه صلاحیت‌ها چه از جامعه مهندسی نیازمندی‌ها یا از جامعه چابک در دسترس نیست. ازاین‌رو ایجاد یک پل کاملاً پذیرفته شده میان مهندسی نیازمندی‌ها و رویکردهای چابک و در نتیجه، همچنین بین مهندسین نیازمندی¬ها و کارشناسان چابک، بسیار امیدوارکننده است تا هر دو بتوانند به طور مؤثر با یکدیگر ارتباط برقرار کنند.

پاسخ IREB به این درخواست، مدرک RE@Agile است.

چنین پلی باید از دو جهت مختلف ساخته شود: ازیک‌طرف، جامعه مهندسی نیازمندی‌ها باید درک کنند که چگونه روش‌ها و تکنیک‌های مختلف خود را در فرایندهای توسعه چابک به طور موفقیت‌آمیز به کار گیرند و همچنین درک کنند که چگونه می‌توانند از تکنیک‌های خاص رویکردهای چابک برای بهبود پرکتیس مهندسی نیازمندی‌ها (Requirements Engineering practice) استفاده کنند. از طرف دیگر، از آنجا که هدف رویکردهای چابک ارائه نرم‌افزاری باارزش دراسرع‌وقت است، متخصصان چابک باید بدانند که چگونه با استفاده از مفاهیم و تکنیک‌های اثبات شده از دیسیپلین مهندسی نیازمندی‌ها، از این اثر استفاده کنند.

اصول اصلی RE@Agile عبارتند از:

- *مهندسی نیازمندی‌ها رویکردهای چابک می‌توانند از یکدیگر استفاده کنند*

RE@Agile مزایا و مشکلات احتمالی تکنیک‌های مهندسی نیازمندی‌ها و چابک را تجزیه‌وتحلیل می‌کند. برای این منظور، RE@Agile به استفاده از محصولات کاری و تکنیک‌های رشته مهندسی نیازمندی‌ها در فرایندهای چابک و همچنین استفاده از محصولات کاری، نقش‌ها و تکنیک‌های رویکردهای چابک در فرایندهای مهندسی نیازمندی‌ها در زمینه روش‌های مختلف توسعه می‌پردازد.

- *فرایندهای سبک و بسیار سازگار*

بر اساس فلسفه RE@Agile ، تمایز بین فرایندهای توسعه پیش‌بینی‌کننده و انطباقی از اهمیت حیاتی برخوردار است. RE@Agile، ایده انجام یک روش سبک و بسیار سازگار برای انجام فعالیت‌های مهندسی نیازمندی‌ها را در توسعه چابک پیشنهاد می‌کند. در RE@Agile، مهندسی نیازمندی‌ها، یک نظم اصلی است و نه یک مرحله واحد: یک فرایند مداوم که باید به‌صورت سیستماتیک انجام شود و به سطح بالایی از مهارت و تجربه نیاز دارد.

- *همکاری نزدیک در تیم و با ذی‌نفعان اصلی و نیازمندی‌های به‌موقع (Just-in-time-requirements)*

ارتباط مکرر و همکاری نزدیک بین همه اعضای تیم و ذی‌نفعان اصلی از اهمیت ویژه‌ای برای موفقیت فرایندهای توسعه چابک برخوردار است. در RE@Agile، تیم به همراه ذی‌نفعان اصلی، نیازمندی‌ها را به روشی کاملاً تعاملی استخراج، تجزیه‌وتحلیل، اصلاح و مستند می‌کند. RE@Agile به پشتیبانی از متخصصان در انتخاب فعالیت‌های مناسب، در زمان مناسب، برای اطمینان از نیازمندی‌های باکیفیت بالا، پیش از پیاده‌سازی آن‌ها می‌پردازد.

- *استخراج ، تجزیه و تحلیل ، ذکر مشخصات (Specification) و پالایش موقعیتی (Situational) و انتخابی (Selective) نیازمندی ها*

RE@Agile مبتنی بر این ایده است که لازم نیست قبل از شروع پیاده‌سازی سیستم، هر نیازمندی دقیقاً و با جزئیات کامل مشخص شود. در عوض، فقط نیازمندی‌هایی که بیش از حد پیچیده (به‌عنوان‌مثال برای

ذی‌نفعان یا تیم توسعه قابل‌درک نیستند) یا مهم هستند (به‌عنوان‌مثال نمی‌توانند ریسک برداشت اشتباه (Misunderstanding) داشته باشند) با جزئیات بیشتری پالایش و مشخص شده‌اند. روند کلی به فلسفه مشترکی متکی است که از تغییر در نیازمندی عملکردی استقبال می‌کند و به‌راحتی قابل‌انطباق است.

- *از فعالیت‌ها و عملکردهای کمتر مرتبط پرهیز کنید و کمینه محصول پذیرفتنی را تضمین کنید*

یکی از اصول چابکی، "سادگی" است. مطابق این اصل، اولین مرحله توسعه سیستم یا محصول در فرایندهای چابک اغلب MVP (کمینه محصول پذیرفتنی) است. MVP یک سیستم مجزا و قابل‌استقرار است که فقط مجموعه‌ای پایه از ویژگی‌ها را ارائه می‌دهد که ارزش تجاری کافی را برای کاربران نهایی ایجاد می‌کند تا امکان یادگیری معتبر را فراهم کند. حداقل دامنه MVP، امکان حذف هدررفت را در حین توسعه فراهم می‌کند و فرصتی برای بازخورد سریع مشتری به وجود می‌آورد. یکی از مراحل بعدی محصول اغلب MMP (کمینه محصول قابل‌عرضه) است - محصولی با کمترین مجموعه ویژگی که نیازهای کاربران را برطرف می‌کند و بنابراین دارای ارزش بازار است. RE @ Agile پاسخ دو سؤال بسیار مهم را ارائه می‌دهد، سؤالاتی که حتی در فرایندهای توسعه غیرچابک نیز بسیار مهم هستند: "چگونه مدیریت انتشار و فرایند تعریف محصول را ساده کنیم؟" و "چگونه MVP یا MMP را بر اساس نیازمندی‌ها تعریف کنیم؟"

درباره مدرک RE@Agile:

آگاهی از سطح پایه CPRE و فرایندهای توسعه چابک به‌عنوان دانش پیش‌نیاز توصیه می‌شود.

گواهینامه RE@Agile Primer برای افراد حرفه‌ای از رشته‌های مرتبط است: مدیران پروژه، تحلیل‌گران کسب‌وکار، معماران، توسعه‌دهندگان، آزمون گران و همچنین افراد کسب‌وکاری. این مدرک بر ارتباط بین مهندسی نیازمندی‌ها و متخصصان چابک و همچنین درک اصطلاحات از هر دو حوزه تمرکز دارد. دارندگان این مدرک می‌توانند با متخصصان چابک در مورد مهندسی نیازمندی‌ها و با متخصصان مهندسی نیازمندی‌ها در مورد رویکردهای چابک و توسعه چابک صحبت کنند.

فرد دارای مدرک RE@Agile Primer:

- با اصطلاحات مربوط با مهندسی نیازمندی‌ها و رویکردهای چابک آشنا است.
- نقش و اهمیت مهندسی نیازمندی‌ها را در فرایندهای چابک و همچنین ارزش چابکی را در مهندسی نیازمندی‌ها درک می‌کند.

۱ انگیزه و ذهنیت‌ها (L1)

مدت‌زمان: ۱ ساعت و نیم

اصطلاحات: ارزش‌ها، بیانیه، روش‌ها، فعالیت‌ها، اسپرینت، چابک

اهداف آموزشی

هدف آموزشی ۱/۱/۱	دانستن انگیزه استفاده از روش های چابک
هدف آموزشی ۱/۲/۱	دانستن اهداف مهندسی نیازمندی ها مطابق با IREB
هدف آموزشی ۱/۲/۲	دانستن ارزش های اصلی بیانیه چابک و اصول ناشی از آن
هدف آموزشی ۱/۳/۱	دانستن تفاوت بین اصل (Principle)، پرکتیس (Practice) و فعالیت (Activity)
هدف آموزشی ۱/۳/۲	دانستن تفاوت میان طرز فکر چابک و طرز فکر مهندسی نیازمندی ها
هدف آموزشی ۱/۳/۳	دانستن هم افزایی طرز فکرها و ارزش ها نسبت به RE @ Agile
هدف آموزشی ۱/۳/٤	دانستن معنی "مستندسازی" در حوزه چابک (همسو با بیانیه چابک)
هدف آموزشی ۱/٤/۱	دانستن مزایا، مشکلات و باورهای غلط استفاده از RE @ Agile
هدف آموزشی ۱/٤/۲	دانستن مثال هایی از باورهای اشتباه
هدف آموزشی ۱/۵/۱	دانستن این مورد که ارزش های چابک را می توان به کارهای مفهومی تبدیل کرد
هدف آموزشی ۱/۵/۲	دانستن رویکردهای نمونه ای که امکان چابکی در کارهای مفهومی را فراهم می کنند

۱/۱ انگیزه استفاده از چابک (L1)

چندین مطالعه ([MeMi۲۰۱۵] را مشاهده کنید) نشان می‌دهند که تجارت فناوری اطلاعات به‌طورکلی در حال تغییر اساسی است: فناوری اطلاعات در حال تبدیل‌شدن به یک محرک اصلی در چندین زمینه تجاری (به‌عنوان‌مثال تجارت الکتریک، رسانه‌های اجتماعی و غیره) و حوزه‌های فنی (به‌عنوان‌مثال صنعت خودرو یا هواپیمایی) است. در نتیجه، سیستم‌ها و محصولات موجود در مشاغل مبتنی بر فناوری اطلاعات مجبور هستند برای سازگاری با نیازهای متغیر مشتریان یا بازار، انطباقی مداوم داشته باشند. به‌محض ایجاد تغییر در بازار، سیستم‌ها باید متناسب با تغییرات سازگار شوند.

روش های موجود توسعه که بر پیش بینی و ثبات دراز مدت متمرکز هستند، برای چنین شرایطی توسعه نیافته اند و اغلب در تغییر سریع کسب و کارها یا شرایط پروژه شکست می خورند. روش های چابک که توسط بیانیه چابک هدایت می شوند (۱/۲ را مشاهده کنید)، برای پرکردن این شکاف پدید آمده اند. چابک (یا چابکی) به خودی خود اصطلاحی دشوار است که می توان آن را به صورت زیر تعریف کرد ([ShYo۲۰۰۶] مشاهده کنید):

یک حرکت سریع کل بدن با تغییر سرعت یا جهت در پاسخ به یک محرک.

این تعریف نه از مهندسی نرم‌افزار بلکه از ورزش نشئت‌گرفته است، بااین‌حال انگیزه اساسی برای استفاده از روش چابک را منعکس می‌کند: درصورتی‌که وضعیت بازار یا پروژه نیاز به تغییرات سریع و کنترل شده داشته باشد، روش‌های چابک مناسب هستند. یک روش چابک البته چیزی فراتر از توسعه سریع است (۱٫۲ را مشاهده کنید) اما در اصل همه اصول در تحویل مکرر با یک کیفیت خاص متمرکز هستند. این امر، منجر به چرخه های بازخورد مکرر می شود که به نوبه خود امکان پاسخگویی سریع به نیازهای مشتری را فراهم می کند.

مهم است که بدانیم که نه روش‌های چابک و نه چابکی به‌خودی‌خود هدف نیستند [Meyer۲۰۱٤]. یک سازمان باید بتواند رویکرد توسعه مناسب و متناسب با نیازهای بازار، مشتریان و سازمان خود را انتخاب کند. گارتنر (Gartner) حتی اظهار داشت که توانایی توسعه فناوری اطلاعات با رویکرد صحیح عامل اصلی موفقیت در تجارت دیجیتال است [MeMi۲۰۱۵].

۱٫۲ طرز فکر و ارزش‌ها در مهندسی نیازمندی‌ها و چابک (L1)

طرز فکر و ارزش‌های RE در تعریف IREB از مهندسی نیازمندی‌ها بیان شده است (به [Glinz۲۰۲۲] نگاه کنید): رویکرد سیستماتیک و منظم تعیین و مدیریت نیازمندی‌ها باهدف درک خواسته‌ها و نیازهای ذی‌نفعان و به‌حداقل‌رساندن خطر ارائه سیستمی که این خواسته‌ها و نیازها را برآورده نمی‌کند.

در مهندسی نیازمندی‌ها ما به‌جای نرم‌افزار یا محصول، از سیستم استفاده می‌کنیم. استفاده از اصطلاح سیستم به معنای حذف محصولات، انواع دیگر نرم‌افزارها، یا حتی موارد دیگر (به‌عنوان‌مثال فرایندهای تجاری یا سخت‌افزار) نیست. مهندسی نیازمندی‌ها اصطلاح سیستم را به این دلیل ترجیح می‌دهد؛ زیرا این اصطلاح بر این واقعیت تأکید دارد که سیستم گروهی از قطعات یا عناصر است که در یک محیط با هم کار می‌کنند. مهندسی نیازمندی‌ها، محیط (Environment) را زمینه سیستم (System context) نام‌گذاری می‌کند. در سرفصل و راهنمای مطالعه، ما همیشه از اصطلاح سیستم استفاده خواهیم کرد و این عبارت باید شامل محصولات و هر نوع عنصر مرتبط با نرم‌افزار باشد.

علاوه بر این، سطح IREB FL [CPREFL۲۰۲۲] مجموعه‌ای از چهار فعالیت اصلی مهندسی نیازمندی‌ها را تعریف می‌کند: استخراج، مستندسازی، اعتبارسنجی / مذاکره و مدیریت نیازمندی‌ها. این لیست فعالیت‌ها نشان‌دهنده مجموعه خاصی از مراحل یا توالی انجام این فعالیت‌ها نیست. یک ارزش اصلی IREB FL این مورد است که مهندسی نیازمندی‌ها یک رویکرد فرایندی اگنوستیک (Prossess-agnostic approach) است: مهندسی نیازمندی‌ها، پیکره دانشی غنی از دانش متشکل از روش‌های مختلف و مجموعه‌ای غنی از تکنیک‌هایی را که می‌توان در هر رویکرد توسعه به کاربرد فراهم می‌کند. مهندسی نیازمندی‌ها، فرایندی را توصیه یا تعیین نمی‌کند.

این سرفصل و راهنمای مطالعه از اصطلاح "روش‌های چابک" برای اشاره به مجموعه غنی رویکردهایی که در زمینه چابک پدیدآمده است، استفاده خواهد کرد (۲ را مشاهده کنید). برای تمایز روش های چابک از سایر روش های توسعه (به عنوان مثال طرح-محور یا آبشاری) ، این سرفصل و راهنمای مطالعه از اصطلاح "روش های غیر چابک" استفاده می کند. این دو اصطلاح، ارزیابی بهترین گزینه را ضروری می دانند - IREB متقاعد شده است که هر دو روش (چابک و غیر چابک) ارزشمند هستند.

طرز فکر چابک توسط بیانیه چابک و دوازده اصل پشتوانه آن تعریف می‌شود ([AgileMan۲۰۰۱] را مشاهده کنید):

> بیانیه چابک
>
> **ما در حال کشف راه‌های بهتری برای توسعه**
>
> **نرم‌افزار با انجام آن و کمک به دیگران در انجام آن هستیم.**
>
> **از این طریق باید به ارزش‌های زیر دست یابیم:**
>
> افراد و تعاملات **ارجح بر فرایندها و ابزارها**
>
> نرم‌افزار کارکننده **ارجح بر مستندسازی جامع**
>
> مشارکت مشتری در انجام کار **ارجح بر مذاکرات بر اساس قرارداد**
>
> پاسخ به تغییر **ارجح بر تبعیت از برنامه**
>
> **درحالی‌که موارد سمت چپ ارزشمند هستند،**
>
> **ما برای موارد سمت راست ارزش‌های بیشتری قائل هستیم.**

اصول چابک

۱. بالاترین اولویت ما جلب رضایت مشتری از طریق تحویل زودهنگام و مداوم نرم‌افزارهای ارزشمند است.
۲. استقبال از تغییر نیازمندی‌ها، حتی در اواخر فرایند توسعه. فرایندهای چابک، تغییر را در راستای مزیتِ رقابتی مشتری کنترل می‌کنند.
۳. تحویل مداوم نرم‌افزار قابل‌استفاده از دو، سه هفته یک‌بار تا دو، سه ماه یک‌بار با ترجیح بر فاصله‌های زمانی کوتاه‌تر.
٤. ذی‌نفعان کسب‌وکار و توسعه‌دهنده‌ها باید به‌صورت روزانه در طول پروژه با هم کار کنند.
٥. پروژه‌ها را بر دوش افراد باانگیزه بنا کنید. فضای لازم را به آنها بدهید و از نیازهای آن‌ها پشتیبانی کنید به آنها اعتماد کنید تا کارها را انجام دهند.
٦. کارآمدترین و مؤثرترین روش انتقال اطلاعات به تیم توسعه و تبادل آن در میان اعضای تیم، گفتگوی رودررو است.

۷. نرم‌افزار قابل‌استفاده اصلی‌ترین معیار سنجش پیشرفت است.
۸. فرایندهای چابک، توسعه پایدار را ترویج می‌دهند. حامیان مالی، توسعه‌دهندگان و کاربران باید بتوانند سرعت پیشرفت ثابتی را براي مدت نامحدود حفظ کنند.
۹. توجه مداوم به برتری فنی و طراحی خوب باعث افزایش چابکی می‌شود.
۱۰. سادگی --هنر به حداکثر رساندن مقدار کار انجام نشده --ضروری است.
۱۱. بهترین معماری‌ها، نیازمندی‌ها و طراحی‌ها از تیم‌های خودسازمانده پدیدآور می‌شوند.
۱۲. در فواصل منظم، تیم بر چگونگی مؤثرتر شدن تأمل و تفکر می‌نماید و سپس رفتار خود را بر اساس بازتاب این تفکر، تنظیم و همسو می‌کند.

اگر ارزش‌ها و تفکرات مهندسی نیازمندی‌ها و چابک را مقایسه کنیم، نمی‌توان بخشی پیدا کرد که باارزش‌ها یا تفکرات دیگری در مغایرت باشد. مهم‌ترین ارزش توسط مهندسی نیازمندی‌ها و چابک به اشتراک گذاشته می‌شود و آن ارزش این است که کاربر نهایی محصول را خوشحال کند، زیرا این راهکار، متناسب با نیاز آن‌ها است یا بزرگ‌ترین درد آن‌ها را درمان می‌کند. بااین‌وجود، باید از این موضوع آگاه باشیم که طرز فکر و ارزش‌های مهندسی نیازمندی‌ها و چابک تا حدی از هم جدا هستند. نقطه اتصال مهندسی نیازمندی‌ها و چابک توسط زمینه RE@Agile تعریف شده است که در واحد درسی بعدی توضیح داده خواهد شد.

۱٫۳ اتصال مهندسی نیازمندی‌ها و چابک در راستای RE@Agile (L1)

قبل از اینکه به جزئیات RE@Agile بپردازیم، باید برخی اصطلاحات را شفاف کنیم. در صنعت و تحقیقات نرم‌افزار، پیکره دانش گسترده‌ای در مورد نحوه کار و رفتار هنگام تولید نرم‌افزار وجود دارد. این دانش در سطوح مختلف انتزاع وجود دارد. در ادامه، ما تمایز اصول، پرکتیس ها و فعالیت‌ها را به‌عنوان سه سطح انتزاع برای صحبت در مورد توسعه نرم‌افزار معرفی خواهیم کرد. ([Meyer۲۰۱٤] را مشاهده کنید)

- یک اصل عبارت تجویزی است که انتزاعی و با قابلیت اثبات نادرستی (Falsifiable) است
- یک تمرین/ تکنیک نمونه‌برداری از یک اصل برای یک زمینه خاص است
- فعالیت، یک اجرای واقعی یا برنامه‌ریزی‌شده از یک عمل است

اصطلاحات مهم برای تمایز این سه تعریف، تجویزی، انتزاعی و جعلی است. تجویز به این معنی است که بیانیه به‌جای بیان یک واقعیت یا خاصیت، اقدام را جهت‌دهی می‌کند. انتزاع یک اصل را از عمل متمایز می‌کند. به‌عنوان‌مثال، "آزمودن ویژگی نرم‌افزار، قبل از تحویل" تجویزی و انتزاعی است، درحالی‌که "ایجاد آزمون واحد (Unit test) برای هر ویژگی نرم‌افزاری" پرکتیسی مبتنی بر یک اصلِ داده شده است. یک فعالیت برای این مثال می‌تواند ایجاد یک تست واحد برای عملکرد جستجو در یک سیستم کتابخانه باشد. قابلیت اثبات نادرستی به این معنا است که شخصی با دانش کافی می‌تواند با یک اصل مخالف باشد. اصل گفته شده در بالا ("آزمون a... ") این معیار را برآورده می‌کند.

ممکن است استدلال شود که آزمون برای ویژگی‌های مهم ایمنی مناسب نباشد و در عوض، باید آن‌ها را با استفاده از روش‌های ریاضی اعتبارسنجی کرد. گزاره‌ای که قابلیت اثبات نادرستی نداشته باشد ("تلاش برای کیفیت بالا")، نباید اصلی برای هدایت رفتار تلقی شود.

دانستن اصول پشت عمل ما (که خود یک اصل است) منجر به تصمیم‌گیری آگاهانه درباره عملکرد ما می‌شود. دانستن روش‌های مختلف برای تحقق اصول، به ما این توانایی را می‌دهد که بسته به شرایط موجود واکنش متفاوتی داشته باشیم. قابلیت اثبات نادرستی، بحثی در مورد کاربرد یک اصل یا عمل را در یک زمینه خاص پرورش می‌دهد و بنابراین به تصمیم‌گیری در مورداستفاده یا عدم استفاده از یک اصل کمک می‌کند.

دانستن اصول و رویه‌ها فقط یک‌قدم ابتدایی محسوب می‌شود: استفاده صحیح از یک روش به‌تنهایی یک صلاحیت محسوب می‌شود. به‌عنوان‌مثال، تعریف موردهای کاربرد (Use cases)، یک روش شناخته شده برای اصل "تعیین نیازمندی‌های عملکردی برای ویژگی‌های مهم" است. بااین‌حال، نوشتن مورد کاربرد باکیفیت بالا در نوع خود یک صلاحیت است و دانستن عناصر الگوی مورداستفاده برای این کار کافی نیست.

در اصل، ما اکنون سه سطح صلاحیت را تعریف کرده‌ایم:

۱. دانستن اصول (مربوط به سطح شناختی L۱)
۲. درک روش‌هایی برای تحقق اصول داده شده (مربوط به سطح شناختی L۱)
۳. اعمال روش در یک زمینه خاص (توانایی انجام یک فعالیت باکیفیت بالا) (مربوط به سطح شناختی L۳)

با مقایسه اصول چابک با اصول مهندسی نیازمندی‌ها (۱٫۲ را مشاهده کنید)، می توان فهمید که چرا این دو اصل در بعضی مواقع تعارضاتی دارند: در حالی که چابک بر اهمیت نرم افزار قابل استفاده در مقایسه با اسناد و مدارک جامع تأکید می کند و افراد و ارتباطات را ارجح بر از فرایندها و ابزارها ارزیابی می کند، مهندسی نیازمندی ها به نوبه خود، مشغول استخراج و مستندسازی سیستماتیک نیازمندی ها به عنوان محصولات کاری است.

اجرای اغراق‌آمیز هر دو طرز فکر، در عمل می‌تواند منجر به تعارض شود: تفسیری نادرست از مهندسی نیازمندی‌ها این است که می‌توان یک مستند کامل، سازگار و مورد توافق را ایجاد کرد که بدون تغییر بیشتر قابل‌اجرا باشد. تفسیر مشابه کاذب از چابک این است که یک پروژه توسعه می‌تواند بدون هیچ کار مقدماتی شروع شود و تنها با ارائه نرم‌افزاری در فواصل منظم که توسط ذی‌نفعان بررسی می‌شود، به موفقیت برسد [توجه: از نگاه مهندسی نیازمندی‌ها، مشتریان زیر مجموعه ذی‌نفعان هستند.] و بر اساس بازخورد ذی‌نفعان بهبود بیابد.

ادعای ما این است که مهندسی نیازمندی‌ها و طرز فکر چابک در واقع در تضاد نیستند: هر دو رویکرد دارای هدف یکسان از تحویل نرم‌افزار در سطح کیفی کاملاً مشخص هستند. روش‌های چابک می‌توانند نرم‌افزارهای قابل‌استفاده را به روشی کارآمد و سریع (زمان چرخه کاهش‌یافته) (Reduced cycle time) تحویل دهند. مهندسی نیازمندی‌ها به فراهم‌کردن روش‌های مناسب برای درک خواسته‌ها و نیازهای ذی‌نفعان برای توسعه نرم‌افزاری می‌پردازد.

مهندسی نیازمندی‌ها به تسهیل موارد زیر کمک می‌کند:

- درک مناسب از خواسته‌ها و نیازهای کاربران برای تولید نرم‌افزار ارزشمند (اصل اول چابک)
- ابزارهای مناسب برای تشخیص تغییرات در بازار برای مزیت رقابتی ذی‌نفعان (اصل دوم چابک)
- ابزارها و تکنیک‌های مناسب برای تقویت همکاری مؤثر میان ذی‌نفعان و توسعه‌دهندگان (اصل چهارم چابک)
- ابزارها و تکنیک‌های مناسب برای حمایت از ارتباط کلامی (اصل ششم چابک)
- درک مناسب از خواسته‌ها و نیازهای ذی‌نفعان برای به‌حداقل‌رساندن توسعه نرم‌افزارهای غیرضروری (اصل دهم چابک).

تفاوت مهم میان استفاده از مهندسی نیازمندی‌ها در چابک و سایر روش‌های توسعه در زمان و فرایند اعمال شده است. IREB با استفاده از این برنامه درسی و راهنمای مطالعه، زمینه RE@Agile را تعریف می‌کند که نحوه اعمال مهندسی نیازمندی‌ها در زمینه روش‌های چابک را نشان می‌دهد.

IREB اصطلاح RE@Agile را بر اصطلاح "Agile Requirements Engineering" یا (مهندسی نیازمندی‌های چابک) ترجیح می‌دهد تا روشن شود مهندسی نیازمندی‌ها مستقل از فرایند است.

بیانیه توسعه نرم‌افزار چابک، بر ارزش اینکریمنتی از نرم‌افزار قابل‌استفاده (محصول در حال کار) در مقایسه با مستندات جامع تأکید دارد. یک تفسیر اغراق‌آمیز منجر به ایجاد این تصور غلط شده است که روش‌های چابک مستندات را به‌کلی کنار گذاشته‌اند. این تفسیر غلط است: در چابک اسنادی که دارای هدف هستند، تنها در زمانی که از توسعه پشتیبانی کنند یا جزئی از محصول باشند مورد استقبال و توصیه قرار می‌گیرند. درگذشته، یک مسئله درک شده این مورد بود که بسیاری از پروژه‌ها، اسنادی را بدون هدف یا ارزش‌افزوده مشخص ایجاد می‌کردند - طبق اصول، از این نوع اسناد جلوگیری می‌شود.

۱٫٤ مزایا، باورهای غلط و مشکلات استفاده از RE@Agile (مهندسی نیازمندی‌ها در چابک) (L1)

RE@Agile مزایای مختلفی را ارائه می‌کند. بااین‌حال، این مزایا به‌راحتی حاصل نمی‌شوند: باورهای غلط و دام‌هایی وجود دارند که باید از آن‌ها اجتناب شود.

۱٫٤٫۱ مزایای RE@Agile:

در یک تیم، صلاحیت‌های مهندسی نیازمندی‌ها و توسعه می‌توانند تحویل‌دادنی‌ها (Handovers) را کاهش دهند:
پرکتیس تیم‌های چند عملکردی در روش‌های چابک، نیازمند این است که تیم، تمام مهارت‌های لازم برای توسعه یک محصول را بر اساس نیازمندی‌های انتخاب شده داشته باشد.

انجام تسک‌های مهندسی نیازمندی‌ها در تیم، می‌تواند نیاز به ایجاد اسناد جامع نیازمندی‌ها به‌صورت مقدماتی (Upfront) را کاهش دهد، زیرا اعضای تیم می‌توانند جزئیات خاصی را مستقیماً برای سایر اعضا توضیح دهند. مستندسازی نتایج حاصل از بحث‌ها و حفظ دانش، از جمله مزیت‌های اسناد این زمینه خواهد بود. مزیت اسناد در این زمینه، مستندسازی نتایج حاصل از بحث‌ها و حفظ دانش خواهد بود.

توسعه تدریجی امکان بهینه‌سازی ایده‌های موجود را فراهم می‌کند: اصل اصلی روش‌های چابک، توسعه تدریجی نرم‌افزار در تکرارها است. روند تکرارشونده به ایجاد محصولات کاری می‌پردازد (به‌عنوان‌مثال یک مدل فرایند کسب‌وکار، یک اپیک، یک داستان کاربر، یک مورد کاربرد، یک نمونه اولیه رابط کاربر، یک توصیف فرایند یا یک نرم‌افزار)، و این موارد را در دنباله‌ای از فعالیت‌های توسعه و بررسی بهبود می‌بخشد. مزیت چنین روشی این است که کیفیت محصول کاری و یا نرم‌افزار به طور مداوم بهبودیافته و بهینه می‌شود. علاوه بر این، افزایش‌های کمتر، امکان بحث و گفتگوهای زودتر با مشتری را فراهم می‌کند و خطر شکاف‌های زیاد بین انتظارات مشتری و پیشرفت را به حداقل می‌رساند.

پالایش (Refinement) یک اصل برای بالغ کردن و اعتبارسنجی نیازمندی‌ها است: در توسعه چابک، روش بسیار خوبی ایجاد شده است - پالایش مداوم. در اینجا جلسات پالایش منظم برای تیم توسعه برگزار می‌شود تا نیازمندی‌ها را با ارتباط نزدیک با ذی‌نفعان به طور مداوم بررسی و جزئیات دهند. علاوه‌براین، از پرکتیس خوب تعریف آماده (Definition of Ready)، به‌عنوان دروازه کیفی (Quality gate) برای اعتبارسنجی آمادگی اجرای یک نیازمندی در تکرار بعدی، استفاده می‌شود.

مهندسی نیازمندی‌ها به تعریف اولیه یک بک‌لاگ محصول، کمک می‌کند: مهندسی نیازمندی‌ها چندین روش برای درک صحیح نیازمندی‌های ذی‌نفعان برای محصول موردنظر ارائه می‌دهد. بدین ترتیب، مهندسی نیازمندی‌ها درک عمیق‌تری از نیازمندی‌های موردنیاز برای تعریف اولیه بک‌لاگ محصول را فراهم می‌کند. مهم است که تشخیص دهیم چنین فعالیت مهندسی نیازمندی‌ها، به تشکیل یک مشخصات با جزئیات کامل منتهی نمی شود. در عوض هدف از انجام این فعالیت‌ها تمرکز بر درک کاملی از محصول در سطح مشخصی از انتزاع است (به‌عنوان‌مثال درک و تعریف موردهای کاربرد ضروری (Essential use cases) یا اپیک ها و داستان‌هایی کاربر). به‌عنوان‌مثال، یک فرایند پیچیده سطح بالای کسب‌وکار با استفاده از روش‌های کاملاً تثبیت شده مهندسی نیازمندی‌ها، می‌تواند برای ایجاد یک بک‌لاگ اولیه [CPREALAGILE۲۰۲۲]، به اپیک ها، فیچرها و داستان‌های کاربر تجزیه شود.

۱٫٤٫۲ تصورات غلط RE@Agile

در دنیای توسعه (عمدتاً توسعه نرم‌افزار)، برخی تصورات غلط و مشکلات در مورد مهندسی نیازمندی‌ها وجود دارد که باید موردبحث قرار گیرند:

تصور غلط - مهندسی نیازمندی‌ها تنها یک تجزیه‌وتحلیل مقدماتی (Upfront) است: اغلب اوقات، مهندسی نیازمندی‌ها تنها به‌عنوان یک فعالیت مقدماتی در نظر گرفته می‌شود. به‌عنوان یک دیسیپلین، مهندسی نیازمندی‌ها در واقع مستقل از روند کار است و از قبل تمامی کارها را اجرا نمی‌کند. در عوض، فعالیت‌های مهندسی نیازمندی‌ها می‌تواند به همان روشی که سایر فعالیت‌ها (به‌عنوان‌مثال کدگذاری یا آزمون) انجام می‌شوند، در یک روش چابک شکل بگیرد. مهندسی نیازمندی‌ها یک فعالیت تعبیه شده در هر تکرار است.

تصور غلط - کار مقدماتی (Upfront) شیطانی است: آماده شدن برای توسعه تکرارشونده، یک قسمت اساسی در هر کار توسعه ضمنی (Non-trivial development) است. تفکر مقدماتی (Upfront thinking) به خودی خود بر چرخه زندگی نرم‌افزاری خاص، یا یک مرحله یا مستند تجزیه‌وتحلیل طولانی، دلالت نمی کند: نحوه و زمان انجام تفکر مقدماتی توسط زمینه پروژه (Project context) به صورت دقیق تعیین می‌شود. متخصصان نباید ادبیات چابک را این‌گونه تفسیر کنند که تفکر مقدماتی به‌خودی‌خود چیز بدی است. بیانیه و اصول ارائه شده در بالا از چنین درکی پشتیبانی نمی‌کنند. شیوه های چابکی که ارزش تفکر مقدماتی را منعکس می کنند شامل نگاشت داستان کاربر ([Patt۲۰۱٤] را مشاهده کنید)، نمونه سازی اولیه ([Martin۱۹۹۱] را مشاهده کنید)، و توسعه مبتنی بر آزمون ([Beck۲۰۰۳] را مشاهده کنید) می شوند.

تصور غلط - مهندسی نیازمندی‌ها برابر است با مستندات: مهندسی نیازمندی‌ها اغلب فقط **با** اسنادی که تولید می‌کند مرتبط است. بااین‌حال، اسناد و مدارک یکی از نتایج بالقوه فعالیتی است که باعث ایجاد دانش می‌شود. مهندسی نیازمندی‌های خوب شامل آگاهی از این واقعیت است که حتی بهترین سند نیز هرگز کامل نیست. در عوض، یک مستند برای پشتیبانی از اهداف مندرج در ۳٫۲ عمل می کند: انطباق قانونی، حفظ اطلاعات ارزشمند، تسهیل ارتباطات و پشتیبانی از فرایندهای فکری.

تصور غلط - داستان‌های کاربر کافی هستند: داستان‌های کاربر یکی از روش‌های محبوب برای استخراج نیازمندی‌های ذی‌نفعان هستند. بااین‌حال، آنها برای شروع ارتباطات و نه نمایش مشخصات کامل، در نظر گرفته شده‌اند. مسیر یک نیاز نامشخص به سمت یک نیازمندی کامل در عمل "۳C - کارت (Card)، ارتباط (Communication)، تأیید (Confirmation)" خلاصه شده است. از طریق ترکیبی از داستان‌های کاربر با سایر تکنیک‌های مهندسی نیازمندی‌های تأیید شده مانند نمودارهای زمینه (Context diagrams)، نمونه‌سازی اولیه (Prototyping)، موردهای کاربرد (Use cases) و سفرهای کاربر (User journeys)، می‌توان به تصویر کامل‌تری از نیازمندی‌ها دست یافت.

تصور غلط - مستندسازی بی‌ارزش است، فقط کد دارای ارزش ماندگار است: گرچه کاملاً ممکن است در برخی از پروژه‌های دارای فرایند سنگین، مشخصات بیش از حد به‌عنوان یک مسئله مطرح باشد، اما نتیجه‌گیری اینکه همه مستندسازی‌ها بی‌ارزش هستند درست نیست. مستندات نیازمندی‌ها، درست مانند طراحی، آزمایش یا مستندات عملیاتی، در زمینه‌های خاصی جایگاه خود را دارد. مستندات همگی به یک اندازه معتبر و محصولات کاری ضروری هستند که از فرایند توسعه برای هر محصول نرم‌افزاری پایدار حاصل می‌شود.

تصور غلط - نرم‌افزار قابل‌استفاده تنها راه تأیید نیازها است: بیانیه چابک برای "نرم‌افزار کارکننده ارجح بر مستندسازی جامع" ارزش می‌نهد. وقتی صحبت از اعتبارسنجی نیازمندی‌ها می‌شود، یک نتیجه‌گیری غلط از این عبارت این است که اعتبارسنجی نیازمندی‌ها بر اساس نرم‌افزار قابل‌استفاده بر اعتبارسنجی مستندات مرتبط با نیازمندی‌ها ارجحیت دارد. نرم‌افزار به‌عنوان ابزاری برای تأیید نیازمندی‌ها در صورتی قابل‌قبول است که هزینه‌ها و ریسک‌های مربوط به چنین رویکرد اعتبارسنجی در مقایسه با نتیجه آن، قابل‌قبول باشد. اگر هزینه‌ها و یا ریسک‌ها زیاد باشد، مهندسی نیازمندی‌ها به فراهم‌کردن چندین ابزار می‌پردازد که امکان بازخورد سریع و تأیید نیازمندی‌ها، قبل از نوشتن یک خط کد را فراهم می‌کند، به‌عنوان‌مثال: موکاپ های رابط کاربر یا استوری بوردها.

۱٫٤٫۳ دام‌های RE@Agile

دام - نیازمندی‌ها را به‌عنوان نوعی اطلاعات یک‌شکل در نظر بگیرید: یک اشتباه معمول مرتبط با مهندسی نیازمندی‌ها در تمام زمینه‌ها / روش‌های پیاده‌سازی، درنظرگرفتن "نیازمندی" به‌عنوان نوعی یک‌شکل از اطلاعات است. بر اساس این برداشت اشتباه، اغلب مستندات نیازمندی‌ها به‌عنوان اتلاف وقت در نظر گرفته می‌شوند، زیرا نیازمندی‌ها به‌سرعت تغییر می‌کنند و به‌محض نوشتن بی‌اعتبار می‌شوند. نیازمندی‌ها یک نوع اطلاعات یکنواخت نیستند. نیازمندی‌ها را می‌توان در سطوح مختلف جزئیات، انتزاع و فرمت‌ها بیان کرد. به‌عنوان‌مثال، چشم‌انداز یا اهداف سیستم برای سیستم، نیازمندی‌هایی در سطح بالایی از انتزاع و به طور معمول با یک‌عمر طولانی هستند. نمونه اولیه قابل‌اجرا ابزاری برای تأیید مجموعه‌ای از نیازمندی‌ها و یا ایجاد الزامات جدید است.

دام - ازدست‌دادن تصویر بزرگ: روش‌های چابک اغلب بد فهمیده می‌شوند و به‌گونه‌ای اجرا می‌شوند که فقط به موضوعاتی بپردازند که همین‌الان در مقابل تیم قرار دارند. از دید یک توسعه‌دهنده، این ممکن است یک اصل مفید تلقی شود، زیرا توسعه‌دهنده می‌تواند انرژی ذهنی خود را بر روی کار موردنظر متمرکز کند و موضوعات طولانی‌مدت حواس او را پرت نمی‌کند. بااین‌حال، اگر همه فقط روی کارهای حاضر تمرکز کنند، تصویر بزرگ و چشم‌انداز طولانی‌مدت از بین می‌رود. روش‌های پایدار چابک در جلسات اختصاصی به چشم‌اندازهای بلندمدت و میان‌مدت می‌پردازند (به‌عنوان‌مثال جلسات پالایش، جلسات طراحی نقشه راه یا ورکشاپ‌های طراحی چشم‌انداز). یک دام مرتبط، ادامه‌دادن راهکار بدون تعریف کامل مشکل کسب‌وکار است (اغلب به‌عنوان نیاز نام‌برده می‌شود).

دام – ارائه بیش از اندازه اطلاعات به ذی‌نفعان: محصولات کاری مهندسی نیازمندی‌ها می‌توانند شامل تراکم بالایی از اطلاعات باشند و در تیم چابک، بسیار سریع ایجاد می‌شوند. این رویکرد غالباً در پروژه‌های "پیشرفته" یا با فناوری بالا مورداستفاده قرار می‌گیرد، جایی که به‌سختی می‌توان متخصصان موضوعی (Subject matter experts) را پیدا کرد. بااین‌حال، انجام یک تکرار کامل برای کار بر روی محصولات کاری مهندسی نیازمندی‌ها، بار بازبینی زیادی را برای ذی‌نفعان ایجاد می‌کند و آن‌ها را ملزم به خلاصه‌کردن مشخصات طولانی می‌کند. نتایج بهتر ممکن است با ترکیب خوبی از مشخصات و توسعه (نمونه‌سازی اولیه) حاصل شود.

دام - تشریح هر مبحث به‌صورت تدریجی و تکرارشونده: نباید هر مبحث نیازمندی‌های سیستم با جزئیات کامل (Fine-grained) و تکرارشونده (Iterative) توسعه یابد. مباحثی با پیچیدگی اضافی (Additive complexity) ([Meyer۲۰۱٤] را مشاهده کنید) برای توسعه تدریجی (Incremental development) مناسب هستند. مثال های معمول از این مباحث، شامل عناوین فرایندهایی هستند که می توانند به عناصر مستقلی تفکیک شوند (به عنوان مثال فرایند خرید در فروشگاه آنلاین). مباحثی با پیچیدگی مطلقا کامل ([Meyer۲۰۱٤] را مشاهده کنید) برای توسعه تدریجی مناسب نیستند، زیرا هر بینش جدید در مورد یک موضوع منجر به درک کاملاً جدیدی از اطلاعات شناخته شده خواهد شد. در اینجا، مثال ها شامل محاسبات با پارامترهای ورودی پیچیده و پارامترهای خروجی ساده (مانند بیمه نامه ها یا اجزای کنترل موتور) است.

دام- توسعه تدریجی ممکن است نوآوری بنیادی (Radical) یا تحول آفرین (Disruptive) را تشویق نکند: فرایند تدریجی چابک ممکن است توسعه ایده‌های نوآورانه و یا تحول آفرین را تشویق نکند، زیرا یک محصول کاری خاص (به‌عنوان‌مثال نرم‌افزار یا ویژگی/ عملکرد نرم‌افزار) پس از تعریف شدن تنها در همان قسمت از برنامه (Locally) بهبود می‌یابد (به‌عنوان‌مثال رفع خطاها یا افزودن عناصر گمشده). اگرچه بیانیه چابک صریحاً از تغییر استقبال می‌کند، اما فرایندهای تدریجی چابک به طور معمول از نوآوری مداوم برای محصولات و خدمات پشتیبانی می‌کنند. نوآوری‌های بنیادی یا تحول¬آفرین از طریق در نظر گرفتن ایده‌های متعدد و ترکیب مجدد ایده‌های موجود پدیدار می‌شوند [LiOg۲۰۱۱]. توسعه ایده های جایگزین از نظر نرم افزاری معمولاً به عنوان هدر رفت (Waste) در محیط های چابک در نظر گرفته می شود (به اصل ۱۰ مراجعه کنید - حداکثر کار انجام نشده). برای نوآوری های بنیادی یا مختل کننده، باید از اقدامات اضافی مانند ایده های استارتاپ ناب یا روش های تفکر دیزاین ارائه شده در۱٫۵ استفاده کرد.

مهم‌ترین و اصلی‌ترین دام - چابک و تغییر فرهنگ با هم همخوانی ندارند: ارزش‌های چابک باعث ایجاد تغییر در نحوه عملکرد سازمان‌ها، ازبین‌رفتن مالکیت برخی از محصولات قابل‌تحویل و مسئولیت جمعی می‌شوند. چابک، بازنگری مستمر در رفتار تیم را برای بهبود روش کار خود ارتقا می‌دهد: تغییر مداوم تیم و سرانجام کل سازمان اجتناب‌ناپذیر است. چنین تغییرات فرهنگی (سازمانی) هم به زمان و هم به افراد واجد شرایط نیاز دارد تا تیم‌ها را در مسیر جدید هدایت کنند.

۱٫۵ RE@Agile و کار مفهومی (L1)

توسعه چابک در جهان مهندسی نرم‌افزار برای رفع چالش‌هایی که از دنیای خارج از مهندسی نرم‌افزار نمایان شدند، به وجود آمد (۱٫۱ را مشاهده کنید). با این وجود، دنیای مهندسی نرم افزار این چالش ها را به طور انحصاری تجربه نکرده است. شاخه های دیگر صنعت و جامعه از چالش های مشابهی مانند مشتری مُصِر (Demanding customer) و چرخه نوآوری سریع تر رنج می بردند. زمینه های دیگر، رویکردهایی را برای کار مفهومی (مانند ایجاد مفاهیم یا

مشخصات سیستم ها) توسعه دادند که کاملاً شبیه به توسعه چابک بود. چندین مورد از آن ها به ویژه از دیدگاه مهندسی نیازمندی ها، برای توسعه نوآوری ها و چشم اندازهای محصول بسیار مفید هستند. در اینجا، آن ها مختصراً از طریق بحث در مورد ارتباطشان با تفکر چابک معرفی می شوند (۱٫۲ را مشاهده کنید). در ادامه، ما سه روش را به عنوان نمونه ای برای چابکی در کارهای مفهومی ارائه خواهیم داد.

تفکر دیزاین ([LiOg۲۰۱۱] ، [Dsch۲۰۱۵] را مشاهده کنید) روشی برای حل مشکلات به اصطلاح دشوار (Wicked) (به عنوان مثال مشکلاتی که به طور ضعیف تعریف شده اند) است. از دیدگاه مهندسی نیازمندی ها، تفکر دیزاین ترکیبی از تکنیک های استخراج و اعتبار سنجی است. در مرکز این روش (الف) یک تیم چند عملکردی است که روی مسئله کار می کند و نمایانگر طیف گسترده ای از دانش لازم برای حل مسئله است. (ب) محیط کاری که تیم بتواند روی ایده ها کار کند. و (ج) یک فرآیند تکرار شونده که شامل مراحل زیر است:

- همدلی: در این مرحله، تیم همدلی را توسعه می‌دهد تا به درک افراد مرتبط با مسئله قابل‌حل بپردازد.
- تعریف: در این مرحله، تیم با استفاده از بیان مجدد مسئله، درک مشترکی درباره جزئیات مسئله‌ای که باید حل شود، به دست آورد.
- ایده‌پردازی: در این فاز، تیم بر ایده‌پردازی تمرکز دارد. هدف در اینجا توسعه ایده نیست. در عوض تیم تا آنجا که ممکن است تعداد بسیاری از ایده‌ها را توسعه می‌دهد. در پایان این مرحله، تیم امیدوارکننده‌ترین ایده‌ها را برای نمونه‌سازی اولیه انتخاب می‌کند.
- نمونه‌سازی اولیه: در این مرحله، تیم نمونه‌های اولیه بسیار ساده‌ای (نه لزوماً نرم‌افزاری!) را از ایده‌های توسعه‌یافته ایجاد می‌کند. اصل در اینجا این است که نمونه اولیه باید تا حد ممکن واقع‌بینانه و ارزان باشد.
- تست: در این مرحله، تیم نمونه اولیه را با مشتریان واقعی تست می‌کند تا از ایده‌های آن‌ها بازخورد بگیرد. یک اصل بنیادی برای مرحله آزمون "نشان دهید، نگویید" (show, not tell) است، به این معنی که نمونه اولیه باید کاملاً گویا باشد تا کاربر بتواند بازخوردی واقعی ارائه دهد.

مراحل تفکر دیزاین مقیاس‌پذیر هستند و می‌توانند در پروژه‌هایی از چند روز تا چند هفته انجام شوند. به‌علاوه، مراحل، توالی دقیقی را تعریف نمی‌کنند. هر زمان که لازم باشد، تیم می‌تواند تصمیم بگیرد که به عقب برگردد یا در این روند به جلو برود. با این حساب، تفکر دیزاین با ارزش چابک "پاسخ به تغییر ارجح بر پیروی از یک طرح" مطابقت دارد. نتیجه نهایی یک فرایند تفکر دیزاین، مجموعه‌ای از نمونه‌سازی اولیه است که نشان‌دهنده راهکارهای معتبر و ابتکاری برای مسئله در ابتدا تعریف شده است. ازاین‌رو، می‌توان از تفکر دیزاین برای توسعه ایده‌هایی برای نرم‌افزارهای باارزش استفاده کرد و در نتیجه از اصل اول چابک پشتیبانی می‌کند. همان‌طور که در بالا گفته شد، تیم چند عملکردی و محیط کار، از عناصر اصلی فرایند هستند که با اصل پنجم چابک مطابقت دارد. هدف اصلی فاز نمونه اولیه تولید نمونه‌های ارزان‌قیمت و سبک است که با اصل سادگی چابک مطابقت دارد.

دیزاین اسپرینت [KnZK۲۰۱۶] یک فرایند پنج‌روزه برای توسعه ایده‌ها بر اساس اصول طراحی، نمونه‌سازی اولیه و آزمودن با مشتریان نهایی است. از منظر مهندسی نیازمندی‌ها، دیزاین اسپرینت ترکیبی از تکنیک‌های استخراج و اعتبارسنجی است. پایه این روش نحوه کار زمان - ثابت (Time-boxed) است. در این روش، هر روز به یکی از فعالیت‌های زیر اختصاص‌داده‌شده است: به‌اشتراک‌گذاری دانش تیم، ارائه ایده‌ها، تصمیم‌گیری در مورد اینکه کدام ایده‌ها را در نمونه اولیه ایجاد کنید، ساخت ایده‌های انتخابی در قالب نمونه اولیه و در آخر، آزمودن ایده‌ها با مشتریان واقعی. در اینجا، تفاوت مهم در مقایسه با توسعه چابک این است که نمونه اولیه نیازی به نرم‌افزار ندارد. دانستن این نکته مهم است که اصطلاح "اسپرینت" به اسپرینت اسکرام اشاره نمی‌کند.

استارتاپ ناب [Ries۲۰۱۱] روشی برای توسعه کسب و کارها و مدیریت استارتاپ ها است که در جامعه چابک بسیار پذیرفته شده است. از دیدگاه مهندسی نیازمندی ها، این روش شامل چندین ایده است که بسیار جالب هستند. دو مثال از این روش رویکرد ویژه توسعه محصول (The special product development approach) و کمینه محصول پذیرفتنی(Minimum viable product) است. رویکرد توسعه محصول، ساخت (Build)، اندازه گیری (Measure)، یادگیری (Learn) نامیده می شود و به ویژه بر یادگیری مداوم نیازهای مشتری متمرکز است. کمینه محصول پذیرفتنی (MVP) "نسخه ای از محصول جدید است که به یک تیم امکان می دهد تا با حداقل تلاش، حداکثر میزان آموزش معتبر در مورد مشتریان را جمع آوری کند" [Ries۲۰۱۱]. مفهوم مهم دیگر استارتاپ ناب، چرخش (Pivot) است، "یک اصلاح ساختار یافته برای آزمون فرضیه اساسی جدید در مورد محصول، استراتژی و موتور رشد" [Ries۲۰۱۱]. از دیدگاه مهندسی نیازمندی ها، این ایده ها ترکیبی از تکنیک های استخراج و اعتبار سنجی هستند. به جای استخراج و اعتبار سنجی نیازمندی های مبتنی بر مفاهیم یا مستندات، استخراج و اعتبار سنجی با محصول واقعی انجام می شود که مطابق با صحبت Ries، تحت شرایط عدم اطمینان شدید ترجیح داده می شود. Ries تأکید می کند که MVP نباید لزوماً یک نرم افزار کاملاً کاربردی و کامل باشد. در عوض، این کتاب به یک صفحه وب بسیار ساده برای فروش کفش با فرایند حمل دستی اشاره کرده است تا نیاز به خرید آنلاین کفش را تأیید کند.

این رویکردها نشان می‌دهد که چابک واقعاً چیزی بیش از اسکرام است. نمونه‌هایی از تفکر دیزاین، دیزاین اسپرینت و استارتاپ ناب نشان می‌دهد که رویکردهایی برای کار مفهومی در RE@Agile وجود دارد که تفکر چابک را دارند و بنابراین با سازمانی که می‌خواهند نرم‌افزار را به روش چابک توسعه دهند سازگاری کامل دارند. این رویکردها و سایر رویکردها نباید به‌عنوان شکل جدیدی از رویکرد آبشاری یا تفکر مقدماتی (Upfront thinking)، در نظر گرفته شوند. آن‌ها نه‌تنها این قابلیت را دارند، بلکه باید در چارچوب توسعه چابک (به‌عنوان‌مثال در یک یا چند تکرار) برای طراحی جنبه‌های اختصاصی یک سیستم استفاده شوند. از سوی دیگر، می‌توان آن‌ها را به‌عنوان فعالیت‌هایی که مقدم بر توسعه چابک هستند (به‌عنوان‌مثال یک مرحله سریع تفکر مقدماتی (Upfront thinking))، استفاده کرد. بدین ترتیب، این رویکردها به غلبه بر پتانسیل محدود نوآوری در توسعه چابک کمک می‌کنند.

۲ مبانی RE@Agile (L2)

مدت‌زمان: ۲ ساعت و نیم

اصطلاحات: مالک محصول، بک‌لاگ محصول، بک‌لاگ اسپرینت، اپیک ها، داستان‌های کاربر، نقشه‌های داستان

اهداف آموزشی

هدف آموزشی ۲٫۱٫۱	دانستن نمونه‌هایی از روش‌های چابک
هدف آموزشی ۲٫۲٫۱	دانستن اسکرام به عنوان یک مثال: نقش‌ها ، فرایندها ، مصنوعات و ارتباط آن‌ها با مهندسی نیازمندی‌ها
هدف آموزشی ۲٫۲٫۲	دانستن مسئولیت‌های یک مالک محصول اسکرام
هدف آموزشی ۲٫۲٫۳	دانستن مفهوم بک‌لاگ محصول و تفاوت‌های آن با مستند مشخصات نیازمندی‌های محصول
هدف آموزشی ۲٫۳٫۱	درک تفاوت بین مهندس نیازمندی‌های کلاسیک و مالک محصول اسکرام
هدف آموزشی ۲٫٤٫۱	درک دلایل خوب برای این مورد که چرا مهندسی نیازمندی‌ها باید بخشی از یک روند مداوم باشد.
هدف آموزشی ۲٫۵٫۱	دانستن توسعه مبتنی بر ارزش (به عنوان مثال اولویت‌بندی نیازمندی‌ها)
هدف آموزشی ۲٫۵٫۲	دانستن اینکه ارزش، مدیریت ریسک و فرصت است
هدف آموزشی ۲٫۵٫۳	دانستن نمونه‌هایی از ارزش در سازمان‌های انتفاعی و غیرانتفاعی
هدف آموزشی ۲٫٦٫۱	دانستن نحوه ساده‌سازی فرایند تعریف محصول و چگونگی تعریف حداقل محصول پذیرفتنی
هدف آموزشی ۲٫۷٫۱	دانستن ارزش فرلیندهای مداوم و منحنی یادگیری آن‌ها

۲٫۱ رویکردهای چابک (مرور کلی) (L1)

متدهای بسیاری برای به‌اشتراک‌گذاری ارزش‌های بیانیه چابک توسعه‌یافته است. این واحد آموزشی برای دیدن تنوع رویکردها، نمای کلی برخی از آن‌ها را به شما ارائه می‌دهد. این لیست جامعیت نداشته و روش‌ها را از نظر مهندسی نیازمندی‌ها موردبحث قرار می‌دهد.

کریستال (Crystal)، خانواده‌ای از متدها است که توسط آلیستر کابِرن توسعه داده شده است [Cock۱۹۹۸]. وی پیشنهاد می کند که هر پروژه به مدل فرایند متناسب با خود نیاز دارد. بسته به‌اندازه، پیچیدگی و اهمیت آن، آلیستر نقش‌ها، فعالیت‌ها و محصولات کاری کافی را پیشنهاد می‌کند. کریستال کِلیر(Crystal Clear) - روشی برای پروژه‌های کوچک و بااهمیت کمتر - بسیار شبیه XP است. کریستال نارنجی (Crystal Orange) و کریستال قرمز (Crystal Red) کمی فرم‌گرایی بیشتری برای کنارآمدن با پروژه‌های بزرگ‌تر ایجاد می‌کنند. از دیدگاه نیازمندی‌ها، (در کنار تمام موارد دیگر) آلیستر پیشنهاد می‌کند تا با مدل‌های مورد کاربرد (Use cases) و موکاپ‌های (Mock-ups) با مستندسازی و رسمیت کم (Low-ceremony) کار شود.

توسعه ناب [Popp۲۰۰۳] **(Lean Development)** و کانبان **(Kanban)** بر اساس اصولی هستند که برای اولین‌بار در دهه ۱۹۴۰ در تولید خودرو استفاده شد. آن ها برای پروژه های فناوری اطلاعات در حوزه متدهای چابک، سازگار شده اند. آن ها در تلاش هستند تا ۷ نوع هدر رفت (Waste) را در فرآیند تولید کشف کنند (محصولات متوسط ناتمام، تولید بیش از حد، نقص، ...) و برای تسریع در تحویل نهایی، هرکدام را به تدریج از بین ببرند.

اسکرام [Scrum۲۰۲۰] چارچوبی برای توسعه و نگهداری راهکارهای ارزشمند به عنوان یک محصول در محیط های پیچیده است. این چارچوب بر توسعه تکرارشونده و تدریجی مبتنی بر تجربه گرایی و تفکر ناب متمرکز است. این چارچوب تنها سه نقش اساسی (در اسکرام به عنوان مسئولیت نام برده شده اند [Scrum۲۰۲۰]) را مشخص می کند: یک مالک محصول (برای مدیریت بک لاگ محصول، به عنوان مثال تعریف چشم انداز و تمام نیازمندی های مرتبط به یک محصول)، توسعه دهندگان برای توسعه این نیازمندی ها در اسپرینت های کوتاه (اصطلاحی برای Iterationها در اسکرام) و یک اسکرام مستر برای راهنمایی و تسهیل گری پیاده سازی اسکرام برای تیم اسکرام و سازمان توسعه دهنده. ما در بخش بعدی در مورد اسکرام با جزئیات بیشتری بحث خواهیم کرد.

تی.دی.دی (توسعه مبتنی بر آزمون) [Beck۲۰۰۳] بر اساس ایده نوشتن آزمون در ابتدا، قبل از کدنویسی ویژگی مربوطه است. موردهای آزمون (Test cases) مشخصات و نیازمندی‌های دقیقی هستند که محصول باید برآورده کند.

XP (برنامه‌نویسی اکستریم) [Beck۲۰۰٤]: XP بر برقراری ارتباط مستقیم بین مشتری و یک برنامه‌نویس ("مشتری در محل" که درست کنار برنامه‌نویس نشسته است، دائماً در مورد نیازمندی‌ها بحث می‌کند و به ارائه بازخوردهایی در قالب ویژگی‌های قابل‌پیاده‌سازی می‌پردازد) تأکید می‌کند.

۲٫۲ اسکرام (به همراه پرکتیس‌های خوب) به‌عنوان‌مثال (L1):

اسکرام محبوب‌ترین و پذیرفته‌شده‌ترین چارچوب چابک است. اسکرام یک چارچوب سبک‌وزن است که به افراد، تیم‌ها و سازمان‌ها کمک می‌کند تا از طریق راه‌حل‌های تطبیقی برای مشکلات پیچیده، خلق ارزش کنند. راهنمای اسکرام [Scrum۲۰۲۰]، به ارائه تعریفی از اسکرام و مؤلفه‌های اساسی آن می‌پردازد.

در این متد، بر اساس تعریف [Scrum۲۰۲۰]، موارد زیر وجود دارند:

- سه مسئولیت (اسکرام مستر، مالک محصول و توسعه‌دهنده). این سه در مجموع به‌عنوان تیم اسکرام نیز شناخته می‌شوند.
- پنج رویداد (اسپرینت، برنامه‌ریزی اسپرینت، اسکرام روزانه، بررسی اسپرینت، بازاندیشی اسپرینت).

- سه مصنوع [1] و تعهدات مرتبط آن (بک‌لاگ محصول، بک‌لاگ اسپرینت، اینکریمنت)

اسکرام هیچ روش مهندسی را توصیه نمی‌کند.

اسکرام پیشنهاد می‌کند محصولات را به‌صورت تکرارشونده و تدریجی در مجموعه‌ای از اسپرینت‌ها شامل تکرار (Iteration) در بازه زمان - ثابت یک‌ماهه یا کمتر توسعه دهید. هر اسپرینت منجر به یک اینکریمنت قابل‌استفاده می‌شود که یک پله‌ی روبه‌جلوی مشخص به سمت هدف محصول است. در اینجا، این تصمیم مالک محصول است که آیا این اینکریمنت را به مشتری تحویل دهد، یا به دلیل مهمی (مانند خطرات احتمالی) از این کار صرف‌نظر کند.

اسپرینت (Sprint)

اسپرینت محرک اساسی برای توسعه است، زیرا یک فرایند تکرار شونده برنامه ریزی-انجام-بررسی-عمل کردن (Plan-Do-Check-Act) است که چرخه های بازخورد کوتاه را امکان پذیرمی کند. هر اسپرینت با برنامه ریزی اسپرینت آغاز می شود، رویدادی که در آن تیم اسکرام برای تعریف آنچه می تواند در اینکریمنت بعدی محصول توسعه داده شود و نحوه دستیابی به آن (تجزیه کردن) همکاری می کند. منشأ این برنامه از بک لاگ محصول (لیستی مرتب و پویا از همه مواردی که ممکن است در محصول مورد نیاز باشد) بر گرفته شده است. بک لاگ محصول برخلاف مستند مشخصات نیازمندی ها در مهندسی نیازمندی های کلاسیک، یک مجموعه ای کامل از همه نیازمندی ها نیست. اغلب (مخصوصاً در ابتدای توسعه محصول)، بک لاگ محصول فقط شامل فهرستی از ایده های تقریباً فرموله شده است که در طول زمان توسعه یافته و به نیازمندی های ملموس تبدیل می شوند. در صورتیکه اولویت یک عنصر از بک لاگ محصول بیشتر باشد و زمان اجرای آن نزدیک‌تر باشد (مثلاً در یکی از دو اسپرینت بعدی)، توصیف آن عنصر از بک لاگ محصول دقیق‌تر است. مالک محصول در قبال بک لاگ محصول که در ساختار فعلی آن توسط هدف واقعی محصول هدایت می شود، مسئول است. برای جزئیات بیشتر به ۳٫۱٫۱، مستند مشخصات (Specification Documents) در مقابل بک‌لاگ محصول (Product Backlog) مراجعه کنید.

برنامه‌ریزی اسپرینت که برای انتخاب عناصری که در اسپرینت بعدی اجرا خواهند شد انجام می‌شود، به نتایج زیر منتهی می‌شود: هدف اسپرینت (چرایی) و بک‌لاگ اسپرینت (چیستی، برای مثال انتخاب آیتم‌ها و تجزیه آنها برای اسپرینت) و برنامه اسپرینت (چگونگی دستیابی).

متعاقباً، *تیم اسکرام* شروع به پیاده‌سازی *اینکریمنت محصول* می‌کند. *توسعه‌دهندگان* مسئول تبدیل موارد انتخاب شده به نتایج ملموس هستند. آن‌ها با مالک محصول برای اصلاح *موارد بک‌لاگ محصول* و به اشتراک گذاشتن

[1] محصولات کاری مطبق با ترمینولوژی CPRE [Glinz۲۰۲۲].

بازخورد درباره پیاده‌سازی فعلی، همکاری می‌کنند. در جهت ارزیابی پیشرفت و به‌روزرسانی بک‌لاگ اسپرینت، توسعه‌دهندگان هر روز در طول جلسه اسکرام روزانه همگام می‌شوند.

کار *توسعه‌دهندگان* باتوجه‌به "تعریف انجام شده" هدایت می‌شود: تعریفی از آنچه برای کامل‌شدن اینکریمنت باید به دست بیاید. "تعریف انجام شده" به ذی‌نفعان کمک می‌کند تا بدون شک پیشرفت محصول را درک کنند.

وقتی زمان - ثابت اسپرینت به اتمام می‌رسد (بین ۱ هفته تا ۱ ماه)، اینکریمنت محصول در رویداد *بررسی اسپرینت* توسط *تیم اسکرام* و ذینفعان اصلی بازرسی می شود تا نتایج اسپرینت را ارزیابی کنند و در مورد آنچه در آینده می توان انجام داد، بحث کنند. بک لاگ محصول به همین ترتیب به روز می شود.

اسپرینت با رویداد *بازاندیشی اسپرینت* به پایان می‌رسد که طی آن تیم اسکرام این موارد را بررسی می‌کند که چگونه می‌تواند کارآمدتر شود، چه چیزی خوب کار می‌کند و چه چیزی می‌تواند بهبود یابد. بلافاصله پس از پایان *بازاندیشی اسپرینت*، اسپرینت فعلی جمع‌بندی‌شده و اسپرینت بعدی شروع می‌شود.

پرکتیس‌های خوب (Good practices):

حتی اگر اسکرام با تکنیک‌هایی مهندسی نیازمندی‌ها همراه نباشد، جامعه چابک به ارائه برخی اقدامات خوب می‌پردازد که مناسب اسکرام هستند.

راهنمای اسکرام از اصطلاح آیتم‌های بک‌لاگ محصول برای موارد ذکر شده در بک‌لاگ محصول استفاده می‌کند. این یک اصطلاح عمومی برای شناسایی هر نوع اطلاعات در مورد محصول درحال‌توسعه است، اما بسیاری از موارد موجود در محصول در واقع نیازمندی‌ها هستند یا می‌توانند برای تبدیل‌شدن به نیازمندی مورداستفاده قرار گیرند.

جامعه چابک موارد زیر را پیشنهاد می‌کند:

- تجزیه نیازمندی‌ها به: اپیک، ویژگی‌های اختیاری و داستان‌های کاربر (سطح دانه‌بندی)، [CPREALAGILE۲۰۲۲] را مشاهده کنید.
- تمایز بین نیازمندی‌های عملکردی (توانایی)، نیازمندی‌های کیفی (رفتار) و قیدها (محیط)
- گروه‌بندی نیازمندی‌ها با استفاده از تم‌ها (Themes)

یک پرکتیس خوب (اضافی) برای توصیف مشخصات بک‌لاگ محصول ایجاد شده است: بک‌لاگ باید "DEEP" (جزئیات مناسب (Detailed)، تخمین زده (Estimated)، در حال تکامل (Emergent)، اولویت‌بندی شده (Prioritized)) باشد [Cohn۲۰۰٤]. در اسپرینت، آراستن (Grooming) و پالایش (Refinement) بک‌لاگ که بخشی اساسی در فرایند توسعه چابک است، وجود دارند.

با استفاده از "تعریف انجام شده" (DoD)، تیم درک مشترکی راجع به کامل‌شدن یک آیتم از بک‌لاگ محصول و آماده‌شدن آن برای تبدیل به بخشی از اینکریمت قابل‌استفاده را توسعه می‌دهد [Scrum۲۰۲۰].

یک پرکتیس خوب دیگر، استفاده از قاعده INVEST است [Wake۲۰۰۳]. این کلمه اختصاری، معیارهای زیر را پوشش می‌دهد:

- **I**: مستقل از یکدیگر (Independent of each other)
- **N**: قابل‌مذاکره (Negotiable)
- **V**: ارزشمند (Valuable)
- **E**: قابل تخمین (Estimable)
- **S**: به‌اندازه کافی کوچک که در یک اسپرینت جای گیرد (Small enough to fit into one sprint)
- **T**: قابل آزمون (Testable)

مهندسی نیازمندی‌ها معیارهای مربوطه را برای نیازمندی‌های خوب ارائه می‌دهد ([CPREFL۲۰۲۲] را مشاهده کنید):

- توافق شده (agreed)
- شفاف (unambiguous)
- ضروری (necessary)
- هم‌خوان (Consistent)
- قابل‌تأیید (Verifiable)
- امکان‌پذیر (feasible)
- قابل‌ردیابی (traceable)
- کامل (complete)
- قابل‌درک (understandable)

۲٫۳ تفاوت‌ها و اشتراک‌های میان مهندسین نیازمندی‌ها و مالکان محصول (L2)

پس از بحث در مورد اصول چارچوب اسکرام، بیایید نقش مهندس نیازمندی‌های سنتی را با نقش مالک محصول مقایسه کنیم. لازم به ذکر است که نقش مالک محصول در این میان توسط بسیاری از رویکردهای چابک (همچنین در خارج از اسکرام) پذیرفته شده است؛ بنابراین، توضیح زیر لزوماً فقط برای اسکرام قابل‌درک و اعمال نیست.

فعالیت‌های اصلی یک مهندس نیازمندی شامل موارد زیر می‌شود [CPREFL۲۰۲۲]:

- استخراج نیازمندی‌ها
- مستندسازی نیازمندی‌ها
- اعتبارسنجی و مذاکره نیازمندی‌ها
- مدیریت نیازمندی‌ها

همان‌طور که در بالا توضیح داده شد، مسئولیت‌های اصلی یک مالک محصول عبارتند از:

- اطمینان از اینکه تیم ارزش ثابت (تجاری) را ارائه می‌کند: این بدان معناست که مالک محصول باید بینش بلندمدت محصول (هدف محصول، چشم‌انداز محصول) را با نیازهای کوتاه‌مدت متعادل کند، باید آیتم‌های بک‌لاگ محصول را طبق معیارهای تعریف‌شده اولویت‌بندی کند و نتایج را در پایان هر تکرار (اسپرینت) به همراه ذی‌نفعان مورد بازرسی قرار دهد.
- مدیریت تمامی ذی‌نفعان: مالک محصول در مورد ارسال نیازمندی‌های هم‌خوان به تیم پاسخگو است. او باید نیازمندی‌ها را از تمامی ذی‌نفعان جمع‌آوری کند و مطمئن شود که مغایرتی با یکدیگر ندارند. هر گونه اختلاف بین ذی‌نفعان باید حل‌وفصل شود تا توسعه‌دهندگان از چنین اختلافاتی رهایی پیدا کنند.
- اقلامی را که دارای بیشترین ارزش (تجاری) هستند، به طور مستمر به تیم ارائه دهید: دانه‌بندی این نیازها باید به‌اندازه کافی کوچک باشد که در یک تکرار (Sprint) قرار گیرند. برای هر گونه سؤال پس از جلسه برنامه‌ریزی اسپرینت، مالک محصول باید در دسترس باشد تا به‌سرعت شفاف‌سازی کند.

با مقایسه این دو نقش، معلوم می‌شود که هر دو، مهندسین نیازمندی و مالکان محصول، باید وظایف اصلی استخراج، مستندسازی، اعتبارسنجی و مدیریت نیازمندی‌ها را (همراه با سایر ذی‌نفعان) انجام دهند. بااین‌حال، معمولاً نمادها (Notations) و ابزارهای استفاده شده در محیط‌های چابک کمتر رسمی هستند:

- برای مثال، کارت‌های داستان (Story cards) به‌جای مستندات نیازمندی‌ها
- مکالمه بیشتر و نوشتن کمتر
- تأکید بیشتر بر وضعیت فعلی نیازمندی‌ها، تأکید کمتر بر نسخه‌بندی (Versioning) و تاریخچه (History)

نقش مالک محصول، درحالی‌که همچنان مسئولیت کلی در مورد نیازمندی‌های باکیفیت بالا را حفظ می‌کند، نسبت به مهندس نیازمندی‌های سنتی گسترده‌تر است، زیرا او مسئول موفقیت محصول به‌عنوان یک کل است و به طور مداوم به جمع‌آوری نظرات از کسب‌وکار می‌پردازد و به همین ترتیب، اقدام به آپدیت و اولویت‌بندی بک‌لاگ می‌کند.

باتوجه‌به فعالیت‌های مهندسی نیازمندی‌ها، می‌توان به طور خلاصه بیان کرد که مالک محصول مسئولیت اجرای آنها را بر عهده دارد. او می‌تواند توسط افراد باتجربه در مهندسی نیازمندی‌ها پشتیبانی شود یا خودش این وظایف را انجام دهد. بااین‌حال، مسئولیت نتیجه مهندسی نیازمندی‌ها و همچنین مسئولیت بک‌لاگ محصول بر عهده مالک محصول است.

۲٫٤ مهندسی نیازمندی‌ها به‌عنوان یک فرایند مداوم (L2)

بنابراین، بیش از اینکه مهندسی نیازمندی‌ها یک مرحله مشخص در طول توسعه در چابک باشد، یک فعالیت تکرارشونده و مداوم است. هدف این نیست که قبل از شروع طراحی و اجرا، همه نیازمندی‌ها استخراج و تحلیل شوند. نیازمندی‌ها و محصولات هر دو به‌صورت تکرارشونده و تدریجی ایجاد می‌شوند.

بنابراین، تا زمانی که خود محصول ادامه داشته باشد، مهندسی نیازمندی‌ها به‌عنوان یک فعالیت مداوم در نظر گرفته می‌شود. بااین‌وجود، این فرایند نتایج میانی کاملاً مشخصی را به همراه دارد: نیازمندی‌های پیش‌بینی‌شده که بیشترین ارزش (تجاری) را مبتنی بر معیارهای تعریف شده نوید می‌دهند، باید "آماده برای اجرا" (Ready for implementation) باشند (به معنای "تعریف آماده" (Definition of ready) که در بالا توضیح داده شد). سایر نیازمندی‌هایی که فوریت کمتری دارند، تنها پس از تکمیل موارد ضروری اصلاح می‌شوند.

"فرایند مداوم" (Continuous process) مانع از اجرای برخی فعالیت‌های مهم پیشین نمی‌شود. حتی اگر نیازمندی‌ها بر اساس "نیاز به دانستن" مشخص شوند، برخی از جنبه‌های این نیازمندی‌ها وجود دارند که باید در اوایل چرخه زندگی مورد توجه قرار گیرند. به‌عنوان‌مثال می‌توان به تعریف چشم‌اندازها یا اهداف، شناخت ذی‌نفعان و تعیین دامنه محصول اشاره کرد. شروع اجرا بدون انجام چنین فعالیت‌هایی میزان خطر را به طور قابل‌توجهی بالا می‌برد.

۲٫۵ توسعه مبتنی بر ارزش (L1)

متدهای چابک تلاش می‌کنند تا به طور مداوم، ارزش (تجاری) را به کاربر نهایی برسانند. اغلب ارزش می‌تواند مستقیماً با اصطلاحات مالی، افزایش سهم بازار یا رضایت مشتری بیان شود. این روش اغلب در سازمان‌های انتفاعی استفاده می‌شود، اما نحوه تعریف ارزش برای سازمان‌های غیرانتفاعی کمتر مشخص است. در اینجا، معیارهایی مانند میزان استفاده یا شاخص شادی (Happiness index) دررابطه‌با یک محصول (به‌عنوان‌مثال نرخ کلیک در وب‌سایت‌ها یا کمک‌های مالی یک سازمان غیرانتفاعی) ممکن است بیشتر مرتبط باشند.

نوع متفاوتی از ارزش ریسک خطر است. رویکردهای خوب چابک سعی می‌کنند ارزش تجاری و کاهش ریسک را نسبت به تکرارها متعادل کنند.

همان‌طور که در بخش بعدی توضیح داده شده است، به‌منظور تعیین اینکه کدام‌یک از نیازمندی‌ها به ارزش مطلوب منجر می‌شوند، متدهای چابک اغلب برای رسیدن به کمینه محصولات پذیرفتنی (MVPs) یا کمینه محصولات قابل‌عرضه (MMPs) تلاش می‌کنند.

۲٫۶ سادگی به‌عنوان مفهومی ضروری (L1)

در یک‌دنیای پیچیده، سادگی راهی برای پذیرش پیچیدگی با فرایند زیر است:

- ایجاد یک پاسخ ساده و ناقص بالقوه (Potentially incomplete) به یک مسئله، در نتیجه ایجاد یک مقدار کوچک از ارزش،
- به‌دست‌آوردن توانایی یادگیری بیشتر در مورد زمینه (Context)، بر اساس تجربه دنیای واقعی،
- تطبیق و تکرار در ایجاد و ارائه ارزش با سرعت پایدار،

- صرفه‌جویی در منابع از یک ایده غیرسودآور، برای تخصیص مجدد آنها در یک ایده جدید با شکست سریع و یادگیری سریع.

سادگی از جهاتی با «کمال» (Perfection) مخالف است. این موضوع بیان می‌شود که سادگی به معنای "کیفیت پایین" نیست؛ بلکه به معنای "حداقل دامنه" (Minimal scope) یا "حداقل خدمات" (Minimal service) است - اما همیشه باکیفیت بالا. کیفیت قابل‌مذاکره نیست.

بیانیه چابک سادگی را به‌عنوان «هنر به حداکثر رساندن میزان کار انجام نشده» توصیف می‌کند. این به معنای پرهیز از هر گونه تلاشی برای کاهش حجم کار نیست، بلکه بیشتر به معنای شفاف‌بودن در مورد کاری که باید انجام شود، ایجاد ارزش و ارائه اهداف خاص است. اگر تیم متوجه نیازمندی‌های ناهمسو شود، آنگاه تعداد کار بی‌ارزش را به حداکثر خواهد رساند - که باید از آن جلوگیری کرد تا هزینه‌ها و زمان را بدون دلیل موجه از دست ندهد.

اغلب، دو نوع محصولات ساده تعریف بدین صورت تعریف می‌شوند: MVPها و MMPها.

کمینه محصول پذیرفتنی (MVP) مفهومی از استارتاپ ناب است ([Ries۲۰۱۱] را مشاهده کنید) و به‌عنوان کوچک‌ترین محصولی تعریف می‌شود که می‌تواند تجربه کاربر نهایی (End user experience) را ایجاد کند و به برای تیم بازخورد فراهم کند. این بازخورد ورودی اصلی برای تکامل محصول است. زیرا این روش بازگشت سریع سرمایه با سطح ریسک پایین را امکان‌پذیر می‌کند و بسیاری از استارتاپ‌ها با آن هماهنگ هستند.

هدف از کمینه محصولات قابل بازاریابی (MMP) چیزی فراتر است. مسئله نه‌تنها ارائه بازخورد زودهنگام و در نتیجه هدایت مراحل بعدی نیازمندی‌ها است، بلکه ارزش‌آفرینی سریع است. بسیاری از محصولات را می‌توان در یک "نسخه ۱" ساده مورداستفاده قرارداد بدون اینکه از قبل از همه ویژگی‌ها و کیفیت‌های موردنظر برخوردار باشد، بنابراین از این طریق درآمد کسب می‌کند تا هزینه بهبود مستمر محصول پرداخت شود. در اینجا اغلب فرایند استارتاپ ناب (Lean Startup) (ایجاد، اندازه‌گیری، یادگیری) به‌صورت تکرارشونده مورداستفاده قرار می‌گیرند.

۲٫۷ بازرسی و سازگاری (L1)

متدهای چابک بر اهمیت بازخورد سریع و مکرر تأکید دارند. بعد از هر بار تکرار (گاهی اوقات حتی بیشتر)، تیم باید درمورد اینکه آیا روند توسعه برای آن‌ها به‌خوبی پیش رفته یا نیاز به بهبود دارد، بحث کند.

در این فرایند بازخورد، همه باید در مراحل اولیه روند توسعه فعلی را بررسی کنند. تیم باید متدهای استفاده شده، ابزارها، همکاری در تیم و سایر موارد دیگر را به چالش بکشد. از همه درخواست می‌شود به سؤالاتی مانند این پاسخ دهند: چه چیزی خوب کارکرد؟ چه چیزی خوب پیش نرفت؟ در تکرار بعدی چه چیزهایی را باید امتحان کنیم؟

مهم است که محصول یا راهکار در پایان هر تکرار بررسی شود. تیم باید به‌سرعت (Velocity) خود نگاه کرده و سرعت برنامه‌ریزی‌شده برای تکرار بعدی را تغییر داده یا تنظیم کند.

این امر همچنین در فرایند نیازمندی‌ها اعمال می‌شود. پیامدهای این بینش‌ها باید شامل انطباق کوتاه‌مدت (Short-term adaptation) با مراحل بهبود فرایند باشد.

۳ محصولات کاری و تکنیک‌ها در RE@AGILE (L2)

مدت‌زمان: ۲ ساعت و نیم

شرایط: چشم‌انداز محصول، نقشه راه محصول، بک‌لاگ محصول، بک‌لاگ اسپرینت، داستان کاربر، معیار پذیرش، ویژگی، نیازمندی عملکردی، نیازمندی کیفی، اپیک، مدل زمینه، نقشه داستان، تعریف آماده، تعریف انجام شده

اهداف آموزشی

هدف آموزشی ۳٫۱٫۱	دانستن تفاوت میان مستندات مشخصات سنتی (Traditional specification documents) و بک‌لاگ‌ها (Backlogs)
هدف آموزشی ۳٫۱٫۲	دانستن ارزش چشم‌اندازها و اهداف
هدف آموزشی ۳٫۱٫۳	دانستن ارزش افزوده استفاده از مدل‌های زمینه در RE
هدف آموزشی ۳٫۱٫٤	دانستن نحوه تفاوت قائل شدن میان سه نوع از نیازمندی‌ها
هدف آموزشی ۳٫۱٫٥	درک سطوح مختلف جزئیات (Granularity) در نیازمندی‌ها
هدف آموزشی ۳٫۱٫٦	درک فرمت‌های مختلف مشخصات برای انواع مختلف محصول کاری در فرایندهای چابک (به عنوان مثال متنی (Textual) در مقابل مبتنی بر الگو (Template-based) و نموداری (Diagrammatic))
هدف آموزشی ۳٫۱٫۷	درک ارزش اصطلاحات ، واژه‌نامه‌ها و مدل‌های اطلاعاتی
هدف آموزشی ۳٫۱٫۸	درک مشخصات نیازمندی‌ها و محدودیت های کیفی در فرآیندهای مهندسی نیازمندی‌های چابک
هدف آموزشی ۳٫۱٫۹	دانستن معیار پذیرش و تناسب
هدف آموزشی ۳٫۱٫۱۰	درک کاربرد استفاده از تعریف آماده و تعریف انجام شده در فرایندهای مهندسی نیازمندی‌های چابک
هدف آموزشی ۳٫۱٫۱۱	دانستن تفاوت میان نمونه اولیه، اینکریمنت و اسپایک (Spike)
هدف آموزشی ۳٫۱٫۱۲	دانستن انواع محصول کاری در فرآیندهای RE@Agile (مدل زمینه، اپیک، داستان کاربر، بک لاگ، نقشه راه، نیازمندی، تعریف انجام شده، تعریف آماده)
هدف آموزشی ۳٫۲٫۱	درک چگونگی استخراج نیازمندی‌ها در مهندسی نیازمندی‌های چابک
هدف آموزشی ۳٫۲٫۲	دانستن چگونگی ایجاد و نگهداری بک‌لاگ‌ها در فرآیندهای مهندسی نیازمندی‌های چابک.
هدف آموزشی ۳٫۲٫۳	دانستن چگونگی اعتبارسنجی و مذاکره در مورد نیازمندی‌ها در RE@Agile
هدف آموزشی ۳٫۲٫٤	دانستن نحوه مدیریت نیازمندی‌ها در RE@AGILE

۳٫۱ محصولات کاری در RE@Agile (L2)

۳٫۱٫۱ مستند مشخصات (Specification Documents) در مقابل بک‌لاگ محصول (Product Backlog)

نمی‌توان تنها به ذکر نیازمندی‌ها برای ایجاد محصولات یا راهکارها بسنده کرد، بلکه باید به‌عنوان لیستی مرتب از همه مواردی که ممکن است در محصول موردنیاز باشند، سازماندهی و مستند شوند [Scrum۲۰۲۰]. فارغ از هر متدولوژی، این مجموعه نیازمندی به‌عنوان محصول کاری اصلی مهندسین نیازمندی‌ها، تحلیل‌گران کسب‌وکار، مالکان محصول یا هر شخص مسئول برای مهندسی نیازمندی‌ها در نظر گرفته می‌شود. متدهای مختلف، انواع و نام‌های مختلفی را برای چنین مجموعه نیازمندی‌ها پیشنهاد می‌دهند.

مهندسی نیازمندی‌ها اغلب این محصول کاری را نتیجه فرایند استخراج، مشخصات نیازمندی‌های کاربر، مشخصات نیازمندی‌های سیستم یا مشخصات نیازمندی‌های نرم‌افزار، بسته به اینکه چه کسی آن را می‌نویسد و جزئیات آن را پوشش می‌دهد، نام‌گذاری می‌کند. این مورد لزوماً شامل یک مستند نیست، بلکه فقط مجموعه‌ای از نیازمندی‌ها به هر شکل فیزیکی (کاغذ (Paper)، مخزن (Repository)، پایگاه‌داده (Database) و...) است.

متدهای چابک، به‌ویژه اسکرام، اصطلاح بک‌لاگ محصول را برای این مجموعه کلی نیازمندی‌ها در راستای پیاده‌سازی در آینده (و سایر اطلاعات مرتبط با محصول، ۲٫۱ را مشاهده کنید) تعریف می کنند و همچنین بک لاگ اسپرینت را برای آن دسته از نیازمندی هایی که برای تکرار بعدی انتخاب شده اند (اسپرینت در اسکرام)، تعریف می کنند [Scrum۲۰۲۰]. مجددا، فرم فیزیکی بک لاگ دارای اهمیت نیست. ممکن است بک لاگ از کارت های ایندکس (Index cards) یا استیکی نوت های روی دیوار تشکیل شده باشند یا در برخی از ابزارهای نرم افزاری مناسب ثبت شوند. اگرچه ممکن است نام و نحوه کار با یکدیگر متفاوت باشد، اما همه محصولات کاری از ایده های یکسانی پیروی می کنند و زمینه ای را برای استخراج، مستندسازی، مذاکرات، اعتبارسنجی و مدیریت نیازمندی ها فراهم می کنند.

ما در ۲٫۲ آموختیم که DEEP یک روش خوب است که به وسیله آن جزئیات بک لاگ محصول به خوبی مشخص شده، تخمین زده شده، تکامل یافته و اولویت بندی می شود. این ویژگی ها برای بک لاگ محصول با اهمیت بوده و به یکدیگر مرتبط یا وابسته هستند. برای مالک محصول به عنوان "بهینه ساز ارزش" (Value optimizer)، رتبه بندی یا مرتب سازی بک لاگ محصول، با ارزش ترین و مهم ترین مورد برای دستیابی است. تخمین از این امر پشتیبانی می کند و همچنین جزئیات مناسب به تمرکز روی بخش های مهم کمک می کند، بنابراین آنها از مرتب سازی (ordering) پشتیبانی می کنند. مستقل از روشی که نیازمندی‌ها مستند می‌شوند، حتما باید عناصر خاصی ثبت شوند. این شامل انواع مختلف نیازمندی ها است: اهداف و چشم اندازها، تعریف دامنه سیستم یا محصول، نیازمندی های عملکردی، نیازمندی ها و محدودیت های کیفی و واژه نامه (به عنوان مثال تعاریف اصطلاحات و اختصارات مربوطه). همانطور که در قبل بحث شد، رویکردها ممکن است در نمادها (Notations)، نحو (Syntax) و سطح

جزئیات برای مشخصات نیازمندی ها متفاوت باشند. نداشتن هیچ گونه مشخصات (به عنوان مثال تنها اعتماد کردن به ارتباط شفاهی بین ذینفعان بدون وجود نیازمندی های کتبی) به طور معمول به عنوان یک گزینه جایگزین شناخته نمی شود، زیرا مستندات مکتوب، اغلب پایه ای برای مذاکره، آزمون پذیرش، اهداف قانونی و موارد دیگر هستند. هرچه تمامی ذینفعان با یکدیگر ارتباط برقرار کنند، ضرورت نوشتن کاهش می یابد، با این حال نتایج و نیازمندی ها هم چنان باید در یک فرم کاملاً شناخته شده (نوشته شده یا رسم شده) ثبت شوند. در پاراگراف های بعدی، بخش های مختلف این مجموعه نیازمندی های کلی، به طور مفصل مورد بحث قرار خواهند گرفت.

۳٫۱٫۲ چشم‌انداز و اهداف

هر فرایند توسعه باید بر اساس چشم‌اندازها یا اهدافی که قابلیت محصول را تعریف می‌کنند، هدایت شود که در صورت پیاده‌سازی، به معنای موفقیت‌آمیز بودن راه‌حل است. داشتن چنین چشم‌اندازها یا اهدافی که دراسرع‌وقت مورد توافق کلیه ذی‌نفعان مربوطه باشد، برای هر فعالیت مربوط به نیازمندی‌های سیستم یا محصول مربوطه از اهمیت بالایی برخوردار است.

در فرایندهای توسعه چابک معمولاً از اصطلاح "چشم‌انداز محصول" استفاده می‌شود تا تأکید شود که هر نتیجه فرایند توسعه باید دارای یک ارزش مشخص مربوط به چشم‌انداز محصول باشد. برای تعریف این ارزش، ممکن است لازم باشد ابتدا به‌وضوح مشخص کنیم که ارزش‌های یک شرکت چیست.

اهداف ذی‌نفعان مختلف می‌تواند متناقض باشد، این بدین معنی است که ذی‌نفعان مربوطه باید برای رسیدن به یک چشم‌انداز یا مجموعه اهداف توافق شده مذاکره کنند. از سوی دیگر، این مورد می‌تواند بیانگر این باشد که انواع مختلفی از محصول (به‌عنوان‌مثال یک آیفون کوچک و بزرگ)، یا حتی به‌طورکلی محصولات مختلف (آیفون و آی پد) موردنیاز است.

توسعه چابک اغلب از اهداف با جزئیات متفاوت مانند اهدافی برای افق‌های زمانی مختلف (Differernt time horizons) یا فواصل برنامه‌ریزی (Planning intervals) استفاده می‌کند. به‌عنوان‌مثال، ممکن است اهداف سالانه جهت مذاکره در مورد تحویل‌دادنی‌ها (زمان و محتوا)، اهداف سه‌ماهه برای برنامه‌ریزی اسپرینت و اهداف تکرار/اسپرینت برای تکرار/اسپرینت بعدی وجود داشته باشند[High۲۰۰۹].

چشم‌اندازهای بلندمدت محصول و اهداف کوتاه‌مدت برای تأکید بر مهم‌ترین دستاوردهایی که باید در یک بازه زمانی خاص به دست آیند مفید هستند و به همسویی کلیه ذی‌نفعان در یک "مأموریت مشترک" کمک می‌کنند. همه این اهداف را می‌توان در یک جدول زمانی در یک نقشه راه مفید نشان داد.

چشم‌انداز یا اهداف محصول، انتزاعی‌ترین شکل نیازمندی‌ها هستند و بدون پالایش بیشتر نمی‌توان آنها را توسعه داد. آن‌ها درک و راهنمایی جامعی برای کل فرایند توسعه چابک ارائه می‌دهند. هر نیازمندی باید در راستای اهداف

بررسی شود تا با این کار، سهم نیازمندی در اهداف مختلف تأیید شود. یک نیازمندی ممکن است بدون ارتباط با مجموعه‌ای از اهداف، شاخصی برای ازدست‌دادن ارزش باشد. در نتیجه، بیانیه چشم‌انداز محصول و اهداف مورد توافق ذی‌نفع، محصولات کاری بسیار مهمی برای موفقیت در فرایندهای توسعه چابک است، زیرا آن‌ها بدون اینکه بی‌جهت خلاقیت توسعه‌دهندگان را محدود کنند، چارچوبی را برای کلیه فعالیت‌های توسعه تعیین می‌کنند.

۳٫۱٫۳ مدل زمینه (Context model)

بیانیه چشم‌انداز به همراه اهداف ذی‌نفعان، باید خواسته‌های کلی سیستم یا محصول را برای تحقق هدف خود مشخص کند. از سوی دیگر، مدل‌های زمینه (Context models) نمایانگر یک دیدگاه متفاوت هستند، زیرا هدف آن‌ها توصیف خصوصیات خاص محیط (زمینه) ای است که سیستم یا محصول در آن کار خواهد کرد.

نیازمندی‌های سیستم یا محصول اغلب با درنظرگرفتن فرضیات مربوط به محیط مشخص می‌شوند. مدل‌های زمینه یک روش ساختاریافته برای مستندسازی فرضیات مربوط به زمینه است. صریح و روشن ساختن چنین فرضیاتی برای ایجاد دیدگاه مشترک و توافق شده در مورد محیط عملیاتی سیستم یا محصول برای کل تیم و سایر ذی‌نفعان مربوطه سودمند است.

در صورت عدم اطمینان، نیازمندی‌های مشخص شده فقط در صورت صحت فرضیات مستند شده در مدل‌های زمینه درست تلقی خواهند شد، به این معنی که مدل‌های زمینه، به نمایش زمینه عملیاتی درست سیستم یا محصول می‌پردازند.

مدل‌های زمینه، محصولات کاری قدرتمندی هستند، زیرا به‌وضوح بین سیستم یا محصولی که قرار است ساخته شود و زمینه آن، به‌عنوان‌مثال شامل سیستم‌های مجاور و کاربران انسانی ([CPREFL۲۰۲۲] را مشاهده کنید)، تمایز ایجاد می‌کنند. با استفاده از این تمایز، عملکرد (functionality) را می‌توان تخصیص داد.

این امر اجازه می‌دهد تا میان مسئولیت‌های سیستم یا محصول و مسئولیت‌های سیستم‌های مجاور یا کاربران انسانی در زمینه که در حال همکاری در طول یک عملیات برای تحقق چشم‌انداز کلی هستند، تمایز قائل شویم؛ بنابراین، از مدل‌های زمینه می‌توان برای شفاف‌سازی و مشخص نمودن رابط‌های خارجی (External interfaces) سیستم یا محصول مورداستفاده نیز استفاده کرد.

چنین مدل‌هایی را می‌توان با استفاده از قالب‌های مختلف، مانند نمودارهای زمینه پیشنهاد شده برای تجزیه‌وتحلیل سیستم ساختاری، استفاده از دیاگرام‌های مورد کاربرد (Use case diagram)، نمودارهای تعریف بلوک سیس.ام.ال (SysML block definition diagram)، نمودارهای مؤلفه‌های یو.ام.ال (UML component diagram) یا نمودارهای کلاس یو.ام.ال (UML Class diagram)، مستندسازی کرد. هر نمادی در صورتی مناسب است که بین سیستم یا محصولی که قرار است ساخته شود و رابط‌های خارجی با اشخاص یا سیستم‌های محیط تفاوت قائل شود.

علاوه بر این باید روابط بین این عناصر و سیستم یا محصول را در سطح مناسب از جزئیات توسعه دهد. نمودارهای ساده جعبه‌ای و خطی دستی (Simple hand-drawn box and lines diagrams) نیز می‌توانند عملی و مفید باشند.

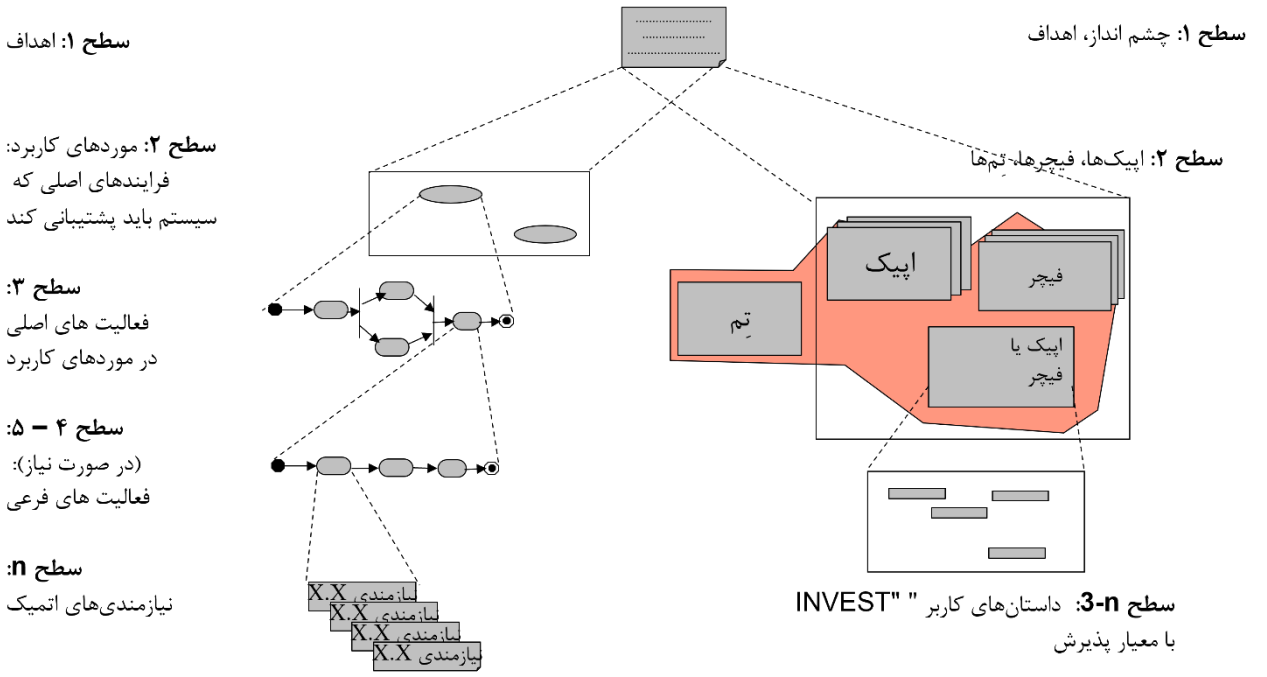
مستقل از فرم مستندات، یک مدل زمینه یک محصول کاری بسیار باارزش است و در طی فرایند توسعه چابک توصیه می‌شود. این مدل، زمانی که رابط‌های خارجی (مرز بین دامنه و زمینه) باید با سیستم‌های مجاور در ارتباط باشند، به تعیین محدوده‌ای (در محدوده حوزه) می‌پردازد که تحلیل‌گران در آن در تصمیم‌گیری آزاد هستند.

۳٫۱٫٤ نیازمندی‌ها

بنابراین، نیازمندی‌ها باید بر اساس چشم‌انداز و اهداف ضبط شده و محدود به مدل زمینه باشند. به طور معمول، بین سه نوع نیازمندی تمایز وجود دارد: نیازمندی‌های عملکردی (Functional requirements)، نیازمندی‌های کیفی (Quality requirements) و محدودیت‌ها (Conflicts).

۳٫۱٫۵ جزئیات نیازمندی‌ها

ذی‌نفعان اغلب نیازهای خود را در سطوح مختلفی از جزئیات اعلام می‌کنند: از نیازهای درشت‌دانه، مانند اهداف کلی کسب‌وکار، تا نیازهای ریزدانه با جزئیات عملکرد سیستم مورد انتظار. نیازمندی‌های عملکردی و کیفی (۳٫۱٫۸ را مشاهده کنید) می‌توانند (و باید) موردبحث و مستندسازی در سطوح مختلف انتزاع قرار گیرند.



شکل 1 : دانه بندی نیازمندی‌ها: تجزیه مورد کاربرد و انواع چابک، موردهای کاربرد، مشخصات مورد کاربرد و دیاگرام‌های فعالیت

یک روش معمول از روش‌های غیرچابک در سمت چپ شکل 1 نشان داده شده است. در این رویکرد، موردهای کاربرد در ابتدا، برای تصحیح چشم اندازها، اهداف و ایجاد یک زیر بخش (Sub-division) از عملکرد سیستم مورد استفاده قرار می گیرند. این تصویر فقط یک مثال برای گروه بندی سطح بالایی از نیازمندی ها را نشان می دهد. نمونه های دیگر شامل گروه بندی براساس ویژگی ها (Features)، اشیا کسب و کار (Business objects)، جریان داده ها (Data flows) یا مولفه های (Components) راهکارهای موجود هستند.

در سطح بعدی دانه‌بندی، هر مورد کاربرد برای شرح مراحل انجام این عملکرد شرح داده شده است. چندین گزینه باتوجه‌به سطح دانه‌بندی در اینجا موجود است:

۱. شرح کلامی (Verbal description) (متن ساده (Plain text))
۲. الگوهای مورد کاربرد (جریان نیمه‌ساختاریافته یا داستانی (Semi-structured format or narrative flows))
۳. دیاگرام فعالیت یو.ام.ال (UML Activity diagram) (یا هر نوع مدل کاربردی گرافیکی دیگر مانند نمودار جریان داده BPMN ،(Data flow diagram)، نمودار زمینه (Context diagram)، نمودار حالت (State diagram)، مدل شی کسب‌وکار (Business object model) و غیره)

به‌عنوان‌مثال، سطح ۳ در شکل 1 موردهای کاربرد تشریح شده با نمودارهای فعالیت یو.ام.ال (UML Activity diagram) را نشان می‌دهد.

بر اساس پیچیدگی مسئله، می تواند سطح بیشتری از جزئیات فراهم شود، به عنوان مثال تجزیه فعالیت‌ها به فعالیت‌های فرعی (سطح ٤ در شکل 1 را مشاهده کنید) یا با مشخصات متنی نیازمندی‌ها از فعالیت‌های واحد (Individual activities) (سطح ٥ در شکل 1 را مشاهده کنید).

بنابراین، نیازمندی‌های دانه‌درشت مشتری (Coarse-grained customer requirements) به طور پی‌درپی به مشخصات ریزدانه (Fine-grained) عملکرد مورد انتظار، پالایش می‌شوند.

توجه داشته باشید که روش فوق به معنی تجزیه‌وتحلیل و تجزیه نیازمندی از بالابه‌پایین است. این دیدگاه تا حدی آرمانی همیشه واقع‌بینانه نیست. در واقع ممکن است نیازمندی‌ها ابتدا در سطح جزئیات راهکار، با اهداف کلی‌تر که بعداً تعیین می‌شوند، بیان شوند؛ بنابراین روند دنیای واقعی ممکن است از بالابه‌پایین، از پایین‌به‌بالا یا ترکیبی از این موارد باشد.

نکته اصلی این است که متدها و تکنیک‌هایی وجود دارند تا از این مباحث در سطوح مختلف انتزاعی‌شان پشتیبانی کنند و با افزایش دانش کلی نیازمندی‌ها، به ایجاد یک سلسله‌مراتب معنی‌دار از نیازمندی‌ها بپردازد.

یک گزینه برای توصیف موردهای کاربرد با پیچیدگی کم می‌تواند استفاده از چند جمله برای شفاف‌کردن گام‌های موردنیاز برای اجرا باشد. برای موردهای کاربرد با پیچیدگی متوسط، ممکن است گزینه‌های دیگری مناسب باشند، زیرا

الگوهای مورد کاربرد می‌توانند شرح مراحل پیچیده‌تر فرایند را که بخشی از مورد کاربرد خاصی هستند، فراهم کنند. این قالب نیمه‌ساختاریافته به درک نیازمندی‌های ثبت شده در یک مورد کاربرد با یک راهنمای مشخص کمک می‌کند. بااین‌وجود، الگوی مورد کاربرد در صورت وجود فعالیت‌های موازی و پیچیدگی زیاد با چندین سناریو، محدودیت‌هایی دارد.

در نهایت، نمودارهای فعالیت، به مهندس نیازمندی‌ها اجازه می‌دهند تا از یک قالب گرافیکی که در آن چندین سناریو قابل نمایش است، استفاده کند. دیاگرام می‌تواند چندین جریان موازی و متوالی از مراحل را نشان دهد. این فرم از توضیحات مورد کاربرد اجازه می‌دهد تا جریان‌های پیچیده مورداستفاده در یک نمودار شرح داده شوند.

موضوعی که همواره منجر به بحث می‌شود، زمانی است که نیازمندی با جزئیات کافی توصیف می‌گردد. تمرکز اصلی در اینجا بر روی همکاری در تیم است. یک نیازمندی زمانی با جزئیات کافی توصیف می‌شود که تیم (به‌ویژه افرادی که نیازمندی را پیاده‌سازی می‌کنند، به‌عنوان‌مثال توسعه‌دهندگان) متوجه شده باشند که چه چیزی از آن‌ها خواسته شده. توصیه می‌شود از قبل معیارهای روشنی برای این موضوع تعریف کنید. تعریف آماده (Definition of Ready) و تعریف انجام شده (Definition of Done) را نیز مشاهده کنید.

اصطلاحات چابک: اپیک‌ها، تم‌ها، فیچرها و داستان‌های کاربر

در سمت راست شکل 1 ، یک رویکرد معادل (An equivalent approach) با استفاده از اصطلاحات چابک مانند اپیک، تِم ها، فیچرها و داستان های کاربر نشان داده شده است.

در اینجا، اهداف سطح بالای کسب‌وکار و عملکرد پیچیده به‌عنوان اپیک یا فیچر ثبت شده و بر اساس تم‌ها دسته‌بندی می‌شوند. چنین موضوعاتی ممکن است به‌قدری پیچیده باشند که به بیشتر از یک اسپرینت برای توسعه توسط تیم اسکرام نیاز داشته باشند [Griff۲۰۱۵].

اینگونه مباحث پیچیده سپس به داستان های کاربر کوچک تر تجزیه می شوند. قبلاً معیار برای داستان کاربر "خوب"، مورد بحث قرار گرفت (۲٫۲). آن ها باید از "تعریف آماده" [Kron۲۰۰۸] نظیر تحقق معیار [Wake۲۰۰۳] INVEST، پیروی کنند. به خصوص "S"، "E" و "T" حائز اهمیت هستند. آن ها باید قابل تخمین (Estimable) و به اندازه کافی کوچک (Small enough) باشند تا در یک تکرار قرار بگیرند و قابل آزمون (Testable) باشند.

مجدداً، اینکه آیا یک پروژه، فرایندی کاملاً از بالابه‌پایین برای پالایش اپیک‌ها (Refining epics) به فیچرها و یا داستان‌های کاربر را دنبال می‌کند، یا اینکه داستان‌های کاربر ابتدا به‌صورت جداگانه با تم‌های مشترک جمع‌آوری می‌شوند و اپیک‌هایی که بعداً از بالا به وجود می‌آیند، به ماهیت پروژه و ذی‌نفعان مرتبط بستگی دارد. درهرصورت چابکی، محصولات کاری را برای بحث و اولویت‌بندی در مورد تصویر کلی (Big picture) و نیازمندی‌های آماده (Development-ready requirements) برای توسعه فراهم می‌کند.

۳٫۱٫۶ مدل‌های گرافیکی و توضیحات متنی

مستقل از دانه‌بندی، همیشه بحث بر سر انتخاب نماد (Notation) در مورد نیازمندی‌های عملکردی وجود دارد. این نمادها را می‌توان به‌صورت متنی نیمه‌ساختاریافته (مانند داستان‌های کاربر منطبق با الگو)، یا به‌عنوان یک‌زبان ساده طبیعی یا رسمی (مانند Gherkin) بیان کرد که به‌روشنی بیان می‌کند راهکار باید چه کاری انجام دهد.

ازآنجایی‌که این موضوع به‌خوبی شناخته شده است که زبان طبیعی گاهی اوقات به‌اندازه لازم دقیق و بدون ابهام نیست، بسیاری از نمادهای گرافیکی برای غلبه بر این ابهام یا نشان‌دادن وابستگی متقابل، برای مثال میان داده‌ها و فرایند، ایجاد شده‌اند. مثال‌ها شامل دیاگرام‌های فعالیت یو.ام.ال (UML Activity diagram)، دیاگرام‌های BPMN، نمودارهای جریان (Flow charts)، نمودارهای حالت (State chart)، دیاگرام‌های توالی (Sequence diagrams) و غیره می‌شوند.

اغلب این نمادهای مبتنی بر گرافیک، بر جنبه‌های مختلفی تأکید دارند. نمودارهای فعالیت یا نمودارهای BPMN برای فرایندهای نسبتاً خطی که مراحل متوالی، گزینه‌ها یا حلقه‌ها را نشان می‌دهند، مناسب هستند. برعکس، نمودارهای حالت هر زمان که رویدادهای ناهمگام (Asynchronous events) بتوانند جریان را تحت‌تأثیر قرار دهند، عالی هستند. نمودارهای توالی برای "بیان مشخصات به همراه مثال" (Specification by example) مناسب هستند، زیرا آن‌ها حتی در صورتی که کامل نباشند، سناریوهای مشخصی از تعاملات را نشان می‌دهند.

در مدل‌سازی فرایندها و داده‌ها، یا داده‌ها و تعاملات، یا فرایندها و رابط‌ها نکات مثبت و منفی وجود دارد. به دلیل اینکه این موارد به هم مرتبط هستند (داده از فرایند پشتیبانی می‌کند)، در نتیجه هرگز نمی‌تواند یک وضعیت وجود داشته باشد که فقط پیشنهاد می‌کند تا از این مدل یا آن مدل استفاده شود. چنین رویکردی کاملاً بر ضد نتیجه خواهد بود. درحالی‌که بسیاری از ذی‌نفعان درک آسان‌تری از نیازمندی‌های متنی دارند، اما این نوع از نیازمندی‌ها به‌راحتی قابل‌تفسیر یا برداشت اشتباه توسط خوانندگان یکسان هستند. مدل‌های گرافیکی رسمیت بیشتری را فراهم می‌کنند؛ بنابراین از تفسیرهای مختلف یا درک اشتباه جلوگیری می‌کنند. انتخاب سبک باید توسط اهداف کلیدی مهندسی نیازمندی‌ها تعیین شود، اطمینان از درک مشترک بین همه ذی‌نفعان ازیک‌طرف و محافظت کافی در برابر ناقص بودن و درک‌های اشتباه از طرف دیگر.

۳٫۱٫۷ تعریف اصطلاحات، واژه‌نامه‌ها و مدل‌های اطلاعاتی

نیازمندی‌های عملکردی، بدون درک دقیق تمام اصطلاحات استفاده شده در جمله‌ها و مدل‌های گرافیکی، ناقص هستند.

بنابراین، مجموعه‌ای از کلیه اصطلاحات مربوط به کسب‌وکار که در محصول کاری نیازمندی‌ها به‌کاررفته است، به‌عنوان یک محصول کاری ضروری در نظر گرفته می‌شوند، اما در هنگام تمرکز بر نیازمندی‌های کسب‌وکار، به‌راحتی نادیده گرفته می‌شوند.

شکل این محصول کاری به‌صورت حداقلی، لیستی از اصطلاحات و اختصارات کسب‌وکاری (توصیف شده به‌صورت متنی) است که معمولاً برای جستجوی آسان‌تر به ترتیب حروف الفبا طبقه‌بندی می‌شوند. برخی اوقات، این مورد را دیکشنری، واژه‌نامه یا لیست تعاریف می‌نامند.

هنگامی که کسب‌وکار بسیار پیچیده است، می‌توان این واژه‌نامه را با گروه‌بندی اصطلاحات ساده در کلاس‌ها (موجودیت‌ها) و نمایش نمای کلی تمامی اصطلاحات در قالب گرافیکی، ساختاردهی کرد. به چنین محصولات کاری مدل‌های داده (Data models)، مدل‌های اطلاعات (Information models)، دیاگرام‌های موجودیت - رابطه (Entity-relationship diagrams) یا نمودار کلاس یو.ام.ال (UML Class Diagram) گفته می شود. علاوه بر تعریف اصطلاحات، این مدل‌ها همچنین شامل ارتباطات مربوطه بین این موجودیت‌ها (Entities)، یعنی روابط ایستا (Static relationship) میان اصطلاحات هستند.

همان‌گونه که در بالا ذکر شد، بک‌لاگ محصول اغلب متشکل از اپیک‌ها، فیچرها و داستان‌های کاربر است که بر عملکرد موردنیاز تأکید می‌کند و تأکید بر تعاریف اصطلاحات کسب‌وکار را کم‌رنگ می‌کند. بااین‌وجود، حتی در رویکردهای چابک، درک صحیحی از اصطلاحات کسب‌وکار برای ایجاد درک مشترک و همسو از اصطلاحات مورداستفاده در محصولات کاری نظیر اپیک‌ها، فیچرها و داستان‌های کاربر لازم است.

۳٫۱٫۸ نیازمندی‌های کیفی و قیدها

علاوه بر نیازمندی‌های عملکردی (مشخص‌کردن عملکردهای سیستم یا محصول موردنیاز برای ارائه)، نیازمندی‌های کیفی و قیدها برای موفقیت سیستم یا محصول درحال‌توسعه از اهمیت اساسی برخوردار است. به طور سنتی، نیازمندی‌ها و قیدهای کیفی در چارچوب اصطلاح "نیازمندی‌های غیرعملکردی" (Non-functional requirements) گنجانده می‌شوند [CNYM۲۰۰۰]. کارشناسان مهندسی نیازمندی ها برای دهه ها، بر اهمیت این نیازمندی های "غیر عملکردی" تاکید کرده اند. حتی با وجود اینکه اصطلاح "نیازهای غیرعملکردی" هنوز به صورت محدود در عمل استفاده می شود، اما به عنوان یک اصطلاح کلی برای نیازمندی ها و محدودیت های کیفی، IREB از دسته بندی های دقیق تر "نیازمندی های کیفی" و "محدودیت ها" مبتنی بر [Glinz۲۰۲۲]، استفاده می کند.

نیازمندی‌های کیفی، مرتبط با ویژگی‌های خاصی است که یک سیستم یا محصول برای ارائه نیاز دارد، به‌عنوان‌مثال، می‌توان از عملکرد (Performance)، قابلیت اطمینان (Reliability)، ایمنی (Safety)، امنیت (Security) یا قابلیت استفاده (Usability) نام برد [ISO۲۵۰۱۰]. با تأکید بر نیازمندی های عملکردی مشتری در قالب داستان های کاربر، این ریسک وجود دارد که نیازمندی های کیفی به صراحت بیان نشوند. نیازمندی های کیفی را نمی توان به عنوان

داستان های کاربری که می توانند در یک تکرار انجام شوند تعریف کرد: آن ها ترجیحاً ویژگی در حال تکامل محصول در حال توسعه را توصیف می کنند، بنابراین باید به طور مداوم برای تمام داستان های کاربر مورد آزمون قرار گیرند. به عنوان مثال، در اینجا چک لیست مهندسی نیازمندی ها از جنبه های کیفی ([CPREFL۲۰۲۲] را مشاهده کنید)، می تواند مفید واقع شود. گزینه دیگر گنجاندن تمام نیازمندی های کیفی در تعریف انجام شده (DoD) است تا از این امر اطمینان حاصل شود که تمام نیازمندی های کیفی برای تبدیل یک مورد بک لاگ به بخشی از اینکریمنت بالقوه قابل استقرار، برآورده شده است.

ایجاد نیازمندی‌های کیفی به‌وسیله بازسازی مجدد (Refactoring) در سیستم‌های موجود، بسیار دشوار است؛ بنابراین درنظرگرفتن چنین جنبه‌هایی در مراحل اولیه بسیار ارزشمندتر است.

قیدها (Constraints) به تعریف محدودیت‌های کلی در فضای راهکار سیستم یا محصول مورد توسعه می‌پردازند. قیدها به اشکال مختلف وجود دارند: قیدهای سازمانی (قیدهای بودجه، برنامه‌های زمانی فشرده، روند توسعه تجویزی و غیره)، قیدهای فنی (نیاز به سیستم خاص DB، استفاده از زبان برنامه‌نویسی خاص، چارچوب‌های انتخاب شده و غیره) یا قیدهای خاص محیط (Environement) که سیستم در آن کار خواهد کرد (استانداردها (Standards)، هنجارها (Norms)، مقررات (Regulations) و غیره).

مشابه نیازمندی‌های کیفی، یادگیری خیلی دیر در مورد قیدهای کلیدی می‌تواند بسیار پر هزینه باشد، زیرا بسیاری از این جنبه‌ها را نمی‌توان به‌صورت تدریجی اضافه کرد. تصمیم‌گیری در زمینه برنامه‌ریزی و طراحی به درک خوب این موضوعات بستگی دارد و در نتیجه مجدداً در اینجا شناسایی قیدهای کلیدی در مراحل اولیه بسیار مهم است.

مشابه نیازمندی‌های عملکردی، ممکت است نیازمندی‌های کیفی و قیدها در سطوح مختلف انتزاع وجود داشته باشند و باید در بک‌لاگ محصول ثبت شوند، برای تیم قابل‌مشاهده باشند و در هر تکرار مورد آزمون قرار گیرند. بنابراین، ممکن است این امر که محدودیت‌ها را به "تعریف انجام شده" (Definition of Done) اضافه کنیم و اعتبارسنجی آن‌ها را در قالب آزمون‌های رگرسیون (Automated regression test) خودکار انجام دهیم، مفید باشد.

۳٫۱٫۹ معیار پذیرش (Acceptance criteria) و معیار تناسب (Fit criteria)

انواع نیازمندی‌ها از ارزش محدودی برخوردارند، اگر تحقق آن‌ها قابل اعتبارسنجی، بررسی یا آزمودن نباشد؛ بنابراین، هر یک از نیازمندی‌ها به مجموعه معیارهایی نیاز دارند که بتوان به‌وسیله آن‌ها، نیازمندی‌ها را مورد آزمون قرار داد تا برآورده‌شدن آن‌ها مورد بررسی واقع شود. علاوه بر این، این معیارها با توصیف دقیق آنچه انتظار می‌رود به درک (و تشویق بازخورد) کمک می‌کنند.

نوع معیارهای استفاده شده با سطح جزئیات نیازمندی مطابقت دارد:

- در سطوح انتزاع بالاتر (چشم‌انداز، اهداف و اپیک‌ها)، معیارهای موفقیت [SAFe] معمولاً به این دلیل تعریف می‌شوند که اندازه‌گیری تنها در صورت فراهم بودن یا عدم وجود قابلیت یا توانایی لازم امکان‌پذیر است.
- به‌منظور کسب مقبولیت، می‌توان در سطوح انتزاع پایین از معیارهای پذیرش برای توصیف چگونگی آزمودن راهکار مرتبط با نیازمندی، استفاده کرد.

در متدهای غیرچابک و چابک این موضوع مورد توافق است که نیازمندی‌ها باید قابل‌تأیید (Verifiable) باشند. متدهای غیرچابک اغلب از اصطلاحاتی مانند "معیارهای کیفیت" (Quality criteria) یا "معیارهای تناسب" (Fit criteria) یا "موردهای آزمون" (Test cases) قدیمی استفاده می‌کنند. در چابک، اصطلاحات "معیارهای پذیرش" (Acceptance criteria) (برای داستان‌های کاربر) یا "معیارهای موفقیت" (Success criteria) (برای اپیک‌ها یا تم‌ها) بیشتر رایج است.

۳٫۱٫۱۰ تعریف آماده (Definition of Ready) و تعریف انجام شده (Definition of Done)

درحالی‌که معیارهای پذیرش و تناسب، به نیازمندی‌های کسب‌وکار تعلق دارند و آنها را تکمیل می‌کنند، محصولات کاری، "تعریف آماده" (DoR) و "تعریف انجام شده" (DoD)، از فرایند رسمی توسعه پشتیبانی می‌کنند و از کیفیت نیازمندی‌ها و اینکریمنت محصول اطمینان حاصل می‌کنند. تعریف تمام شده (DoD) بخشی رسمی از راهنمای اسکرام است [Scrum۲۰۲۰] و به‌عنوان یک تعهد به اینکریمنت تشریح شده است. تعریف تمام شده به‌عنوان یک دروازه کیفیت (Quality gate) برای فرایند توسعه چابک عمل می‌کند، درحالی‌که DoR یک پرکتیس خوب است که باید به ایجاد نیازمندی‌های ارزشمند کمک کند و از اضافه‌بار تیم با نیازمندی‌های نامشخص جلوگیری کند. DoR کمک می‌کند تا نیازمندی‌ها به جزئیات مناسب برسند و اطلاعات کافی را برای مذاکره میان مالک محصول/ مهندس نیازمندی‌ها و توسعه‌دهندگان ارائه می‌دهد.

۳٫۱٫۱۱ نمونه اولیه در مقابل اینکریمنت‌ها (Prototype vs Increments)

روش دیگر کار با نیازمندی‌ها، استفاده از نمونه‌های اولیه است، زیرا بسیاری از ذی‌نفعان که به سیستم یا کالایی نیاز دارند، نمی‌خواهند مستنداتی برای تعریف محصول بنویسند یا بخوانند - آنها موفقیت فوری می‌خواهند. برای آن دسته از افراد، بهترین راه برای درک نیازمندی‌های آن‌ها و دریافت بازخورد، نمایش عملکردها یا قابلیت‌های سیستم در قالب یک سیستم در حال اجرا است.

یکی از راه‌های انجام این کار استفاده از محصولات حداقلی و تدریجی (Minimal and incremental products) است. مهندسی نیازمندی‌ها دو نوع را پیشنهاد می‌کند: اینکریمنت‌های "افقی" محصول (Horizontal product

increments) برای نشان‌دادن انواع مختلف برای اعتبارسنجی ("انجام کارهای بیشتر") و اینکریمنت "عمودی" محصول برای تأیید درست بودن رویکرد توسعه ("انجام کار درست").

با ارائه تعامل مستقیم با سیستم کاری، نمونه‌هایی اولیه می‌توانند برای به‌دست‌آوردن بازخورد بسیار مفید باشند. به‌عنوان‌مثال مشخص‌کردن ویژگی‌های کاربردی، مانند زمان واکنش (Reaction time) دشوار است؛ اما هنگام استفاده از نرم‌افزار قابل‌استفاده (Working software)، شناسایی آن آسان است. بااین‌حال، نمونه‌های اولیه نیز ممکن است مأیوس‌کننده باشند - مأیوس‌کننده برای کاربران، به این دلیل که معتقد هستند که توسعه به اتمام رسیده است و مأیوس‌کننده برای توسعه‌دهندگان، زیرا اغلب این نمونه‌های اولیه دور ریخته می‌شوند تا بعدها با فناوری بهتر جایگزین شوند.

متدهای چابک سعی دارند با ایجاد سریع اینکریمنت باکیفیت خوب، سیستم یا محصول "واقعی" ، از کنار گذاشتن نمونه‌های اولیه جلوگیری کنند. چابک برای تکرارهای کوتاه تلاش می‌کند تا محصولات قابل‌مشاهده‌ای را نشان دهد که به نوبه خود برای کسب اطلاعات بیشتر در مورد نیازمندی‌ها استفاده می‌شوند. باتوجه‌به اینکه هدف این نیست که کد توسعه‌یافته را دور ریخته شود، بازسازی مجدد (Refactoring) یک بِه روش چابک است و به‌عنوان پاسخی به تغییر نیازمندی‌های عملکردی یا کیفی بر اساس بازخورد کاربر در نظر گرفته می‌شود.

اصطلاح "اسپایک" (Spike) در چابک برای اشاره به یک تکرار توسعه (Development iteration) است که باهدف درک صحیح بخشی از پیچیدگی (به‌عنوان‌مثال معماری سیستم) و در نتیجه کاهش ریسک استفاده می‌شود. این اصطلاح اگرچه به طور دقیق تعریف نشده است، اما می‌تواند به اعتبارسنجی یک تسک یا تکرار کامل (Whole iteration) اشاره داشته باشد. نمونه‌سازی اولیه یک تکنیک معتبر است که در آن برخلاف سایر تکرارهای چابک، هدف اصلی آن به‌جای کد قابل‌استفاده، به‌دست‌آوردن دانش است.

۳٫۱٫۱۲ خلاصه محصولات کاری

همان‌طور که در پاراگراف‌های گذشته ذکر شد، برخی از محصولات کاری برای توسعه موفقیت‌آمیز بسیار مهم هستند:

- شروع با چشم‌اندازها و اهداف، یک پرکتیس خوب است.
- شناسایی و دانستن مهم‌ترین ذی‌نفعان، همیشه یک پرکتیس خوب است.
- تعیین صریح دامنه و محدودکردن آن از زمینه، یک پرکتیس خوب است.

حتی اگر مهندسی نیازمندی‌های غیرچابک و مهندسی نیازمندی‌های چابک اصطلاحات مختلفی به کار ببرند، بر ضروری بودن درک عملکرد و همچنین اصطلاحات کسب‌وکار و ضبط نیازمندی‌های عملکردی (از جمله عملکرد و داده‌ها) توافق می‌کنند.

علاوه بر این، متدهای چابک و غیرچابک باید نیازمندی‌های کیفی و قیدها را شناسایی کنند، زیرا این امر ممکن است به‌شدت بر تصمیمات طراحی تأثیر بگذارد. بی‌توجهی به آن‌ها یا بسیار دیر آموختن درباره آن‌ها، ممکن است منجر به کار مجدد و همچنین محصولی شود که انتظارات مشتری را برآورده نکند.

مهندسی نیازمندی‌های خوب تضمین می‌کند که هیچ موضوع مرتبطی فراموش نمی‌شود؛ بنابراین متدهای غیرچابک اغلب بر مستندکردن بسیاری از زمینه‌های بالقوه موردعلاقه مربوط به نیازمندی‌های یک سیستم اصرار دارند.

رویکردهای چابک از محصولات کاری کمتر رسمی استفاده می‌کنند و مستندات ازدست‌رفته را جایگزین حلقه‌های ارتباط مستقیم و بازخورد سریع می‌کنند که از طریق توسعه تدریجی محصول یا نمونه‌های اولیه امکان‌پذیر است.

اصل دوم مانیفست چابک که اغلب نقل می‌شود [AgileMan۲۰۰۱]، دقیقاً همان چیزی را که موردبحث ما قرار گرفت را توضیح می‌دهد:

„... از طریق این کار، ما به این ارزش رسیده‌ایم:... نرم‌افزار کارکننده ارجح بر مستندات جامع... "

ما متقاعد شده‌ایم که بررسی عمدی آنچه باید به‌صورت مکتوب بیان شود، آنچه می‌توان در مورد آن بحث کرد و آنچه که می‌توان با نمونه اولیه یا به‌صورت اینکریمنت نشان داد، برای هر سازمانی بسیار مثمرثمر است. بهترین نتایج زمانی حاصل خواهد شد که مستندات، ارتباطات و نمونه‌سازی/توسعه تدریجی باتوجه‌به محدودیت‌ها و فرهنگ شرکت متعادل شود. فصل ٤ بیشتر، موضوع تعادل بخشیدن به فعالیت های مقدماتی (Upfront activities) (و معماری) با فعالیت های تکراری (Iterative activities) را پوشش خواهد داد.

۳٫۲ تکنیک‌های RE@Agile (L2)

در ۳٫۱ شما در مورد محصولات کاری مهم نیازمندی ها مواردی را یاد گرفته اید. در این واحد آموزشی، با فعالیت های کلیدی که باید در مهندسی نیازمندی ها انجام شوند، آشنا خواهید شد. کتاب راهنما برای سطح مبانی CPRE [CPREFL۲۰۲۲]، این فعالیت ها را به روش زیر ساختاردهی می کند:

- استخراج نیازمندی‌ها
- مستندسازی نیازمندی‌ها
- اعتبارسنجی و مذاکره نیازمندی‌ها
- مدیریت نیازمندی‌ها

در ادامه، زیربخش‌های ارائه شده این فعالیت‌ها را از منظر چابک به بحث می‌کشد.

۳٫۲٫۱ استخراج نیازمندی‌ها

مهندسی نیازمندی‌ها در توسعه چابک بر پایه ارتباطات نزدیک میان تمامی ذی‌نفعان (از جمله توسعه‌دهندگان) برای استخراج نیازمندی‌ها، بنا شده است. ارتباطات غیررسمی و مستقیم بین اعضای تیم می‌تواند به‌خودی‌خود، ترکیب خوبی از مصاحبه و توفان فکری در نظر گرفته شود. تکنیک‌هایی مانند "مشتری در محل" (Onsite customer) متد XP، می‌توانند به همان اندازه در قالب جلسات بین مالک محصول و ذی‌نفعان (به طور معمول توسعه‌دهندگان) نتیجه‌بخش باشد. هدف در تمامی موارد، دستیابی به بینش فراتر از نیازهای واقعی است.

مهندسی نیازمندی‌ها بیش از اینکه به طور معمول در زمینه توسعه چابک بحث کند، به فراهم‌کردن طیف وسیعی از تکنیک‌ها برای کشف و استخراج نیازمندی‌ها می‌پردازد این موارد شامل تکنیک‌های Q/A (نه‌تنها مصاحبه، بلکه پرسش‌نامه)، تکنیک‌های مشاهده، تکنیک‌های مبتنی بر محصول کاری (استفاده مجدد System (Reuse)، archeology و غیره) [CPREFL۲۰۲۲] و تکنیک‌های خلاقیت مانند توفان فکری یا تفکر دیزاین است. این تکنیک‌ها، از ایده ارائه نیازمندی‌ها در قالب داستان کاربر پشتیبانی می‌کنند که مبنایی برای بحث ساختاریافته هستند و نسخه‌ای برای اجرا تلقی نمی‌شوند.

همان‌طور که در بخش ۳٫۱٫۱۱ ذکر شد، نمونه‌های اولیه و اینکریمنت محصول از جمله روش‌هایی برای کسب اطلاعات بیشتر در مورد نیازمندی‌ها هستند. به محض این که اینکریمنت محصول در جلسه بررسی اسپرینت یا جلسه دمو ارائه شود، ممکن است ایده‌های جدیدی ارائه شوند که می‌توانند مستقیماً به بک‌لاگ محصول وارد شده و توسط مالک محصول اولویت‌بندی شوند.

مهندسی نیازمندی‌ها در توسعه چابک می‌تواند با مطالعه تکنیک‌های مختلف استخراج و تصمیم‌گیری حساب شده در مورد انتخاب ترکیبی از تکنیک‌ها، از مزایای مهندسی نیازمندی‌های غیرچابک در پروژه‌ها و توسعه محصولات بهره‌مند شود. درحالی‌که برقراری ارتباط نزدیک میان ذی‌نفعان و بازخورد سریع از طریق اینکریمنت محصول، ایده بسیار خوبی است، اما استخراج چیز دیگری بیش از این است. به‌عنوان‌مثال، تنها بهره‌مندی از ارتباط کلامی در زمانی که صدها یا هزاران ذی‌نفع وجود داشته باشند، کافی نیست. هنگام تلاش برای کشف نیازمندی‌ها یا نیازمندی‌ها نوآورانه، در جایی که ذی‌نفعان حتی از امکان آن آگاه نیستند، ممکن است تکنیک‌های خلاقیت موردنیاز باشد.

هنگام کار با محدودیت زمان‌بندی دقیق (وقت کافی برای بحث‌های فشرده)، ممکن است تکنیک‌هایی مانند کارت‌های برفی (Snow cards) بسیار کارآمد باشند. هنگام کاوش در فن آوری‌های جدید، اسپایک‌ها (جدول زمانی، اینکریمنت‌های ساده برای کشف راه‌حل‌های بالقوه) یک ایده عالی به‌حساب می‌آیند.

۳٫۲٫۲ مستندسازی نیازمندی‌ها

در چابک، نیازمندی‌ها درون بک‌لاگ سازماندهی می‌شوند. ممکن است نیازمندی‌ها در قالب کارت‌های داستان (Story cards) دارای ارزش و اولویت ثبت شوند. ممکن است این موارد در نگاشت‌های داستان (Story maps) سازماندهی شده یا به داستان‌های ساده‌تری تجزیه شوند. اصل مدنظر در مورد کارت‌ها این است که اندازه کارت، موجب محدودشدن آن چه روی آن نوشته شده می‌شود و به تمرکز روی جزئیات اصلی کمک می‌کند.

بااین‌وجود، مستندسازی نیازمندی‌ها همچنان یک فعالیت مهم در جهت تقویت ارتباط بین همه ذی‌نفعان به‌حساب می‌آید. اگر جزئیات به‌صورت کلامی ابلاغ شود، رسمیت مستندات (Formalism for documentation) به حداقل می‌رسد.

تعریف میزان کافی مستندات، به عوامل زیادی مانند اندازه پروژه‌ها، تعداد ذی‌نفعان درگیر، محدودیت‌های قانونی یا اهمیت ایمنی (Safety-criticality) پروژه یا محصولات بستگی دارد. باتوجه‌به چنین عواملی، فرایندهای توسعه چابک تلاش می‌کنند از مصرف بیش از حد مستندات جلوگیری کنند و حداقل مجموعه‌ای از محتوای مفید مستندات را مدنظر قرار دهند.

درحالی‌که کار با یک محصول "در حال استفاده" (Living product) یک روش کارآمد برای رسیدگی به مستندات است، اما این مورد همیشه کافی نیست؛ بنابراین، بیایید نگاهی به انواع دیگر مستندات بیندازیم.

از دیدگاه RE، چهار نوع مستندات را از هم تفکیک می‌کنیم:

۱. **مستندات برای اهداف حقوقی (Legal):** دامنه‌ها یا زمینه‌های خاص پروژه (به‌عنوان‌مثال نرم‌افزار در بخش مراقبت‌های بهداشتی (Healthcare) یا هواپیمایی (Avionics)) برای اخذ تأییدیه قانونی نیاز به مستندسازی اطلاعاتی خاص (به‌عنوان‌مثال نیازمندی‌ها و موردهای آزمون (Test cases) سیستم) برای مخاطبان خاص دارند.

اصل ایجاد مستندات برای اهداف قانونی این است: مستندات قانونی ضروری باید از قوانین یا استانداردهای مربوطه گرفته شده و جزئی جدانشدنی از محصول باشند.

۲. **مستندات برای اهداف نگهداری (Preservation):** برخی اطلاعات خاص در مورد یک سیستم فراتر از تلاش اولیه توسعه، دارای ارزش پایدار هستند. مثال‌ها شامل اهدافی هستند که سیستم برای دستیابی به آن‌ها ساخته شده است، موردهای کاربرد اصلی که از آن پشتیبانی می‌کنند یا تصمیماتی که در طول توسعه آن گرفته شده است که به‌عنوان‌مثال موجب حذف برخی از ویژگی‌های خاص می‌شوند. مستندات برای اهداف نگهداری، می‌توانند به بایگانی مشترک تیم، یک محصول یا یک سازمان تبدیل شوند. این مورد می‌تواند

وابستگی به ظرفیت حافظه افراد تیم را از بین ببرد و بحث در مورد تصمیمات قبلی را کاهش دهد (به‌عنوان‌مثال "چرا ما تصمیم گرفتیم این مورد را اجرا نکنیم؟").

اصل این است: تیم تصمیم می‌گیرد که چه مواردی را برای اهداف نگهداری، مستند کند.

۳. **مستندات برای اهداف ارتباطی (Communication):** ارتباط مؤثر و کارآمد به دلیل تعامل و چرخه کوتاه بازخورد، ابزاری مهم در متدهای چابک است. در عمل، چندین موقعیت وجود دارد که ممکن است مانع از ارتباط مستقیم کلامی شود: تیم‌های توزیع شده (Distributed teams)، موانع زبان (Language barriers) یا محدودیت‌های زمانی افراد درگیر (Time restrictions). علاوه‌براین، اطلاعات گاهی اوقات به‌قدری پیچیده هستند که ممکن است ارتباط مستقیم، ناکارآمد یا گمراه‌کننده باشد. به‌عنوان‌مثال، نمونه اولیه کاغذی یا نموداری از یک الگوریتم پیچیده می‌تواند بعدها مورد بازخوانی قرار گیرند. گاهی اوقات ذی‌نفعان به‌سادگی، ارتباط کتبی را به خواندن کد منبع یا مرور نرم‌افزار ترجیح می‌دهند. در این موارد اسناد و مدارک، روند ارتباط میان همه طرف‌های درگیر را تسهیل می‌کنند و نتایج را حفظ می‌کند.

اصل ایجاد مستندات برای اهداف ارتباطی این است: درصورتی‌که ذی‌نفعان یا توسعه‌دهندگان ارزشی در وجود مستندسازی ببینند، یک مستند می‌تواند به‌عنوان وسیله ارتباطی اضافی ایجاد شود. هنگامی‌که ارتباط موفقیت‌آمیز باشد، این مستند بایگانی می‌شود. هنگامی که ارتباط موفقیت‌آمیز است، این مستند بایگانی می‌شود.

٤. **مستندات برای اهداف تفکر (Thinking):** جنبه‌ای که اغلب از نوشتن یک مستند فراموش می‌شود این است که نوشتن همیشه وسیله‌ای برای بهبود و پشتیبانی از روندهای فکری نویسنده است. حتی اگر مستند، بعدها در طول فرایند کنار گذاشته شود، مزیت بهبود و حمایت از تفکر، پایدار و باقی می‌ماند. به‌عنوان‌مثال، نوشتن یک مورد کاربرد (Use case) نویسنده را مجبور می‌کند تا در مورد تعاملات مشخص بین سیستم و بازیگران از جمله، به‌عنوان‌مثال، موارد استثنا (Expectations) و سناریوهای جایگزین (Alternative scenarios)، بیندیشد؛ بنابراین می‌توان نوشتن یک مورد کاربرد را ابزاری برای سنجش دانش و درک شما از یک سیستم دانست.

اصل ایجاد مستندات برای اهداف تفکر این است: متفکر (Thinker) در مورد مستندی که بهترین پشتیبانی را از تفکر او کند، تصمیم می‌گیرد. متفکر نیازی به توجیه انتخاب فرم مستندات برای تفکر ندارد. با اتمام روند تفکر، ممکن است مستند کنار گذاشته شود.

در صورتی که این چهار نوع مستندات شناسایی و در زمینه مناسب به کار برده شوند، متدهای چابک می‌توانند از مزیت قابل‌توجهی بهره‌مند شوند. به طور خلاصه می‌توان گفت که مستندسازی نیازمندی‌ها به‌خودی‌خود هدف نیست، بلکه باید ارتباط میان ذی‌نفعان را، به‌ویژه بین درخواست‌کننده (Requester) (که اغلب مالک محصول به‌عنوان جایگزین او انتخاب می‌شود) و توسعه‌دهندگان را تسهیل کند.

۳٫۲٫۳ اعتبارسنجی و مذاکره نیازمندی‌ها

درحالی‌که مهندسی نیازمندی‌ها بر اعتبارسنجی نیازمندی‌ها از طریق متدهایی نظیر بررسی (Review)، رویدادهای بررسی فرآورده (Walkthrough)، بازرسی‌ها (Inspections) یا خواندن مبتنی بر چشم‌انداز (Perspective based reading) تأکید دارد، متدهای چابک تلاش می‌کنند تا به تأیید نیازمندی‌ها را از طریق بازخورد زودهنگام و مکرر درباره اینکریمنت‌های ارزشمند محصول بپردازند یک پرکتیس خوب برای پشتیبانی از این مورد، آزمون رگرسیون خودکار (Automated regression testing) است که اعتبارسنجی مداوم توسعه و نیازمندی‌های مربوطه را فراهم می‌کند. هدف از تأیید نیازمندی‌ها شامل شناسایی نیازمندی‌های گمشده، مبهم یا نادرست و همچنین نیازمندی‌های بحث‌برانگیز یا متناقضی است که در آن می‌توان از تکنیک‌های مذاکره و حل تعارض استفاده کرد.

ازآنجاکه توسعه تکراری (Incremental development)، یک استراتژی اصلی در متدهای چابک است، نیاز به اعتبار رسمی مستندات کاهش می‌یابد. این مورد، با مذاکره مداوم میان تمامی ذی‌نفعان در مورد نیازمندی‌ها جایگزین می‌شود تا تعارضات در اوایل کشف و حل شود.

اعتبارسنجی رسمی (Formal validation) نیز با نشان‌دادن نتایج سریع به‌صورت اینکریمنت‌های یکپارچه محصول، کاهش می‌یابد. اگر این اینکریمنت پاسخگوی تمام شرایط ذی‌نفعان نباشد، دلتا (Delta) به‌عنوان نیازمندی جدید به بک‌لاگ محصول وارد می‌شود و با سایر آیتم‌های بک‌لاگ ارزیابی و اولویت‌بندی می‌شود.

بااین‌وجود، بررسی‌های مربوط به بک‌لاگ محصول، بحث در مورد ارزش کسب‌وکار، بحث در مورد ریسک‌ها و مذاکرات فوری در موردنیازها، همگی از تکنیک‌های ارزشمند در RE@Agile هستند. همه این تکنیک‌ها ممکن است در جلسات پالایش (Refinement meetings) به کار روند که در آن مالک محصول و توسعه‌دهندگان (و در صورت دردسترس‌بودن، ذی‌نفع) با هم کار می‌کنند تا سطح جزئیات نیازمندی را با استفاده از اصول پالایش مداوم (Continuous refinement) پیدا کنند.

در اولین نسخه‌های راهنمای اسکرام، جلسه پالایش بک‌لاگ محصول تنها به‌صورت غیرمستقیم ذکر شده بود. در نسخه جدید [Scrum۲۰۲۰]، پالایش (Refinement) به‌صراحت به‌عنوان فعالیتی برای استخراج، مستندسازی و اعتبارسنجی نیازمندی‌ها ذکر شده است. این فرایند پالایش که جلسه پالایش بخشی ضروری از آن است، مشکلات احتمالی را زود آشکار می‌کند و بعدها، زمان موردنیاز برای جلسه برنامه‌ریزی اسپرینت را کاهش می‌دهد.

۳٫۲٫٤ مدیریت نیازمندی‌ها

در RE سنتی، مدیریت نیازمندی‌ها مربوط به کلیه فعالیت‌ها برای رسیدگی به نیازمندی‌ها در طول زمان است. این مورد شامل مدیریت نسخه (Version management)، مدیریت تغییر (Change management)، مدیریت پیکربندی (Configuration management)، قابلیت ردیابی (Traceability) و همچنین افزودن ویژگی‌هایی مانند وضعیت (Status)، تخمین‌ها (Estimation)، اولویت‌ها (Priorities)، پیوند به نیازمندی‌های مورد تعارض و افراد مشارکت‌کننده در ضبط، بررسی، امضا، پیاده‌سازی یا آزمون نیازمندی می‌شود.

همان‌طور که قبلاً بحث شد (۳٫۱٫۱ را مشاهده کنید)، محصولات کاری اصلی جهت نگهداری نیازمندی ها، در بک لاگ جمع آوری می شوند. برخلاف مدیریت نیازمندی های سنتی، بک لاگ ها به گونه ای طراحی شده اند که فقط آخرین و بهترین نسخه از تمام نیازمندی هایی را که هنوز اجرا نشده اند، نگه دارند. آیتم های بک لاگ معمولاً به محض تحویل محصول حاوی این نیازمندی، آرشیو یا حذف می شوند.

فعالیت‌هایی که در بک‌لاگ‌ها در قالب مدیریت نیازمندی‌ها انجام می‌شوند، شامل موارد زیر هستند:

۱. اولویت‌بندی نیازمندی‌ها: تعیین ارزش (تجاری) نیازمندی‌ها برای تصمیم‌گیری در مورد زمان اجرای آن‌ها. هرچه ارزش کسب‌وکار بالاتر باشد، اولویت نیازمندی بالاتر است، زیرا پروژه‌های چابک سعی می‌کنند ابتدا عناصری را با بالاترین ارزش تجاری ارائه دهند. عوامل مؤثری که ارزش کسب‌وکار و همچنین اولویت‌بندی حاصل از آن را تعیین می‌کنند در سطح Practitioner و Specialist ماژول RE@Agile پوشش داده شده است [CPREALAGILE۲۰۲۲].
۲. تخمین نیازمندی‌ها: تعیین کنید که چه مقدار کار برای تحقق آن‌ها انجام می‌شود. تخمین‌های بیش از حد بزرگ پیام واضحی به مالک محصول هستند که باید کار بیشتری انجام شود تا بتواند آن‌ها را به تعریف آماده (DoR) برساند (۲ را مشاهده کنید). بنابراین او باید آیتم مربوطه را به واحدهای کوچکتر تقسیم کند تا تخمین ممکن شود.

این بدان معنا نیست که سایر فعالیت‌های مربوط به جنبه‌های تاریخی مدیریت نیازمندی‌ها نمی‌توانند در چابک انجام شوند. این فعالیت‌ها معمولاً در خارج از آیتم‌های بک‌لاگ ثبت می‌شوند.

در تصمیم‌گیری در مورد مناسب‌بودن فعالیت‌های مدیریت نیازمندی‌ها در یک شرایط خاص، مهندس نیازمندی‌ها باید به دنبال ایجاد تعادل میان به‌حداقل‌رساندن هزینه‌های اضافی، امکان تحویل به‌موقع راهکارهای قابل‌استفاده و نیازمندی‌های بلندمدت سازمان مانند انطباق قانونی (Legal compliance)، مستندات عملیاتی (Operational documentation) یا تحویل آن به اعضای جدید توسعه‌دهندگان باشد.

نتیجه‌گیری

فعالیت‌های نیازمندی‌ها نظیر استخراج (Elicitation)، مستندسازی (Documentation)، اعتبارسنجی (Validation) و مذاکره (Negotiation) و همچنین مدیریت (Management) نیازمندی‌ها به‌طورکلی باید در توسعه چابک انجام شوند. تکنیک‌های ترجیحی برای استخراج و مستندسازی ممکن است در توسعه چابک و غیرچابک متفاوت باشند، اما یادگیری از یکدیگر بهترین راه‌حل را نشان می‌دهد، زیرا این امر نه‌تنها موجب ترکیب قدرت‌ها می‌شود؛ بلکه باعث کاهش هدررفت (Waste) یا سربار (Overhead) نیز خواهد شد. این روش کار نشان‌دهنده پنج ارزش اسکرام (تعهد (Commitment)، تمرکز (Focus)، صراحت (Openness)، احترام (Respect) و شجاعت (Courage)) و نماینده آن‌ها در سه ستون اسکرام (شفافیت (Transparency)، بازرسی (Inspection) و سازگاری (Adaptation)) است [Scrum۲۰۲۰].

به‌خصوص صراحت و احترام، هنگام واردکردن استاد کاری مهندسی نیازمندی‌ها (RE craftsmanship) به دنیای چابک و آوردن ایده‌ها و اصول چابک به استاد کاری مهندسی نیازمندی‌ها، بسیار مفید هستند.

در این واحد آموزشی، شما آموخته‌اید که حتی در RE@Agile، به جز داستان‌های کاربر، محصولات کاری دیگری در بک‌لاگ محصول وجود دارند و مهم‌ترین فعالیت‌های مهندسی نیازمندی‌ها را نباید فراموش کرد - اما ممکن است با تأکید و روش‌های مبتنی بر اصول چابکی ذکر شده در ۱، انجام شوند.

٤ جنبه‌های سازمانی RE@Agile (L2)

مدت‌زمان: ۱ ساعت و نیم

اهداف آموزشی

هدف آموزشی ٤/١/١	درک تعامل بین ساختار سازمانی و RE@Agile
هدف آموزشی ٤/٢/١	دانستن تعامل با ذینفعان در فرایندهای مهندسی نیازمندی‌های چابک
هدف آموزشی ٤/٢/٢	دانستن اینکه چگونه ارتباط و همکاری می تواند به بهبود نتایج کمک کند
هدف آموزشی ٤/٢/٣	دانستن نقش مدیریت در چابک
هدف آموزشی ٤/٣/١	دانستن انگیزه برای مقیاس‌پذیری (Scaling)
هدف آموزشی ٤/٣/٢	دانستن ابعاد سازماندهی تیم‌ها
هدف آموزشی ٤/٣/٣	دانستن رویکردهای سازماندهی ارتباطات میان تیم‌ها
هدف آموزشی ٤/٣/٤	دانستن چارچوب‌های نمونه برای مقیاس‌پذیری
هدف آموزشی ٤/٣/٥	دانستن تأثیرات اصلی مقیاس‌پذیری در RE
هدف آموزشی ٤/٤/١	دانستن معیارهایی برای تصمیم گیری در سطح مقدماتی (Upfront) در مقابل مهندسی مداوم نیازمندی‌ها
هدف آموزشی ٤/٤/٢	دانستن مقدار مناسب جزئیات برای آیتم های بک‌لاگ
هدف آموزشی ٤/٤/٣	دانستن ارزش اعتبارسنجی در RE@AGILE
هدف آموزشی ٤/٤/٤	دانستن چرخه به روزرسانی مناسب برای بک‌لاگ محصول
هدف آموزشی ٤/٤/٥	دانستن چگونگی یافتن زمان مناسب چرخه توسعه

٤/١ تأثیر سازمان بر RE@Agile (L2)

ریشه چابک در تولید و کنترل فرآیند تجربی (Empirical process-control) است ([TaNo۱۹۸٦] را مشاهده کنید). اصول چابک به آسانی قابل درک است و شیوه ها و چارچوب های چابک مانند اسکرام در یک محیط جدید (Greenfield) مانند استارتاپ ها یا شرکت های کوچک به راحتی قابل استفاده هستند. پیاده سازی آن ها در سازمان های بزرگتر که مانند سلول (Organism) زنده ای از خود دفاع می کنند، دشوار است. اما از سوی دیگر، وقتی سازمان ها مزایای استفاده از چنین اصول و روش هایی را کشف می کنند، سعی می کنند آن ها را با DNA خود مانند آن چه در قانون کانوی (Conway's law) شرح داده شده، ترکیب کنند [Conw۱۹٦۸] . انجام این کار باعث افزایش علاقه به موضوعاتی مانند مدیریت چابک ([ACP/PMI] را مشاهده کنید) و سازمان های چابک ([Appel۲۰۱۱]، [Denn۲۰۱٥] را مشاهده کنید) شده که منجر به بحث در مورد جنبه هایی از چابک می شود که اغلب فراتر از توسعه (نرم افزار) است.

ایده‌های قراردادن مشتری در مرکز توجه (Customer at the center)، تیم‌های خودسازمانده (Self-organizing teams)، فعال‌کردن (Enabling) و توانمندسازی (Empowering) افراد و پیشرفت مداوم (Continuous

improvement)، در دنیای وسیع کسب‌وکار طنین‌انداز است. چارچوب Scrum at Scale یک گسترش حداقلی از چارچوب اسکرام اصلی است که ساختار ماژولار را در مرکز چارچوب اسکرام حفظ کرده و اجازه می‌دهد تا یک پیاده‌سازی اسکرام متناسب با نیازهای منحصربه‌فرد شرکت شما مقیاس‌پذیر شود.

در دنیای توسعه نرم‌افزار، بسیاری از پیاده‌سازی‌های فرایندهای توسعه چابک در مفهومی‌سازی (Conception) شکست می‌خورند، زیرا سایر اعضای سازمان قادر به تغییر در جهت پشتیبانی از تیم‌های چابک نبوده‌اند.

چرا چنین تغییر سازمانی لازم است؟

بیایید دو نمونه از اهمیت چرایی تغییر سازمان در راستای پشتیبانی از توسعه چابک را بررسی کنیم:

۱. سازمان (بخشی از سازمان) تقاضادهنده باید بتواند نیازمندی‌های به‌اندازه کافی خوب را ارائه دهند (معنی خوب به‌اندازه کافی دقیق (Detailed enough) است؛ اما خیلی دقیق نیست - باتوجه‌به تعریف آماده (Defintion of Ready)) تا روند توسعه را با سرعت ثابت ادامه دهد. این مورد، در ترکیب با نیازمندی‌های در حال تغییر مداوم، نیاز به جریان تقاضای مداوم (Continuous demand flow) دارد.
۲. بخش منابع انسانی باید درک کند که چه افرادی را برای پشتیبانی صحیح از تیم‌های چابک استخدام کند. می‌توان نمونه‌های بدی از پیشنهادهای شغلی را برای اسکرام مستر یافت که در آن بر داشتن مهارت برنامه‌نویسی و گواهینامه تأکید شده است که این نشان‌دهنده آن است که اصول سه نقش اسکرام به‌درستی درک نشده‌اند.

در ٤/٢ ما جنبه های سازمانی را هنگام تعبیه یک واحد سازمانی چابک، مانند یک تیم اسکرام، در یک محیط غیر چابک مورد بحث قرار می دهیم. در بسیاری از موارد، اگرچه معرفی چابکی از توسعه محصول آغاز می شود، اما ممکن است تمام سازمان لزوما از اصول چابک پیروی نکنند.

تأثیر در سازمان در مواردی که بیش از یک تیم چابک برای حل یک مسئله پیچیده موردنیاز است، در ٤/٣ مورد بحث قرار گرفته است. تمرکز اصلی دوباره بر سازمان فناوری اطلاعات و رابط های آن با کسب و کار است. تبدیل کسب و کار به یک سازمان کاملا چابک در اینجا به صراحت مورد توجه قرار نگرفته است، زیرا از حوصله بحث در مورد RE خارج است.

٤/٤ بر جنبه های سازمانی در زمان بندی (Timing)، تجزیه و تحلیل (Analyzing)، و به ویژه این سوال که فعالیت های RE در چه زمانی باید انجام شوند، تمرکز دارد.

٤٫٢ توسعه چابک در محیط غیرچابک (L1)

٤٫٢٫١ تعامل با ذی‌نفعان خارج از سازمان فناوری اطلاعات

نقش سازمان توسعه (Development organization) در سازمان (Enterprise)، ارائه راهکارها و خدمات به مشتریان سازمانی (چه در داخل و چه در خارج از سازمان) است.

چابک، مشتریان را در مرکز توسعه محصول قرار می‌دهد. این بدان معنا است که مشتری در کل چرخه تولید محصول درگیر است، جایی که بازخورد تحویل تدریجی (Incremental deliveries) به طور فعالانه درخواست می‌شود و نیازمندی‌های جدید مطابق با نیازمندی‌های کسب‌وکار را برآورده می‌کند.

با مشارکت مداوم مشتری، RE نیز به یک‌روند مستمر تبدیل می‌شود (٢٫٤ را مشاهده کنید). فرد مسئول، مانند مالک محصول، باید مشتریان خود را درگیر ارتباطات باز و مستقیم کند، به نیازهای جدید و تغییر در انتظارات گوش دهد و تمامی این موارد را در بک لاگ ضبط کند.

در عمل، این ارتباط ممکن است اشکال مختلفی داشته باشد. اگر مشتری در واقع برای کار روزانه با تیم در دسترس باشد، در واقع، اگر مشتری برای کار روزانه با تیم در دسترس باشد، ارتباطات می‌تواند ماهیتی مستقیم و غیررسمی داشته باشد، جایی که فقط نتایج یا تصمیمات ثبت می‌شوند

شناخت عوامل خارج از کنترل توسعه‌دهندگان (به‌عنوان‌مثال، تفکیک جغرافیایی یا صرفاً کمبود دسترسی) از اهمیت زیادی برخوردار است، این بدان معنی است که تعامل مستقیم همیشه عملی نیست. در اینجا، باید یک نوع ارتباط مؤثرتر، یا با دعوت از نمایندگان مشتری به جلسات برنامه‌ریزی و بررسی منظم، یا با انجام جلسات فشرده با زمان ثابت (Time-boxed)، (برای مثال تفکر دیزاین (Design Thinking) یا دیزاین اسپرینت (Design Sprint)، ١٫٥ را مشاهده کنید) در مرحله اولیه با ورودی ضبط شده در بک‌لاگ محصول، برنامه‌ریزی شود.

٤٫٢٫٢ سازمان مبتنی بر محصول در مقابل سازمان مبتنی بر پروژه

چابک در طول زمان، روابط مستقیم، باز و غیر سلسله‌مراتبی درون‌سازمانی و انعطاف‌پذیری را دقیقاً در تکامل محصولات، ارتقا می‌بخشد. سازمان‌های بزرگ‌تر، به طور سنتی بر اساس ساختارهای مدیریتی بالابه‌پایین (Top-down management)، برای برنامه‌ریزی (Planning) و قابلیت پیش‌بینی (Predictability)، اهمیت زیادی قائل هستند. به‌عنوان‌مثال، برنامه‌ریزی پروژه و منابع، واقعیت‌هایی هستند که سازمان‌ها نمی‌توانند به‌راحتی از آن‌ها منصرف شوند.

به طور سنتی، توسعه نرم‌افزار اغلب مبتنی بر پروژه بوده است، به این معنی که به‌عنوان مجموعه‌ای از تعهدات موقت برای تولید محصولات، خدمات یا نتایج منحصربه‌فرد انجام می‌شود ([PMI] را مشاهده کنید). گروه‌هایی از

پروژه‌های مرتبط با اهداف مشترک غالباً برنامه (Program) نامیده می‌شوند، درحالی‌که برنامه‌ریزی و کنترل کل پروژه‌ها و برنامه‌ها در داخل سازمان به‌عنوان مدیریت پورتفولیو (Portfolio Management) نامیده می‌شود.

رویکرد چابک که به ریشه‌های خود در توسعه محصول وابسته است، بیشتر محصول محور است. بک‌لاگی از پیشرفت‌های محصول (Product improvements) حفظ می‌شود و این موارد، در یک فرایند تکرارشونده بهبود مستمر اجرا می‌شوند چابک به‌تنهایی تاریخ پایان را به‌این‌ترتیب تعریف نمی‌کند. اصولاً تا زمانی که پیشرفت‌هایی باید انجام شود و یا منافع قابل‌تحقق باشند، در صورتی که منافع بیشتر از تلاش/هزینه‌ها (Effort/Costs) باشد، باید کار ادامه یابد.

گرچه این رویکردها به طور متقابل انحصار ندارند (محدوده یک پروژه ممکن است تحویل یک محصول خاص باشد و پروژه‌های اضافی نیز برای بهبودهای بعدی ایجاد شده‌اند) اما اختلاف دیدگاه و اصطلاحات ممکن است منشأ تنش و سوءتفاهم میان توسعه نرم‌افزار چابک و سازمان‌های غیرچابک شوند.

یک روش برای حل چنین تنش‌هایی این است که درحالی‌که توسعه نرم‌افزار به‌صورت کاملاً چابک انجام می‌شود، عملکرد مدیریت پورتفولیو و مدیریت برنامه، سطح بالاتری از برنامه‌ریزی و کنترل را فراهم کنند که از رویکردهای هر دو جهان استفاده می‌کند (٤/٣ را مشاهده کنید). نکته اساسی در ایجاد چنین رویکردی، توانایی برطرف کردن شکاف مفهومی میان تحقق برنامه ریزی شده اهداف و ویژگی های تجاری در پورتفولیو و برنامه ها و تحویل تکراری و انعطاف پذیر ویژگی های نرم افزاری واحد است.

مهندسی نیازمندی‌ها، مفاهیم و روش‌های لازم را برای تمایز میان نیازمندی‌های موجود در این سطوح مختلف انتزاع، با نیازمندی‌های سطح کسب‌وکار در پورتفولیو و برنامه‌ها و با نیازمندی‌های نرم‌افزاری مشتق شده و با جزئیات مناسب برای پیشبرد توسعه، فراهم می‌کند. رویکرد نیازمندی‌ها از رویکرد با جزئیات که به‌عنوان دستوراتی دقیق برای توسعه استفاده می‌شود، به نیازمندی‌هایی تغییر می‌یابد که به‌عنوان یک مبنای مشترک برای بحث و هماهنگی به‌موقع و به همان اندازه که برای سطح فعلی برنامه‌ریزی لازم است، مورداستفاده قرار می‌گیرد.

٤/٢/٣ نقش مدیریت در زمینه چابک

دیسیپلین‌ها و تیم‌ها (Disciplines & Teams): کارمندان بخش‌های فناوری اطلاعات به طور سنتی توسط دیسیپلین سازماندهی می‌شدند: توسعه‌دهندگان، آزمون‌کنندگان، مهندسان نیازمندی‌ها، تحلیلگران کسب‌وکار، مدیران پروژه و غیره. تیم‌های پروژه متشکل از این مهارت‌های مختلف، برای مدت‌زمان مشخصی از انجام کار، گرد هم می‌آمدند. چابک و به‌ویژه اسکرام، ایده تیم‌های چند عملکردی (Cross-functional) بیشتر را ترویج می‌دهند. جدا از نقش‌های تخصصی مالک محصول و اسکرام مستر، تمامی اعضای تیم باید توانایی بازی در ظرفیت‌های مختلف را داشته باشند، به‌طوری‌که هر فرد در صورت نیاز، از فعالیت‌های مهندسی نیازمندی‌ها یا فعالیت‌های آزمون پشتیبانی کند همچنین به همین دلیل، این اعضای تیم معمولاً در اسکرام "توسعه‌دهندگان" نامیده می‌شوند. هدف این تیم این است که

بتواند نیازمندی‌های مشتری را به طور کامل و بدون درنظرگرفتن عناصر فنی یا سازمانی ارائه دهد. این را می‌توان با داشتن صلاحیت‌های RE در تیم توسعه (راه‌حل ترجیحی) یا خارج از توسعه‌دهندگان (که در واقع اغلب به‌عنوان پشتیبانی از مالک محصول استفاده می‌شود) که به طور رسمی بخشی از چارچوب اسکرام نیست، این هدف را به دست آورد.

مدیران IT: این وظیفه مدیران IT است (در [SAFe]، "توسعه دهندگان" (People developers) نامیده می شوند) که تعادل مناسبی را در سازمان خود بین مهارت های تخصصی و عمومی پیدا کنند و به تیم ها کمک کنند تا این مهارت ها را در تعداد مناسب تیم ها سازماندهی کنند. با احترام به قانون کانوی [Conw۱۹٦۸] (Conway's Law)، ساختار تیم آینه ای از ساختار محصول (اجزای سازنده) یا ساختار سیستم در IT خواهد بود. این موضوع حتی یک مورد مهم تر در مقیاس پذیری (Scaling) تعدادی مناسب از تیم ها است. اگر شرکتی تیم های خود را توسط اجزا یا سیستم ها سازماندهی کند، مقیاس پذیری با افزایش تعداد تیم ها کمکی نخواهد کرد، زیرا حتی باعث وابستگی بیشتر می شود. ممکن است مقیاس پذیری فقط با مقیاس پذیر کردن تعداد اعضای تیم امکان پذیر باشد، اما فقط تا یک مرحله مشخص، زیرا تلاش برای برقراری ارتباط نیز به طرز چشمگیری افزایش می یابد

تیم‌های چند عملکردی که همه قابلیت‌ها برای انتقال کل مراحل از فرانت‌اند (Frontend) به بک‌اند (Backend) را به ارث می‌برند، می‌توانند به‌راحتی مقیاس‌پذیر شوند - اما ایجاد آن‌ها آسان نیست و کاری وقت‌گیر است.

مالک محصول در مقابل مدیر پروژه: همان‌گونه که در قبل ذکر شد (را مشاهده کنید)، مالکان محصولات با بر عهده گرفتن مسئولیت اولویت‌های کاری در تیم‌های توسعه محصول، مسئولیت مهندسین نیازمندی‌ها را گسترش می‌دهند. مالکان محصول باید فعال بوده (دانش) و توان تصمیم‌گیری در زمینه کسب‌وکار را داشته باشند. برخی از تصمیماتی که قبلاً توسط مدیران پروژه گرفته شده بود، دیگر در کار با رویکردهای چابک موردنیاز نیست، زیرا توسعه‌دهندگان در محدوده سازمان خود، خودسازمانده می‌شوند و تنها نیاز به کاری جهت انجام‌دادن دارند (نیازمندی‌هایی در سطحی از جزئیات که امکان توسعه در یک تکرار را فراهم می‌کنند) تا کار خود را (تجزیه تسک و تعیین تکلیف در تیم) سازماندهی کنند.

آنچه از مدیران پروژه چابک IT مورد انتظار است، تعیین روشن چشم‌انداز توسعه چابک و ارتباط واضح پیش‌شرط‌های فرهنگی برای موفقیت این رویکرد است.

رسیدن به تعادل، درست در حین برآورده‌کردن انتظارات از کسب‌وکار، موضوع ساده‌ای نیست. برخی از معیارهای موفقیت در زیر در ٤/٤ مورد بحث واقع شده است.

مرتبط بودن مهندسی نیازمندی‌ها (RE): الگوهای قبلی به‌عنوان‌مثال برای اسکرام نیز اعمال می‌شود. افرادی که بر اساس اصول RE کار می‌کنند، می‌توانند بخشی از تیم باشند و بنابراین به طور جداگانه نامی از آنها برده نمی‌شود. از سوی دیگر، می‌توانند خودشان تیمی را تشکیل دهند که از یک یا چند مالک محصول، در قالب یک تیم پشتیبانی

می‌کند. هر دو روش مزایای خود را دارند و می‌توانند بدون نقض اصول چابک به‌صورت ترکیبی مورداستفاده قرار گیرند. RE به ستون فقرات توسعه موفق چابک تبدیل خواهد شد.

٤/٣ رسیدگی به مشکلات پیچیده با مقیاس‌پذیری (L1)

٤/٣/١ انگیزه مقیاس‌پذیری

ارزش‌های چابک به طور معمول نمایانگر ارتباط مستقیم، روزانه، غیر سلسله‌مراتبی توسط تیم‌های کوچک و نزدیک به هم مانند تیم اسکرام، با تعداد توصیه شده کمتر از ۱۰ نفر (توسعه دهندگان + مالک محصول + اسکرام مستر) هستند. اعضای تیم در حالت ایده‌آل باید از نظر از نظر فیزیکی در یک مکان بوده و از نظر مهارت‌های تجاری و فنی، چند عملکردی باشند. در سازمان های بزرگتر، به دلایل زیادی این دید ایده آل گرایانه، در مورد توسعه چابک امکان پذیر نیست:

- مشکلات پیچیده ممکن است شامل ذی‌نفعان و دانش بخش‌های مختلف کسب‌وکار باشد که به‌راحتی در یک تیم واحد جای نمی‌گیرد.
- مشکلات پیچیده ممکن است شامل طیف وسیعی از متخصصان فنی و دانشی باشد که به‌راحتی در یک تیم واحد جای نمی‌گیرد.
- دامنه موردنیاز برای یک تاریخ عرضه (Rollout date) مشخص، فراتر از سرعت قابل‌دستیابی یک تیم واحد است.
- کارمندان در مشاغل جهانی ممکن است از نظر جغرافیایی توزیع شوند.

اصطلاح چابک مقیاس‌پذیر (Scaled Agile) برای توصیف شرایطی استفاده می‌شود که چندین تیم لازم است با هم روی یک محصول / راه‌حل مشترک اهداف مشترک کار کنند. رویکردهای چابک مقیاس‌پذیر نیاز به تصمیم‌گیری در مورد چگونگی سازماندهی تیم‌ها و چگونگی هماهنگی ارتباط بین تیم‌ها دارند. هدف، در حین حفظ مزایای چابکی، دستیابی به یک رویکرد مؤثر برای رسیدگی به مشکلات پیچیده است.

٤/٣/٢ رویکردهایی برای سازماندهی تیم‌ها

سؤالی که در اینجا باید پاسخ داده شود این است که سازمان چگونه می‌تواند باتوجه‌به ارتباط و همسویی در عین مؤثربودن (حداکثر اندازه)، خود را در قالب تیم‌هایی با اندازه مناسب قرار دهد و به آن‌ها اجازه چند عملکردی بودن دهد. سازماندهی در امتداد خطوط عملکردی (به‌عنوان‌مثال یک تیم متخصص در یک حوزه کسب‌وکاری مشخص، یا حتی در مورد ویژگی در یک حوزه کسب‌وکار) این مزیت را دارد که دانش کسب‌وکار را در یک تیم واحد متمرکز می‌کند. به‌عنوان‌مثال، این مورد ممکن است به تسهیل در استخراج نیازمندی‌ها از طریق کاهش تعداد ذی‌نفعانی که به‌صورت مستقیم با تیم ارتباط دارند، منجر شود.

یک نقطه‌ضعف این رویکرد این است که ارائه قابلیت‌های انتها به انتها (End-to-End) در یک حوزه کسب‌وکار، احتمالاً شامل چندین تخصص فنی مختلف مانند طراحی رابط کاربر، موتورهای پردازش، پایگاه‌داده و پلتفرم‌های اصلی مانند ERP یا پردازنده‌های مرکزی (Mainframes) اصلی است که به‌راحتی می‌توانند بیش از ظرفیت تیم، فشار وارد کنند.

یک روش جایگزین برای سازماندهی توسعه، خطوط فنی (Technical lines) است و تیم‌های متخصص در مؤلفه‌های فنی (Technical components) یا پلتفرم‌ها فعالیت می‌کنند. مزیت تیم‌های مؤلفه یا پلتفرم، دانش عمیق در زمینه فناوری مربوطه است. نقطه‌ضعف، وابستگی‌هایی است که چنین رویکردی بین تیم‌هایی که با هم کار می‌کنند ایجاد می‌کند تا کل اینکریمنت محصول را طبق برنامه (Schedule) ارائه دهند.

چارچوب‌های مقیاس‌پذیر (مانند آن‌هایی که در ٤/٣/٤ نام برده شده اند) راه هایی را برای کنترل این شرایط نشان می دهند. فعالیت های مهندسی نیازمندی ها باید با چارچوب هماهنگ باشد تا به یک دیدگاه مشترک در مورد آن چه می توان ارائه داد، برسد.

در این سناریو، RE به‌ویژه نقش اساسی در تجزیه اهداف و نیازمندی‌های کسب‌وکار به نیازمندی‌های زیر سیستم تشکیل‌دهنده دارد که می‌تواند به تیم‌های جداگانه اختصاص داده شود و ثانیاً به دنبال توسعه از طریق اطمینان از نتیجه‌گیری یک راهکار یکپارچه و اطمینان از تحویل ارزش کسب‌وکار است.

ممکن است ترکیبی از انواع تیم‌ها با درنظرگرفتن محدودیت‌های خاص شرکت، منجر به ارائه بهترین و عملی‌ترین راهکار برای بهره‌مندی از ایده‌های تیم‌های برجسته شود (برای جزئیات بیشتر به [CPREALAGILE۲۰۲۲] مراجعه کنید).

٤/٣/٣ رویکردهایی برای سازماندهی ارتباط

در پاسخ به این سؤال که چند تیم می‌توانند به طور کارآمد با هم کار کنند، می‌توان بین دو رویکرد مختلف تمایز قائل شد:

١. استفاده از رویکردهای تیم واحد (Single team approaches)
٢. معرفی مفاهیم اضافی برای سازماندهی ارتباطات و مسئولیت‌ها (Communication and responsibilities)

استفاده از رویکردهای تیم واحد: رویکرد اول از این ایده پیروی می‌کند که محصولات کاری و نقش‌های اضافی موردنیاز نیستند و ارتباط بین تعدادی از تیم‌ها باید با تکنیک‌های موجود از رویکردهای تیم واحد پشتیبانی شود. نقش‌ها و محصولات کاری اضافی، مغایر تفکر چابک بوده و منجر به پیچیدگی بیشتری در سازمان می‌شود. باتوجه‌به اصل اساسی "ساده نگه دارید" ، هیچ‌گونه هزینه اضافی بر اساس نقش‌های جدید ایجاد نمی‌شود. ارتباط و هماهنگی بین

تیم‌ها معمولاً توسط مالکان محصول تیم‌ها آغاز می‌شود؛ اما توسط نمایندگان تیم (Team delegates) انجام می‌شود. ساختارهایی نظیر انجمن‌های پرکتیس (Communities of Practice)، در سراسر تیم اعضا را قادر می‌سازد تا تجربیات خود را به اشتراک بگذارند و فرایندهای کلی را هماهنگ کنند.

معرفی مفاهیم اضافی: روش دوم توصیه می‌کند که مشکلات بزرگ‌تر را به مشکلات کوچک‌تر تقسیم کنید و مسئولیت‌ها را بر اساس نقش‌های مختلف برای انتزاع‌های مختلف مدیریت کنید؛ بنابراین باید محصولات کاری اضافی برای سطوح مختلف انتزاع معرفی شوند (به‌عنوان‌مثال اپیک‌های تجاری، اپیک‌های معماری، تم‌های سرمایه‌گذاری، ویژگی‌ها، داستان‌های کاربر).

بسته به چارچوب مورداستفاده، نقش‌های دیگر مسئول در سطوح مختلف انتزاع معرفی می‌شوند (به‌عنوان‌مثال مدیران پورتفولیو، مدیران محصول و مالکان محصول ارشد). به دلیل پیچیدگی روزافزون، مصنوعات و نقش‌های اضافی برای مدیریت برنامه‌ریزی و ارتباط بین تیم‌های مختلف و دستیابی به نتایج یکپارچه محصولات کاری هر تکرار (به‌عنوان‌مثال نقشه‌های راه و مدیران انتشار) موردنیاز است. علاوه بر این، برای ارتقای ارتباط میان نقش‌های جدید و تأسیس شده، جلسات خاصی باید تشکیل شود. RE به ارائه روش‌های زیادی می‌پردازد که می‌تواند به تقسیم مشکلات بزرگ‌تر و حمایت از نقش‌های جدید در سطوح مختلف انتزاع (به‌عنوان‌مثال مدل‌سازی زمینه (Context modeling)، مدل‌سازی هدف (Goal modeling)) کمک کند.

هر دو رویکرد مزایا و معایب خود را دارند و هر کاربر / سازمان باید بهترین راه خود را بر اساس مورد کسب‌وکار (Business Case) پیدا کند.

٤/٣/٤ چارچوب‌های نمونه برای مقیاس‌پذیری RE@Agile

چارچوب‌هایی که از مقیاس‌پذیری اسکرام و چابک پشتیبانی می‌کنند: چارچوب‌های مختلفی وجود دارند که از این رویکردها پشتیبانی می‌کنند و به دلیل اهمیت روزافزون مقیاس‌پذیری، این تعداد نیز به‌سرعت درحال‌رشد است. مجموعه‌ای از معروف‌ترین چارچوب‌ها را در زیر خواهید یافت:

- **چارچوب چابک مقیاس‌پذیر (Scaled Agile Framework) (SAFe)**

  SAFe یک پایگاه دانش از الگوهای موفق اثبات شده برای پیاده‌سازی نرم‌افزاری ناب - چابک و توسعه سیستم در مقیاس سازمانی است. [SAFe]

- **اسکرام مقیاس - بزرگ (Large-Scale Scrum) (LeSS)**

  LeSS در واقع استفاده از اسکرام برای مجموعه‌ای از تیم‌ها است که به همراه یکدیگر با یک مالک محصول، بر روی یک محصول کار می‌کنند. [Less]

- **نکسِس (Nexus)**

Nexus چارچوبی است که به قلب مقیاس‌پذیری می‌رود: وابستگی‌های تیمی متقابل (Cross-team dependencies) و مسائل ادغام (Integration issues). [Nexus۲۰۱۵]

- **Scrum of Scrums**

Scrum of Scrums تکنیکی است که در آن از اسکرام برای هماهنگ‌کردن چندین تیم استفاده می‌شود. [SofS] هر تیم یک نفر (یک سفیر) را به نمایندگی از آنها در جلسات هماهنگی که معمولاً دو یا سه بار در هفته برگزار می شود، اختصاص می دهد [CPREALAGILE۲۰۲۲].

- **اسکرام در مقیاس (Scrum@Scale)**

چارچوب Scrum at Scale یک گسترش حداقلی از چارچوب اسکرام اصلی است که ساختار ماژولار را در مرکز چارچوب اسکرام حفظ کرده و اجازه می‌دهد تا یک پیاده‌سازی اسکرام متناسب با نیازهای منحصربه‌فرد شرکت شما مقیاس‌پذیر شود. [SatS].

٤/٣/٥ تأثیرات مقیاس‌پذیری در RE@Agile

لایه‌های انتزاعی که در بالا موردبحث قرار گرفتند به این معنی هستند که فعالیت‌های RE توسط نقش‌های بیشتری مانند مالک محصول ارشد، مدیر محصول، مدیر پورتفولیو و تحلیلگر کسب‌وکار مدیریت می‌شوند. هر نقشی محصولات کاری RE مربوطه را برای حوزه موردنظر خود ایجاد می‌کند (اپیک، ویژگی‌ها، داستان‌های کاربر و قابلیت ردیابی (Traceability) بین آن‌ها). درحالی‌که ارتباط با ذی‌نفعان با نقش‌های مختلف در سطوح مختلف درون سازمان انجام می‌شود، جلساتی اضافی برای فعالیت‌های RE موردنیاز خواهد بود (به‌عنوان‌مثال زمان‌های داستان (Story times)، زمان‌های دمو (Demo times)، دمو سیستم (System demo)).

ازآنجایی‌که اسکرام خود به‌راحتی مقیاس‌پذیر نمی‌شود، RE نقش مهمی در تعیین چگونگی تجزیه و تقسیم نیازمندی‌های اساسی بین چندین تیم چابک برای زنده نگه‌داشتن هر روش مقیاس‌بندی بازی می‌کند. تکنیک‌های RE در سازماندهی تجزیه‌وتحلیل مسئله و اصلاح نیازمندی‌های درشت‌دانه (Coarse-grained requirements) به نیازمندی‌های ریزدانه (Fine-grained requirements) مانند ویژگی‌ها و داستان‌های کاربر مناسب، برای تیم‌های مختلف کمک می‌کنند. یک رویکرد ساختاری، با استفاده از انتزاعات مناسب و دیدگاه‌های تحلیلی مختلف، مبنایی مناسب برای زیرمجموعه‌ای از تسک‌ها (sub-division of tasks) در میان تیم‌های نیمه‌مستقل چابک فراهم می‌کند.

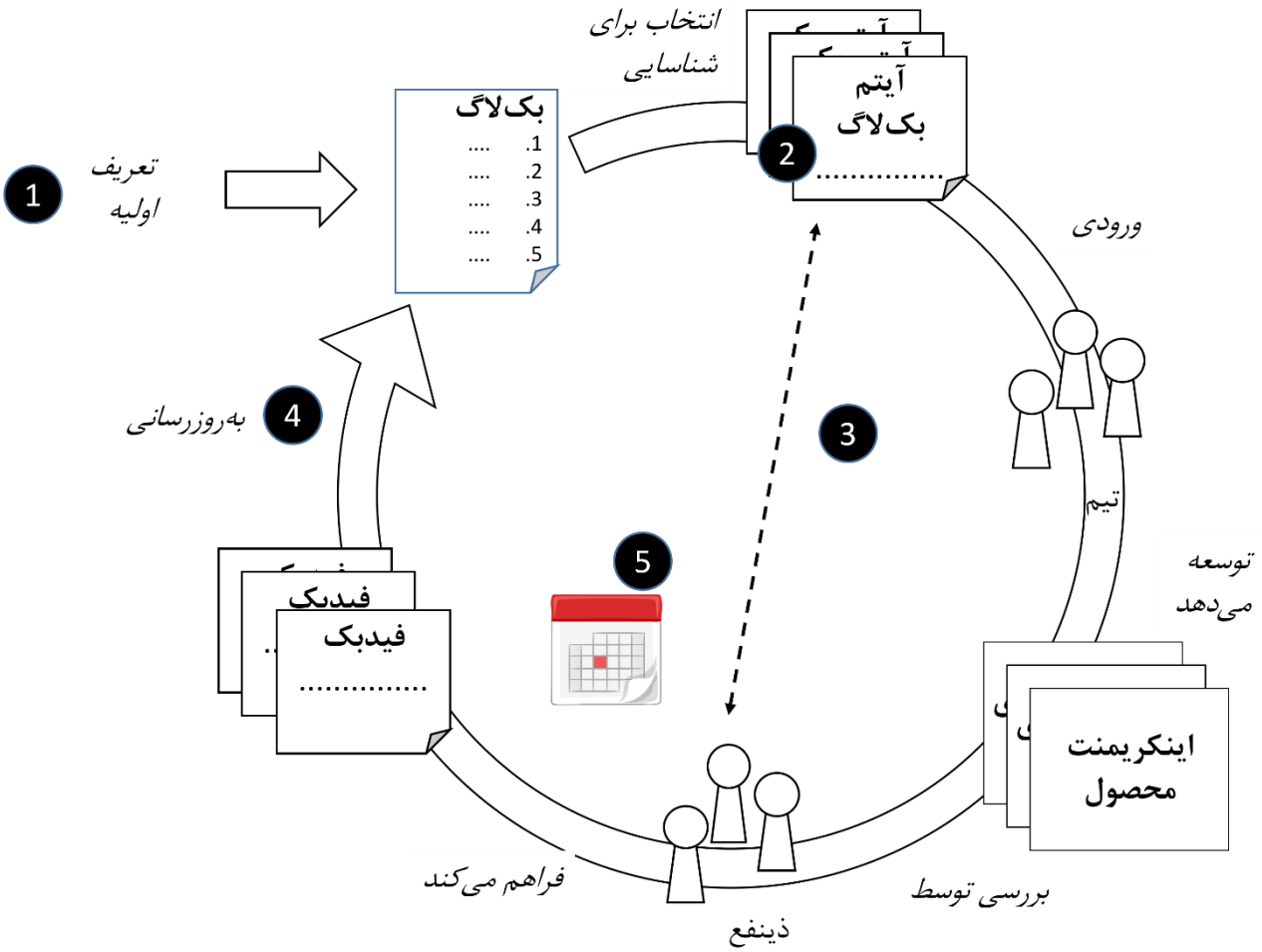
این امر به‌ویژه در مورد وابستگی‌هایی که باید دراسرع‌وقت پیدا و موردبحث قرار گیرند، برای جلوگیری از اتلاف در توسعه اعمال می‌شود.

اگر تیم‌ها یا نقش‌های دیگر در مکان‌های مختلف کار کنند، یک چالش اضافی رخ خواهد داد. پیچیدگی مدیریت ارتباطات، تنها به دلیل شیفت زمان ممکن یا زبان‌های مختلف افزایش نمی‌یابد. در بیشتر موارد، چالش اصلی بر

اساس فرهنگ‌های مختلف است. ازآنجاکه ارتباطات یکی از وظایف اصلی مالکان محصول و مدیران محصول است، این موضوع تأثیر مهمی در مهارت‌های موردنیاز نقش‌های مربوط به RE دارد.

٤٫٤ متعادل‌سازی مهندسی نیازمندی‌های مقدماتی و مداوم در زمینه مقیاس‌پذیری (L1)

در سطح کاملاً انتزاعی، روش‌های چابک را می‌توان به‌عنوان یک فرایند تکرارشونده و مداوم توصیف کرد که در آن سیستم به‌تدریج بر اساس موارد بک‌لاگ توسعه می‌یابد. از منظر مهندسی نیازمندی‌ها، می‌توان پنج پارامتر (فصل‌های فرعی زیر را ببینید) را شناسایی کرد که این فرایند را هدایت می‌کند:



شکل 2: فرایند مهندسی نیازمندی های مداوم

٤/٤/١ تعریف اولیه نیازمندی‌ها

قبل از شروع فرایند توسعه مداوم، باید یک بک‌لاگ اولیه ایجاد شود ([TaNo۱۹۸٦] را مشاهده کنید). این تعریف اولیه از بک‌لاگ اغلب به‌عنوان "تعریف مقدماتی" (Upfront definition) شناخته می‌شود. گاهی اوقات مهندسی نیازمندی‌ها به این صورت درک می‌شود که این تعریف اولیه، مشخصات کامل و دقیق کلیه نیازمندی‌ها است. ممکن است این مورد برای برخی از مدل‌ها یا دامنه‌های فرایندی باشد، اما همیشه این‌طور نیست و در توسعه Agile نسبتاً غیرمعمول است.

مبنای تعریف اولیه بک‌لاگ، اطلاعاتی است که برای شروع تکرار اول فرایند توسعه مداوم موردنیاز است. اینکه چه موقع فرایند تکرار (Iteration process) را شروع خواهید کرد، یک تصمیم اساسی است. بسته به سیستم و زمینه، عدم اطمینان در مورد نیازمندی‌ها (شروع بدون دانش کافی) ممکن است به تأخیرهای اضافی، هزینه‌ها یا به طور بالقوه به شکست پروژه منجر شود. برعکس، شروع خیلی دیر ممکن است تأثیر منفی بر زمان بازاریابی و مناسب‌بودن نیازهای نهایی کاربران در یک‌زمان خاص داشته باشد.

مهندسی نیازمندی‌ها به ما می‌گوید که باید آن دسته از نیازمندی‌هایی که می‌تواند تأثیر زیادی بر معماری، امکان‌سنجی کلی راه‌حل و یا گزینه‌های اصلی مربوط به زیرساخت‌ها و سخت‌افزارها داشته باشد، شناسایی کرد و هر چه زودتر آن‌ها را تشریح کرد. چنین مسائل مربوط به معماری در حقیقت، باید قبل از تکرار اول توسعه استخراج و تحلیل شود. نیازمندی‌های با اثر کمتر (Lower impact) ممکن است در طول تکرارها، در قالب یک فرایند مداوم مورد اصلاح واقع شوند.

در فرایندهای تکرارشونده و تدریجی چابک، مهندسی نیازمندی‌ها به یک فرایند مداوم تبدیل می‌شود که نیازمندی‌ها را به‌موقع و با جزئیات کافی برای تغذیه چرخه توسعه ارائه می‌دهد. این فرایند به یک قیف (Funnel) خردکردن سنگ شباهت دارد، جایی که سنگ‌های بزرگ وارد قیف می‌شوند و تا زمان اتمام فرایند به اندازه‌های متوسط خرد می‌شوند و در نهایت به سنگ‌های کوچک خرد می‌شوند.

٤/٤/٢ سطح جزئیات برای آیتم‌های بک‌لاگ

سطح جزئیات یک آیتم بک‌لاگ، آزادی توسعه‌دهندگان را برای تحقق آیتم بک‌لاگ محدود می‌کند ([Pich۲۰۱۰] را مشاهده کنید). تمام جنبه‌هایی از آیتم بک‌لاگ که هنوز تعریف نشده‌اند، به‌عنوان تصمیماتی برای تیم باقی می‌ماند که باید به آن اجازه دهد تا در مرزهای مشخص شده کسب‌وکار، خلاقیت بیشتری داشته باشد.

صلاحیت‌های موجود توسعه‌دهندگان می‌تواند به‌عنوان یک اصل کلی عمل کند. در صورتی که توسعه‌دهندگان صلاحیت و تجربه کافی برای تصمیم‌گیری در مورد جزئیات یک آیتم بک‌لاگ را دارد (به‌عنوان‌مثال، یک متخصص احراز

هویت (Authentication) و مجوز (Authorization) در میان توسعه‌دهندگان است)، تصمیم‌گیری در مورد جزئیات باید به توسعه‌دهندگان سپرده شود. همچنین ۳٫۱٫۵ را مشاهده کنید.

۴٫۴٫۳ اعتبار آیتم‌های بک‌لاگ

دانستن اعتبار (Validity) یک آیتم بک‌لاگ قبل از اجرا، به کاهش کارهای غیرضروری پیاده‌سازی کمک می‌کند ([Denn۲۰۱۵] را مشاهده کنید). انتظار برای تشخیص یک نیازمندی غلط یا ناقص در نرم‌افزار پیاده‌سازی شده، یک رویکرد بسیار گران‌قیمت برای اعتبارسنجی نیازمندی‌ها است.

تعیین اعتبار یک آیتم بک‌لاگ، ارتباط نزدیکی با سطح جزئیات آن آیتم مدنظر دارد. اعتبار یک آیتم بک‌لاگ محصول فقط زمانی مشخص می‌شود که آیتم بک‌لاگ از جزئیات کافی برخوردار باشد؛ بنابراین، تلاش برای شرح و اعتبارسنجی یک نیازمندی قبل از اجرا، باید با تلاش فرضی برای اجرای این نیازمندی و اعتبارسنجی پس از آن، در نرم‌افزار تحویل شده مقایسه شود.

وقتی می‌توان اولویت ذی‌نفع مربوط به یک آیتم بک‌لاگ را با تلاشی قابل‌قبول تعیین کرد، تأیید چنین شرایطی باید قبل از توسعه آیتم بک‌لاگ انجام شود.

نمونه‌های رایج این نوع نیازمندی (از جمله رویکرد اعتبارسنجی نمونه‌ای (Exemplary validation approach)) عبارتند از: طراحی کلی رابط کاربری (به‌عنوان‌مثال با موکاپ‌های رابط کاربر (UI mockups))، سازوکارهای مجوز (Authorization) و تأیید اعتبار (Authentication) (به‌عنوان‌مثال با استفاده از بررسی موارد کاربرد (Use case reviews))، ساختارهای داده‌ای که باید در سیستم (به‌عنوان‌مثال با بررسی مدل داده) و نیازمندی‌های رابط‌های سیستم‌های موجود (به‌عنوان‌مثال با بررسی نمودار فعالیت (Activity diagram)).

آیتم‌های بک‌لاگی که دارای ریسک بالایی هستند (به‌عنوان‌مثال عملکردهای حیاتی کسب‌وکار (Critical business functionalities)، عملکردهای حیاتی ایمنی (Safety critical functions)، عملکردهای نوآورانه (Innovative functionalities)) یا آیتم‌هایی که هزینه تست بالایی برای اجرا دارند (به‌عنوان‌مثال نرم‌افزار باید در نمونه اولیه‌ای گران‌قیمت مورد آزمون واقع شود)، باید قبل از اجرا، اعتبارسنجی شوند.

۴٫۴٫۴ بازخورد و به‌روزرسانی بک‌لاگ

بک‌لاگ‌ها اغلب بر اساس بازخورد از یک فعالیت بازرسی (Inspection activity) مانند بررسی اسپرینت در اسکرام، به‌روز می‌شوند. چنین رویکردی برای مقیاس‌های کوچک یا نیازمندی‌های جزئی امکان‌پذیر است که می‌توان در یک محیط با یک یا دو تیم توسعه، تأثیر یک تغییر را تحلیل و درک کرد. اصلاح نیازمندی‌هایی که از پیچیدگی بیشتری برخوردار هستند و یا وابستگی‌های متعددی در کوتاه‌مدت دارند، توصیه نمی‌شود. در چنین شرایطی که اصلاح آیتم‌های بک‌لاگ زمان بیشتری می‌برد، ممکن است ذی‌نفعان در دسترس نباشند و تجزیه‌وتحلیل اضافی لازم باشد.

عامل دیگری که ممکن است در تغییر آیتم‌های بک‌لاگ تأثیر بگذارد، فرایند تصمیم‌گیری سازمان است. در سازمان‌هایی که گرفتن تصمیمات مهم ممکن است مدتی طول بکشد (به‌عنوان‌مثال شورای مسئول فقط هر سه ماه یکبار تشکیل جلسه می‌دهد)، اصل پالایش مداوم (Continuous refinement) باید به شکل جلسات مشخصی با مشارکت همه ذی‌نفعان تأثیرگذار باشد. چنین جلساتی، به مهندسی نیازمندی‌ها برای آماده‌سازی و پشتیبانی از تصمیم‌گیری نیاز دارند.

٤/٤/٥ زمان چرخه توسعه

برنامه زمانی (Time schedule) یا طول تکرار (Length of the iteration)، پارامتر نهایی برای فرایند توسعه است. این مورد، تأثیر مهمی بر روی فعالیت‌های مهندسی نیازمندی‌هایی دارد که باید هنگام کار روی مواردی که هنوز در دست توسعه نیستند، اجرا شوند. علاوه‌براین، طول تکرار، توالی تحویل نتایج به ذی‌نفعان کسب‌وکار یا مشتریان در دسترس را برای بررسی تعیین می‌کند.

در نتیجه، تکرارهای کوتاه‌تر به سه دلیل باعث افزایش حجم کاری ذی‌نفعان کسب‌وکار می‌شوند (برای مثال مقایسه کنید با [Rein۱۹۹۷])

۱. ذی‌نفعان کسب‌وکار باید در کل تکرار برای کار بر روی آیتم‌های بک‌لاگ در دسترس باشند تا ورودی برای تکرار بعدی ایجاد کنند.
۲. ذی‌نفعان کسب‌وکار باید نتایج ایجاد شده توسط تیم را برای ارائه بازخورد بررسی کنند.
۳. ذی‌نفعان کسب‌وکار از تعویض‌های زمینه (Context switches) بین کسب‌وکار روزانه (Daily business) و کار پروژه (Project work) رنج می‌برند.

با این که طول‌های تکرار بیشتر (Longer iteration lenghts) موجب کاهش فشار می‌شوند؛ اما موجب کاهش توانایی تأثیر (Capability of influencing) بک‌لاگ برای تولید محصول می‌شوند.

تعریف طول تکرار باید با درنظرداشتن ذی‌نفعان کسب‌وکار انجام شود. ذی‌نفعان کسب‌وکار به طور معمول به‌صورت ۱۰۰٪ برای فعالیت های توسعه در دسترس نیستند، زیرا آن ها وظایف دیگری در سازمانی دارند که سیستم برای آن توسعه یافته است.

به‌عنوان یک قاعده کلی، زمان تکرار کوتاه‌تر بازخورد مکرر و فرصت‌های بیشتری برای کشف زودهنگام خطاها را فراهم می‌کند، بنابراین تکرارهای کوتاه‌تر سرعت فعالیت‌های توسعه را افزایش می‌دهند. اگر زمان چرخه کوتاه بار غیرقابل‌قبولی را برای ذی‌نفعان به همراه داشته باشد، باید طول تکرار مناسب پیدا شود، اگرچه ممکن است این تکرار متناسب با اولویت پروژه کوتاه شود.

عامل دیگری که می‌تواند در زمان چرخه (Cycle time) تأثیر بگذارد، متوسط اندازه و پیچیدگی آیتم‌های بک‌لاگ است. آیتم‌های بک‌لاگ بزرگ‌تر یا پیچیده‌تر، زمان بیشتری را برای درک و تجزیه‌وتحلیل صرف می‌کنند؛ بنابراین زمان چرخه در جهت رسیدگی به آیتم‌های بزرگ‌تر یا پیچیده‌تر، می‌تواند افزایش پیدا کند. به‌عنوان‌مثال، اگر سیستم درحال‌توسعه در مراحل اولیه باشد، ممکن است یک‌چرخه طولانی‌تر توصیه شود تا تیم به زمان بیشتری برای درک اولیه سیستم برسد. بااین‌وجود، این عامل باید باهدف استفاده از زمان‌های چرخه کوتاه‌تر برای دریافت بازخورد مکرر متعادل شود. تصمیم‌گیری در مورد داشتن زمان‌های چرخه طولانی‌تر یا کوتاه‌تر باید با هم به‌عنوان تیمی انجام شود که نیاز به تجزیه‌وتحلیل باهدف بازخورد زودرس را ارزیابی می‌کند. باتوجه‌به اینکه تجربه گذشته همیشه تضمینی برای آینده نیست، تغییر زمان چرخه همیشه بر اساس اصل "بازرسی و سازگاری" (Inspect & Adapt) است. تغییر زمان چرخه هرگز نباید در داخل تکرار صورت پذیرد و تنها قبل از آغاز آن امکان‌پذیر خواهد بود. علاوه بر این، زمان چرخه باید تاحدامکان سازگار باشد تا پیچیدگی کاهش یابد و قابلیت پیش‌بینی پایدار ایجاد شود. تغییر بیش از حد مدت‌زمان ممکن است باعث بی‌ثباتی، کاهش تعهد تیم و به‌احتمال زیاد مانع از سرعت افزایش محصول شود.

۵ تعاریف اصطلاحات، واژه‌نامه (L2)

واژه‌نامه به تعریف اصطلاحات مرتبط با زمینه RE@Agile Primer می‌پردازد. واژه‌نامه در صفحه خانگی IREB در https://www.ireb.org/en/downloads/#re-agile-glossary برای دانلود در دسترس است.

٦ منابع

[ACP/PMI] Agile Certified Practitioner: http://www.pmi.org/certification/Agile-management-.acp.aspx. Last visited June ٢٠٢٤

[AgileMan٢٠٠١] .Agile Manifesto: http://Agilemanifesto.org, ٢٠٠١. Last visited June ٢٠٢٤

[AmLi٢٠١٢] Ambler S.; Lines, M.: Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press, ٢٠١٢

[Ande٢٠١٢] ,Anderson, D.J.: Lessons in Agile Management: On the Road to Kanban. Blue Hole Press ٢٠١٢

[Appel٢٠١١] Appelo J.: Management ٣,٠. Addison-Wesley Professional, ٢٠١١

[Beck٢٠٠٣] Beck, K.: Test-Driven Development by Example. Addison Wesley – Vaseem, ٢٠٠٣

[Beck٢٠٠٤] Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, ٢٠٠٤

[CNYM٢٠٠٠] Chung, L.; Nixon, B. A.; Yu, E.; Mylopoulos, J. : Non-Functional Requirements in Software Engineering. Springer Science & Business Media, ٢٠٠٠

[Cock١٩٩٨] Cockburn, A.: Surviving Object-Oriented Projects. Addison-Wesley, ١٩٩٨

[Cohn٢٠٠٤] Cohn, M.: User Stories Applied: For Agile Software Development. Addison Wesley Professional, ٢٠٠٤

[Conw١٩٦٨] Conway, M.: How Do Committees Invent? Datamation ١٤(٤):٢٨–٣١, ١٩٦٨. Article available at http://www.melconway.com/Home/Conways_Law.html. Last visited June ٢٠٢٤.

[CPREFL٢٠٢٢] .IREB e.V.: Syllabus CPRE Foundation Level, version ٣,١ https://www.ireb.org/downloads/#cpre-foundation-level-syllabus-٣-٠. Last visited June ٢٠٢٤.

.[CPREALAGILE٢٠٢٢] IREB e.V.: Syllabus CPRE RE@Agile Practitioner | Specialist, version ٢,٢,٠ https://www.ireb.org/downloads/#cpre-advanced-level-re-agile-syllabus. Last visited .June ٢٠٢٤

[Denn٢٠١٥] .Denning, S.: How To Make The Whole Organization Agile http://www.forbes.com/sites/stevedenning/٢٠١٥/٠٧/٢٢/how-to-make-the-whole-.organization-agile/#٦٥٨d٣f٦٥١٣٥b, ٢٠١٥. Last visited June ٢٠٢٤

[Dsch٢٠١٥] .d.school: An Introduction to Design Thinking – Process Guide https://web.stanford.edu/~mshanks/MichaelShanks/files/٥٠٩٥٥٤.pdf, ٢٠١٥. Last visited .June ٢٠٢٤

[Glinz٢٠٢٢] ,Glinz, M.: A Glossary of Requirements Engineering Terminology .https://www.ireb.org/downloads/#cpre-glossary. Last visited June ٢٠٢٤

[Griff٢٠١٥] Griffiths, M.: PMI-ACP Exam Prep. Rmc Publications, ٢٠١٥

[GoAk٢٠٠٣] Gordijn, J.; Akkermans, J.M.: Value-based Requirements Engineering: exploring innovative e-commerce ideas. Springer, ٢٠٠٣

[High٢٠٠٩] Highsmith, J.: Agile Project Management: Creating Innovative Products. Addison-Wesley Professional, ٢٠٠٩

[ISO٢٥٠١٠] ISO/IEC Systems and software engineering - Systems and software Quality Requirements and Evaluation. ISO/IEC Standard ٢٥٠١٠:٢٠١١

[KnZK٢٠١٦] Knapp, J.; Zeratsky, J; Kowitz, B.: Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days. Simon & Schuster, ٢٠١٦

[Kron٢٠٠٨] Kronfaelt, R.: Ready-ready: the Definition of Ready for User Stories going into Sprint planning. http://scrumftw.blogspot.de/٢٠٠٨/١٠/ready-ready-definition-of-ready-.for.html, ٢٠٠٨. Last visited June ٢٠٢٤

[Less] Large Scale Scrum: http://less.works. https://less.works/resources/LeSS-brochure.pdf .Last visited June ٢٠٢٤

.[LiOg٢٠١١] Liedtka, J.; Ogilvie, T.: Designing for Growth: A Design Thinking Tool Kit For Managers Columbia Business School Publishing, ٢٠١١

[Martin١٩٩١] Martin, J.: Rapid Application Development. Macmillan Coll Div, ١٩٩١

[Meyer٢٠١٤] Meyer, B.: Agile! – the good, the hype, and the ugly. Springer, ٢٠١٤

[MeMi٢٠١٥] ,Mesaglio, M., Mingay, S.: Bimodal IT: How to Be Digitally Agile Without Making a Mess .Gartner ٢٠١٥, https://www.gartner.com/en/documents/٢٧٩٨٢١٧. Last visited June ٢٠٢٤

[Nexus٢٠١٥] Nexus Guide https://www.scrum.org/Portals/٠/NexusGuide%٢٠v١,١.pdf, ٢٠١٥. Last .visited June ٢٠٢٤

[Patt٢٠١٤] Patton, J.: User Story Mapping. O'Reilly, ٢٠١٤

[Pich٢٠١٠] Pichler, R: Make the product backlog deep. http://www.romanpichler.com/blog/make-.the-product-backlog-deep/, ٢٠١٠. Last visited June ٢٠٢٤

[PMI] .PMI Project Management Institute. http://www.pmi.org/. Last visited June ٢٠٢٤

[Popp٢٠٠٣] Poppendieck, M.: Lean Software Development: An Agile Toolkit. Addison-Wesley Professional, ٢٠٠٣

[Rein١٩٩٧] & Reinerstsen, D. G.: Managing the Design Factory – A Product Developer's Toolkit. Simon Schuster, ١٩٩٧

[Ries٢٠١١] Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Publishing Group, ٢٠١١

[SAFe] SAFe – Scaled Agile Framework. http://www.scaledAgileframework.com. Last visited June .٢٠٢٤

[SatS] Scrum at Scale Framework: https://www.scruminc.com/scrum-incs-scrum-at-scale-.framework/. Last visited June ٢٠٢٤

[Scrum٢٠٢٠] Schwaber, K. & Sutherland, J.: The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game, July ٢٠٢٠. https://scrumguides.org/docs/scrumguide/v٢٠٢٠/٢٠٢٠-Scrum-.Guide-US.pdf. Last visited June ٢٠٢٤

[ShYo٢٠٠٦] .Sheppard, J. M.; Young W. B.: Agility literature review: Classifications, training and testing Journal of Sports Sciences ٢٤(٩): ٩١٩-٩٣٢, ٢٠٠٦

[SofS] ,Scrum of Scrums https://scrumguide.de/scrum-of-scrums. Last visited June ٢٠٢٤ .available in German only

[TaNo١٩٨٦] Takeuchi, H.; Nonaka, I.: The new new product development game. Harvard Business Review ٦٤(١), January/February ١٩٨٦, p.١٣٧-١٤٦

[Wake٢٠٠٣] Wake, B.: Invest in Good Stories and Smart Tasks, http://xp١٢٣.com/articles/invest-in-.good-stories-and-smart-tasks/. Last visited June ٢٠٢٤