

CPRE

Certified Professional for Requirements Engineering

RE@Agile Primer

سرفصل و راهنمای یادگیری

Baumann, Peter Hruschka,
Kim Lauenroth, Markus Meuten,
Sasch Reis, Gareth Rogers,
François Salazar,
Hans-Jörg Steffe, Thorsten Weyer

1. افراد و آموزش دهندگان ممکن است از این برنامه درسی و راهنمای یادگیری به عنوان 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 بین المللی مهندسی نیازمندی‌ها احتیاج دارد.
2. فرد یا گروهی از افراد می‌توانند از این برنامه درسی و راهنمای یادگیری به عنوان پایه‌ای برای مقالات، کتاب‌ها یا سایر نشریات مشتق شده استفاده کنند، به شرطی که داشتن حق چاپ برای نویسندگان و IREB e.V به عنوان منبع و صاحب این سند در چنین نشریاتی تایید شده باشد.

© IREB e.V.

تمام حقوق محفوظ است. هیچ بخشی از این مستند بدون اجازه کتبی قبلی از نویسندگان و یا IREB e.V قابل چاپ و ذخیره‌سازی در سیستم‌های بازتابی یا هر شکل و وسیله دیگری نظیر وسایل الکترونیکی، مکانیکی، فتوکپی، ضبط کردن یا موارد دیگر نیست.

قدردانی

این سیلابس و راهنمای آموزشی توسط: Lars Baumann, Peter Hruschka, Kim Lauenroth, Markus Meuten, Sacha Reis, Gareth Rogers, François 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 شده اند.

نقد و بررسی انگلیسی توسط Joy Beatty, Gareth Rogers and Candase Hokanson انجام شده است. ترجمه انگلیسی به فارسی این متن توسط آیدین ضیاپور سَهی، آراز ساعی ارسنی، حسن حقیقی و محمود نشاطی انجام شده است. ما از تمامی این افراد بابت مشارکتشان تشکر می‌کنیم.

در تاریخ 2 مارچ 2017 به توصیه Xavier Franch توسط شورای IREB تایید شده است.

حق کپی رایت © 2016-2026 این سرفصل و راهنمای یادگیری برای نویسندگان ذکر شده در بالا محفوظ است. این حقوق به بورد بین المللی مهندسی نیازمندی‌ها (IREB) منتقل شده است.

هدف از این مستند

این سرفصل و راهنمای یادگیری به تعریف سطح مبانی اولیه گواهینامه حرفه‌ای RE@Agile که توسط بورد بین المللی مهندسی نیازمندی‌ها (IREB) معرفی شده است، می‌پردازد. این سرفصل و راهنمای یادگیری، مقدمات طراحی و ایجاد مطالب درسی را برای آموزش دهندگان 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 توسعه دهندگان نرم افزار 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 طبقه بندی و ارزیابی محصولات کاری و تکنیک‌های مهندسی نیازمندی‌ها در زمینه چابک، محصولات کاری و تکنیک‌های چابک در زمینه مهندسی نیازمندی‌ها و عناصر اساسی فرآیند در توسعه محصول چابک است. RE@AGILE به انگیزه استفاده از چابک در یک فرآیند توسعه اشاره می‌کند.

یک موضوع بسیار، مهم هم افزایی بین مهندسی نیازمندی ها و چابک است: اصول چابک در مورد مهندسی نیازمندی ها و طرز فکر چابک در رابطه با ارزش های اصلی مهندسی نیازمندی ها.

مدارک RE@Agile مورد بین المللی مهندسی نیازمندی ها، حمایت از گروه های زیر را به عنوان هدف خود قرار داده است:

- مهندسین نیازمندی هایی که می خواهند در توسعه چابک نقش داشته باشند و قصد دارند تکنیک های خود را با موفقیت در این محیط به کار گیرند.
- مهندسین نیازمندی هایی که می خواهند مفاهیم و تکنیک های تثبیت شده از رویکردهای چابک را برای بهبود فرایندهای مهندسی نیازمندی ها خود استفاده کنند.
- متخصصان چابکی که می خواهند ارزش و مزایای دیسپلین مهندسی نیازمندی ها را در پروژه های چابک درک کنند.
- متخصصان چابکی که می خواهند با استفاده از تکنیک ها و روش های اثبات شده مهندسی نیازمندی ها، توسعه چابک را بهبود بخشند.
- افرادی از رشته های مرتبط - مدیران فناوری اطلاعات، آزمون گران (Testers)، توسعه دهندگان، معماران و سایر نمایندگان مشاغل درگیر در توسعه (عمدتاً، اما نه تنها توسعه نرم افزار) - که می خواهند بدانند چگونه می توان رویکردهای مهندسی نیازمندی ها 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 بین المللی می شود. 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 اختصاص داده شده است. سطوح بالاتر شامل سطوح پایین تر می شوند. فرموله کردن اهداف آموزشی با استفاده از افعال "دانستن" برای سطح L1 و "تسلط و استفاده" برای سطح L2 بیان می شود. این دو فعل به افعال زیر اشاره دارند:

کلیه واحدها و اهداف آموزشی در این سرفصل به یک سطح شناختی اختصاص داده شده اند. سطوح زیر در این سرفصل مورد استفاده قرار گرفته اند:

- **L1: دانستن** (توصیف، برشمردن، مشخص کردن، شناسایی، نامگذاری، به یاد آوردن، ...) - یادآوری یا بازیابی مطالب قبلاً آموخته شده.
 - **L2: درک** (تشریح کردن، تفسیر کردن، تکمیل کردن، خلاصه کردن، توجیه کردن، طبقه بندی، مقایسه کردن، ...) - درک / ساختن معنی از مواد یا موقعیت های داده شده.
 - **L3: اعمال** (مشخص کردن، نوشتن، طراحی، توسعه دادن، پیاده سازی، ...) - اعمال دانش و مهارت در موقعیت های معین.
- توجه داشته باشید که یک هدف یادگیری در سطح دانش شناختی Ln حاوی عناصر تمام سطوح شناختی زیر آن (L1 تا Ln-1) است.

مثال:

یک هدف یادگیری از نوع "به کارگیری تکنیک مهندسی نیازمندی های xyz" در سطح دانش شناختی (L3) است. با این حال، توانایی اعمال مستلزم آن است که فراگیران تکنیک مهندسی نیازمندی های xyz سطح (L1) را بدانند و بدانند که این تکنیک برای چیست (L2).

تمامی اصطلاحات تعریف شده در واژه نامه CPRE و راهنمای اسکرام (SCRUM Guide) باید مورد یادگیری قرار گیرند (L1)، حتی اگر به صراحت در اهداف آموزشی ذکر نشده باشند.



واژه نامه CPRE برای دانلود در صفحه خانگی وبسایت IREB در دسترس است:

<https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-glossary>

راهنمای اسکرام (SCRUM Guide) برای دانلود در دسترس است:

<https://scrumguides.org/download.html>

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

ساختار سرفصل و راهنمای یادگیری

این سرفصل و راهنمای یادگیری از 4 بخش اصلی تشکیل شده است. هر بخش یک واحد درسی (EU) را پوشش می‌دهد. عنوان فصل اصلی شامل سطح شناختی فصل‌های آن‌ها 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: فصل ۲ مبانی RE@Agile (L1)

مدت زمان: 1 4/1 ساعت

اصطلاحات: مالک محصول، بک لاگ محصول، بک لاگ اسپرینت، اپیک‌ها، داستان‌های کاربر، نقشه‌های داستان

این مثال نشان می‌دهد که فصل 2 شامل اهداف آموزشی در سطح L1 است و 75 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. شماره گذاری آن‌ها نشان می‌دهد که به کدام زیر فصل تعلق دارند.

مثال: EO 3.1.2

این مثال نشان دهنده این است که هدف آموزشی EO 3.1.2 در زیر فصل 3.1 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 چند گزینه ای به همراه مدرک رسمی RE@AGILE Primer، شبیه آزمون‌های چند گزینه ای سطح مبانی CPRE و ماژول RE@AGILE Practitioner، اما با مدت زمان 40 دقیقه.
- آزمون خود ارزیابی چند گزینه ای آنلاین به همراه گواهی شرکت در آزمون.

امتحانات نظارتی می‌توانند بلافاصله پس از یک دوره آموزشی یا به صورت مستقل از دوره‌ها (به عنوان مثال، در یک مرکز امتحان) برگزار شوند. لیستی از ارائه دهندگان شناخته شده آزمون را می‌توان در صفحه اصلی IREB پیدا کرد.

آزمون خود ارزیابی از طریق صفحه اصلی IREB در دسترس خواهد بود: <https://cpre.ireb.org/en/training-certification/online-self-assesment>



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 1 تا EU 4) می‌توانند مورد آزمون قرار گیرند.

نسخه	تاریخ	نظر
1.0.0	1 سپتامبر 2021	اولین نسخه فارسی
1.2.0	ژانویه 2023	<ul style="list-style-type: none"> • رفع باگ ها • همسویی با راهنمای اسکرام 2020 • همسویی با نسخه 3 سطح مقدماتی CPRE و نسخه 2 سطح پیشرفته CPRE RE@Agile • سطوح دانش شناختی با توجه به تعریف جدید IREB اعمال شده است. • همسویی اهداف آموزشی با سطوح دانش شناختی جدید. <p>زمان پیشنهادی تدریس واحدهای آموزشی متناسب با سطوح دانش شناختی جدید اصلاح شده است.</p>
1.3.0		•
1.4.0	ژوئن 2024	<ul style="list-style-type: none"> • رفرنس‌های دارای مشکل اصلاح شدند. • طراحی سازمانی جدید اعمال شد.
1.5.0	ژوئن 2026	<ul style="list-style-type: none"> • به روزرسانی به دلیل حذف واژه نامه RE@Agile (به روزرسانی CPRE) و به روزرسانی در مراجع.

1	انگیزه و ذهنیت ها (L1)	11
1.1	انگیزه استفاده از چابک (L1)	11
1.2	طرز فکر و ارزش ها در مهندسی نیازمندی ها و چابک (L1)	12
1.3	اتصال مهندسی نیازمندی ها و چابک در راستای (L1) RE@Agile	13
1.4	مزایا، باورهای غلط و مشکلات استفاده از RE@Agile (مهندسی نیازمندی ها در چابک) (L1)	15
1.4.1	مزایای RE@Agile:	15
1.4.2	تصورات غلط RE@Agile	16
1.4.3	دام های RE@Agile	17
1.5	RE@Agile و کار مفهومی (L1)	18
2	RE@Agile (L2) مبانی	20
2.1	رویکردهای چابک (مرور کلی) (L1)	20
2.2	اسکرام (به همراه پرکنش های خوب) به عنوان مثال (L1):	21
2.3	تفاوت ها و اشتراک های میان مهندسی نیازمندی ها و مالکان محصول (L2)	23
2.4	مهندسی نیازمندی ها به عنوان یک فرآیند مداوم (L2)	25
2.5	توسعه مبتنی بر ارزش (L1)	25
2.6	سادگی به عنوان مفهومی ضروری (L1)	25
2.7	بازرسی و سازگاری (L1)	26
3	RE@AGILE (L2) محصولات کاری و تکنیک ها در	27
3.1	محصولات کاری در RE@Agile (L2)	27
3.1.1	مستند مشخصات (Specification Documents) در مقابل بک لاگ محصول (Product Backlog)	27
3.1.2	چشم انداز و اهداف	28
3.1.3	مدل زمینه (Context model)	29
3.1.4	نیازمندی ها	29
3.1.5	جزئیات نیازمندی ها	30
3.1.6	مدل های گرافیکی و توضیحات متنی	32
3.1.7	تعریف اصطلاحات، واژه نامه ها و مدل های اطلاعاتی	32
3.1.8	نیازمندی های کیفی و قیدها	33
3.1.9	معیار پذیرش (Acceptance criteria) و معیار تناسب (Fit criteria)	33
3.1.10	تعریف آماده (Definition of Ready) و تعریف انجام شده (Definition of Done)	34
3.1.11	نمونه اولیه در مقابل اینکریمنت ها (Prototype vs Increments)	34

3.1.12	خلاصه محصولات کاری	35
3.2	تکنیک های <i>RE@Agile (L2)</i>	35
3.2.1	استخراج نیازمندی ها	36
3.2.2	مستندسازی نیازمندی ها	36
3.2.3	اعتبارسنجی و مذاکره نیازمندی ها	38
3.2.4	مدیریت نیازمندی ها	39
4	جنبه های سازمانی <i>RE@Agile (L2)</i>	41
4.1	تأثیر سازمان بر <i>RE@Agile (L2)</i>	41
4.2	توسعه چابک در محیط غیر چابک (<i>L1</i>)	42
4.2.1	تعامل با ذی نفعان خارج از سازمان فناوری اطلاعات	42
4.2.2	سازمان مبتنی بر محصول در مقابل سازمان مبتنی بر پروژه	43
4.2.3	نقش مدیریت در زمینه چابک	43
4.3	رسیدگی به مشکلات پیچیده با مقیاس پذیری (<i>L1</i>)	44
4.3.1	انگیزه مقیاس پذیری	44
4.3.2	رویکردهایی برای سازماندهی تیم ها	45
4.3.3	رویکردهایی برای سازماندهی ارتباط	45
4.3.4	چارچوب های نمونه برای مقیاس پذیری <i>RE@Agile</i>	46
4.3.5	تأثیرات مقیاس پذیری در <i>RE@Agile</i>	46
4.4	متعادل سازی مهندسی نیازمندی های مقدماتی و مداوم در زمینه مقیاس پذیری (<i>L1</i>)	47
4.4.1	تعریف اولیه نیازمندی ها	48
4.4.2	سطح جزئیات برای آیتم های بک لاگ	49
4.4.3	اعتبار آیتم های بک لاگ	49
4.4.4	بازخورد و به روزرسانی بک لاگ	49
4.4.5	زمان چرخه توسعه	50
5	تعاریف اصطلاحات، واژه نامه (<i>L2</i>)	51
6	منابع	52

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 (Requirements Engineering discipline) در استفاده از آن‌ها در روش‌های خاص توسعه (مانند مدل آشناری یا اسکرام) بسیار مفید هستند با این حال، مهندسی نیازمندی‌ها اغلب به عنوان یک دیسیپلین غیر چابک 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 جبه‌ای اینکه با هم در نظر 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 قائل شدن به صلاحیت مهندسی نیازمندی‌ها (Requirements 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. از این رو ایجاد یک پل کاملاً پذیرفته شده میان مهندسی نیازمندی‌ها و رویکردهای چابک و در نتیجه، همچنین بین مهندسی نیازمندی‌ها 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.

درباره RE@Agile

چنین پلی باید از دو جهت مختلف ساخته شود: از یک طرف، 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 پرکنش مهندسی نیازمندی‌ها (Requirements Engineering practice) استفاده کنند. از طرف دیگر، از آنجا که هدف رویکردهای چابک ارائه نرم‌افزاری با ارزش در اسرع وقت 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 تجزیه و تحلیل 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 به صورت سیستماتیک انجام شود و به سطح بالایی از مهارت و تجربه نیاز دارد.

- همکاری نزدیک در تیم و با ذینفعان اصلی و نیازمندی‌های به موقع (*Just-in-time-requirements*) ارتباط مکرر و همکاری نزدیک بین تمامی اعضای تیم و ذینفعان اصلی، از اهمیت ویژه‌ای برای موفقیت فرآیندهای توسعه چابک برخوردار است. در RE@Agile، تیم به همراه ذینفعان اصلی، نیازمندی‌ها را به روشی کاملاً تعاملی استخراج، تجزیه و تحلیل، اصلاح و مستند می‌کند. RE@Agile به پشتیبانی از متخصصان در انتخاب فعالیت‌های مناسب، در زمان مناسب، برای اطمینان از نیازمندی‌های با کیفیت بالا، پیش از پیاده‌سازی آن‌ها می‌پردازد.
- استخراج، تجزیه و تحلیل، ذکر مشخصات (*Specification*) و پالایش موقعیتی (*Situational*) و انتخابی (*Selective*) نیازمندی‌ها

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 (به‌عنوان مثال برای ذینفعان یا تیم توسعه قابل‌درک نیستند) یا مهم هستند (به‌عنوان مثال 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 به راحتی قابل انطباق 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 فرآیندهای چابک اغلب MVP (کمینه محصول پذیرفتنی) است. MVP یک سیستم مجزا و قابل استقرار 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 هدر رفت را در حین توسعه فراهم می‌کند و فرصتی برای بازخورد سریع مشتری به وجود می‌آورد. یکی از مراحل بعدی محصول اغلب MMP (کمینه محصول قابل عرضه) است - محصولی با کمترین مجموعه ویژگی که نیازهای کاربران را برطرف می‌کند و بنابراین دارای ارزش بازار است. RE @ Agile پاسخ دو سوال بسیار مهم را ارائه می‌دهد، سوالاتی که حتی در فرآیندهای توسعه غیر چابک نیز بسیار مهم هستند: "چگونه مدیریت انتشار و فرآیند تعریف محصول را ساده کنیم؟" و "چگونه MVP یا MMP را براساس نیازمندی‌ها تعریف کنیم؟"

درباره مدرک RE@Agile:

آگاهی از سطح پایه CPRE و فرآیندهای توسعه چابک به عنوان دانش پیش نیاز توصیه می‌شود.

گواهینامه RE@Agile Primer برای افراد حرفه‌ای از رشته‌های مرتبط است: مدیران پروژه، تحلیل‌گران کسب و کار، معماران، توسعه دهندگان، آزمون‌گران و همچنین افراد کسب و کاری. 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 به مهندسی نیازمندی‌ها و رویکردهای چابک آشنا است
- نقش و اهمیت مهندسی نیازمندی‌ها را در فرآیندهای چابک و همچنین ارزش چابکی را در مهندسی نیازمندی‌ها درک می‌کند

1 انگیزه و ذهنیت ها (L1)

مدت زمان: 1 ساعت و نیم

اصطلاحات: ارزش ها، بیانیه، روش ها، فعالیت ها، اسپرینت، چابک

اهداف آموزشی

- 1.1.1 هدف آموزشی دانستن انگیزه استفاده از روش های چابک
- 1.2.1 هدف آموزشی دانستن اهداف مهندسی نیازمندی ها مطابق با IREB
- 1.2.2 هدف آموزشی دانستن ارزش های اصلی بیانیه چابک و اصول ناشی از آن
- 1.3.1 هدف آموزشی دانستن تفاوت بین اصل (Principle)، پرکتنیس (Practice) و فعالیت (Activity)
- 1.3.2 هدف آموزشی دانستن تفاوت میان طرز فکر چابک و طرز فکر مهندسی نیازمندی ها
- 1.3.3 هدف آموزشی دانستن هم افزایی طرز فکرها و ارزش ها نسبت به RE @ Agile
- 1.3.4 هدف آموزشی دانستن معنی "مستندسازی" در حوزه چابک (همسو با بیانیه چابک)
- 1.4.1 هدف آموزشی دانستن مزایا، مشکلات و باورهای غلط استفاده از RE @ Agile
- 1.4.2 هدف آموزشی دانستن مثال هایی از باورهای اشتباه
- 1.5.1 هدف آموزشی دانستن این مورد که ارزش های چابک را می توان به کارهای مفهومی تبدیل کرد
- 1.5.2 هدف آموزشی دانستن رویکردهای نمونه ای که امکان چابکی در کارهای مفهومی را فراهم می کنند

1.1 انگیزه استفاده از چابک (L1)

چندین مطالعه ([MeMi2015] را مشاهده کنید) نشان می دهند که تجارت فناوری اطلاعات به طور کلی در حال تغییر اساسی است: فناوری اطلاعات در حال تبدیل شدن به یک محرک اصلی در چندین زمینه تجاری (به عنوان مثال تجارت الکترونیک، رسانه های اجتماعی و غیره) و حوزه های فنی (به عنوان مثال صنعت خودرو یا هواپیمایی) است. در نتیجه، سیستم ها و محصولات موجود در مشاغل مبتنی بر فناوری اطلاعات مجبور هستند برای سازگاری با نیازهای متغیر مشتریان یا بازار، انطباقی مداوم داشته باشند. به محض ایجاد تغییر در بازار، سیستم ها 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 (1.2 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 ([ShYo2006] مشاهده کنید):

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

این تعریف نه از مهندسی نرم افزار بلکه از ورزش نشئت گرفته است، با این حال انگیزه اساسی برای استفاده از روش چابک را منعکس می کند: در صورتی که وضعیت بازار یا پروژه نیاز به تغییرات سریع و کنترل شده داشته باشد، روش های چابک مناسب هستند. یک روش چابک البته چیزی فراتر از توسعه سریع است (1.2 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 روش های چابک و نه چابکی به خودی خود هدف نیستند [Meyer2014]. 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 [MeMi2015].

1.2 طرز فکر و ارزش ها در مهندسی نیازمندی ها و چابک (L1)

طرز فکر و ارزش های RE در تعریف IREB از مهندسی نیازمندی ها بیان شده است (به [Glinz2026] نگاه کنید): رویکرد سیستماتیک و منظم تعیین و مدیریت نیازمندی ها با هدف درک خواسته ها و نیازهای ذینفعان و به حداقل رساندن خطر ارائه سیستمی که این خواسته ها و نیازها را برآورده نمی کند.

در مهندسی نیازمندی ها ما به جای نرم افزار یا محصول، از سیستم استفاده می کنیم. استفاده از اصطلاح سیستم به معنای حذف محصولات، انواع دیگر نرم افزارها، یا حتی موارد دیگر (به عنوان مثال فرآیندهای تجاری یا سخت افزار) نیست. مهندسی نیازمندی ها اصطلاح سیستم را به این دلیل ترجیح می دهد زیرا این اصطلاح بر این واقعیت تأکید دارد که سیستم گروهی از قطعات یا عناصر است که در یک محیط با هم کار می کنند. مهندسی نیازمندی ها، محیط (Environment) را زمینه سیستم (System context) نامگذاری می کند. 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 نرم افزار باشد.

علاوه بر این، سطح IREB FL

[CPREFL2022] مجموعه ای از چهار فعالیت اصلی مهندسی نیازمندی ها را تعریف می کند: استخراج، مستندسازی، اعتبارسنجی / مذاکره و مدیریت نیازمندی ها. این لیست فعالیت ها نشان دهنده مجموعه خاصی از مراحل یا توالی انجام این فعالیت ها نیست. یک ارزش اصلی IREB FL این مورد است که مهندسی نیازمندی ها یک رویکرد فرایندی آگنوستیک (Process-agnostic approach) است: مهندسی نیازمندی ها، پیکره دانشی غنی از دانش متشکل از روش های مختلف و مجموعه ای غنی از تکنیک هایی را که می توان در هر رویکرد توسعه به کار برد فراهم می کند. مهندسی نیازمندی ها، فرایندی را توصیه یا تعیین نمی کند. این سرفصل و راهنمای مطالعه از اصطلاح "روش های چابک" برای اشاره به مجموعه غنی رویکردهایی که در زمینه چابک پدید آمده است، استفاده خواهد کرد (2 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 می شود ([AgileMan2001] را مشاهده کنید):

بیانیه چابک

ما در حال کشف راه های بهتری برای توسعه نرم افزار با انجام آن و کمک به دیگران در انجام آن هستیم.

از این طریق باید به ارزش های زیر دست یابیم:

افراد و تعاملات ارجح بر فرآیندها و ابزارها

نرم افزار کارکننده ارجح بر مستندسازی جامع

مشارکت مشتری در انجام کار ارجح بر مذاکرات بر اساس قرارداد

پاسخ به تغییر ارجح بر تبعیت از برنامه

در حالی که موارد سمت چپ ارزشمند هستند،

ما برای موارد سمت راست ارزش های بیشتری قائل هستیم.

1. بالاترین اولویت ما جلب رضایت مشتری از طریق تحویل زودهنگام و مداوم نرم افزارهای ارزشمند است.
2. استقبال از تغییر نیازمندی ها، حتی در اواخر فرآیند توسعه. فرآیند های چابک، تغییر را در راستای مزیت رقابتی مشتری کنترل میکنند.
3. تحویل مداوم نرم افزار قابل استفاده از دو، سه هفته یک بار تا دو، سه ماه یک بار با ترجیح بر فاصله های زمانی کوتاه تر.
4. ذی نفعان کسب و کار و توسعه دهنده ها باید به صورت روزانه در طول پروژه با هم کار کنند.
5. پروژه ها را بر دوش افراد با انگیزه بنا کنید. فضای لازم را به آنها بدهید و از نیازهای آن ها پشتیبانی کنید و به آنها اعتماد کنید تا کارها را انجام دهند.
6. کارآمدترین و موثرترین روش انتقال اطلاعات به تیم توسعه و تبادل آن در میان اعضای تیم، گفتگوی رو در رو است.
7. نرم افزار قابل استفاده اصلی ترین معیار سنجش پیشرفت است.
8. فرآیندهای چابک، توسعه پایدار را ترویج می دهند. حامیان مالی، توسعه دهندگان و کاربران باید بتوانند سرعت پیشرفت ثابتی را برای مدت نامحدود حفظ کنند.
9. توجه مداوم به برتری فنی و طراحی خوب باعث افزایش چابکی می شود.
10. سادگی--هنر به حداکثر رساندن مقدار کار انجام نشده--ضروری است.
11. بهترین معماری ها، نیازمندی ها و طراحی ها از تیم های خود سازمانده پدید آور می شوند.
12. در فواصل منظم، تیم بر چگونگی مؤثرتر شدن تأمل و تفکر می نماید و سپس رفتار خود را بر اساس بازتاب این تفکر، تنظیم و همسو می کند.

اگر ارزش ها و تفکرات مهندسی نیازمندی ها و چابک را مقایسه کنیم، نمی توان بخشی پیدا کرد که با ارزش ها یا تفکرات دیگری در مغایرت باشد. مهم ترین ارزش توسط مهندسی نیازمندی ها و چابک به اشتراک گذاشته می شود و آن ارزش این است که کاربر نهایی محصول را خوشحال کند، زیرا این راهکار، متناسب با نیاز آن ها است یا بزرگترین درد آن ها را درمان می کند. با این وجود، باید از این موضوع آگاه باشیم که طرز فکر و ارزش های مهندسی نیازمندی ها و چابک تا حدی از هم جدا هستند. نقطه اتصال مهندسی نیازمندی ها و چابک توسط زمینه RE@Agile تعریف شده است که در واحد درسی بعدی توضیح داده خواهد شد.

1.3 اتصال مهندسی نیازمندی ها و چابک در راستای (L1) RE@Agile

قبل از اینکه به جزئیات RE@Agile بپردازیم، باید برخی اصطلاحات را شفاف کنیم. در صنعت و تحقیقات نرم افزار، بیکره دانش گسترده ای در مورد نحوه کار و رفتار هنگام تولید نرم افزار وجود دارد. این دانش در سطوح مختلف انتزاع وجود دارد. در ادامه، ما تمایز اصول، پرکتیس ها و فعالیت ها را به عنوان سه سطح انتزاع برای صحبت در مورد توسعه نرم افزار معرفی خواهیم کرد. [Meyer2014] را مشاهده کنید)

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

اصطلاحات مهم برای تمایز این سه تعریف، تجویزی، انتزاعی و جعلی است. تجویز به این معنی است که بیانیه به جای بیان یک واقعیت یا خاصیت، اقدام را جهت دهی می کند. انتزاع یک اصل را از عمل متمایز می کند. به عنوان مثال، "آزمودن ویژگی نرم افزار، قبل از تحویل" تجویزی و انتزاعی است، در حالی که "ایجاد آزمون واحد (Unit test) برای هر ویژگی نرم افزاری" پرکتیسی مبتنی بر یک اصل داده شده است. یک فعالیت برای این مثال می تواند ایجاد یک تست واحد برای عملکرد جستجو در یک سیستم کتابخانه باشد. قابلیت اثبات نادرستی به این معنا است که شخصی با دانش کافی می تواند با یک اصل مخالف باشد. اصل گفته شده در بالا ("آزمون a...") این معیار را برآورده می کند. ممکن است استدلال شود که آزمون برای ویژگی های مهم ایمنی مناسب نباشد و در عوض، باید آن ها را با استفاده از روش های ریاضی اعتبارسنجی کرد. گزاره ای 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 مورد استفاده یا عدم استفاده از یک اصل کمک می‌کند.

دانستن اصول و رویه‌ها فقط یک قدم ابتدایی محسوب می‌شود: استفاده صحیح از یک روش به تنهایی یک صلاحیت محسوب می‌شود. به عنوان مثال، تعریف مورد‌های کاربرد (Use cases)، یک روش شناخته شده برای اصل "تعیین نیازمندی‌های عملکردی برای ویژگی‌های مهم" است. با این حال، نوشتن مورد کاربرد با کیفیت بالا در نوع خود یک صلاحیت است و دانستن عناصر الگوی مورد استفاده برای این کار کافی نیست.

در اصل، ما اکنون سه سطح صلاحیت را تعریف کرده ایم:

1. دانستن اصول (مربوط به سطح شناختی L1)
2. درک روش‌هایی برای تحقق اصول داده شده (مربوط به سطح شناختی L1)
3. اعمال روش در یک زمینه خاص (توانایی انجام یک فعالیت با کیفیت بالا) (مربوط به سطح شناختی L3)

با مقایسه اصول چابک با اصول مهندسی نیازمندی‌ها (1.2 را مشاهده کنید)، می‌توان 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 می‌کند و افراد و ارتباطات را ارجح بر از فرایندها و ابزارها ارزیابی می‌کند، مهندسی نیازمندی‌ها به نوبه خود، مشغول استخراج و مستندسازی سیستماتیک نیازمندی‌ها به عنوان محصولات کاری است.

اجرای اغراق آمیز هر دو طرز 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 قابل اجرا باشد. تفسیر مشابه کاذب از چابک این است که یک پروژه توسعه می‌تواند بدون هیچ کار مقدماتی شروع شود و تنها با ارائه نرم افزاری 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 ذی نفعان 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 نرم افزار در سطح کیفی کاملاً مشخص هستند. روش‌های چابک می‌توانند نرم افزارهای قابل استفاده را به روشی کارآمد و سریع (زمان چرخه کاهش یافته) (Reduced cycle time) تحویل دهند. مهندسی نیازمندی‌ها به فراهم کردن روش‌های مناسب برای درک خواسته‌ها و نیازهای ذی نفعان برای توسعه نرم افزاری می‌پردازد.

مهندسی نیازمندی‌ها به تسهیل موارد زیر کمک می‌کند:

- درک مناسب از خواسته‌ها و نیازهای کاربران برای تولید نرم افزار ارزشمند (اصل اول چابک)
- ابزارهای مناسب برای تشخیص تغییرات در بازار برای مزیت رقابتی ذی نفعان (اصل دوم چابک)
- ابزارها و تکنیک‌های مناسب برای تقویت همکاری موثر میان ذی نفعان و توسعه دهندگان (اصل چهارم چابک)
- ابزارها و تکنیک‌های مناسب برای حمایت از ارتباط کلامی (اصل ششم چابک)
- درک مناسب از خواسته‌ها و نیازهای ذینفعان برای به حداقل رساندن توسعه نرم افزارهای 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 فرآیند است.

بیانیه توسعه نرم‌افزار چابک، بر ارزش اینکریمنتی از نرم‌افزار قابل استفاده (یا محصول در حال کار) در مقایسه با مستندات جامع تأکید دارد. یک تفسیر اغراق آمیز منجر به ایجاد این تصور غلط شده است که روشهای چابک مستندات را به کلی کنار گذاشته اند. 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. در گذشته، یک مسئله درک شده این مورد بود که بسیاری از پروژه‌ها، اسنادی را بدون هدف یا ارزش افزوده مشخص ایجاد می‌کردند - طبق اصول، از این نوع اسناد جلوگیری می‌شود.

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

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

1.4.1 مزایای RE@Agile:

در یک تیم، صلاحیت‌های مهندسی نیازمندی‌ها و توسعه می‌توانند تحویل‌دانی‌ها (Handovers) را کاهش دهند: پرکنش تیم‌های چندعملکردی 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 به صورت 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 تکرار شونده به ایجاد محصولات کاری می‌پردازد (به عنوان مثال یک مدل فرآیند کسب و کار، یک اپیک، یک داستان کاربر، یک مورد کاربرد، یک نمونه اولیه رابط کاربر، یک توصیف فرآیند 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 بهبود یافته 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. علاوه بر این، از پرکنش خوب تعریف آماده (Definition of Ready)، به عنوان 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 مورد نظر ارائه می‌دهد. بدین ترتیب، مهندسی نیازمندی‌ها درک عمیق‌تری از نیازمندی‌های مورد نیاز 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 نمی‌شود. در عوض هدف از انجام این فعالیت‌ها تمرکز بر درک کاملی از محصول در سطح مشخصی از انتزاع است (به عنوان مثال درک و تعریف موردهای کاربرد ضروری (Essential use cases) یا اپیک‌ها و داستان‌های کاربر). به عنوان مثال، یک فرآیند پیچیده سطح بالایی کسب و کار با استفاده از روش‌های کاملاً تثبیت شده مهندسی نیازمندی‌ها، می‌تواند برای ایجاد یک بک‌لاگ اولیه [CPREALAGILE2022]، به اپیک‌ها، فیچرها و داستان‌های کاربر تجزیه شود.

در دنیای توسعه (عمدتاً توسعه نرم افزار)، برخی تصورات غلط و مشکلات در مورد مهندسی نیازمندی ها وجود دارد که باید مورد بحث قرار گیرند:

تصور غلط - مهندسی نیازمندی ها تنها یک تجزیه و تحلیل مقدماتی (Upfront) است: اغلب اوقات، مهندسی نیازمندی ها تنها به عنوان یک فعالیت مقدماتی در نظر گرفته می شود. به عنوان یک دیسیپلین، مهندسی نیازمندی ها در واقع مستقل از روند کار است و از قبل تمامی کارها را اجرا نمی کند. در عوض، فعالیت های مهندسی نیازمندی ها می تواند به همان روشی که سایر فعالیت ها (به عنوان مثال کدگذاری یا آزمون) انجام می شوند، در یک روش چابک شکل بگیرد. مهندسی نیازمندی ها یک فعالیت تعبیه شده در هر تکرار است.

تصور غلط - کار مقدماتی (Upfront) شیطانی است: آماده شدن برای توسعه تکرار شونده، یک قسمت اساسی در هر کار توسعه ضمنی (Non-trivial development) است. تفکر مقدماتی (Upfront thinking) به خودی خود بر چرخه زندگی نرم افزاری خاص، یا یک مرحله یا مستند تجزیه و تحلیل طولانی، دلالت نمی کند: نحوه و زمان انجام تفکر مقدماتی توسط زمینه پروژه (Project context) به صورت دقیق تعیین می شود. متخصصان نباید ادبیات چابک را اینگونه تفسیر کنند که تفکر مقدماتی به خودی خود چیز بدی است. بیانیه و اصول ارائه شده در بالا از چنین درکی پشتیبانی نمی کنند. شیوه های چابکی که ارزش تفکر مقدماتی را منعکس می کنند شامل نگاشت داستان کاربر (Patt2014) را مشاهده کنید)، نمونه سازی اولیه (Martin1991) را مشاهده کنید)، و توسعه مبتنی بر آزمون (Beck2003) را مشاهده کنید) می شوند.

تصور غلط - مهندسی نیازمندی ها برابر است با مستندات: مهندسی نیازمندی ها اغلب فقط با اسنادی که تولید می کند مرتبط است. با این حال، اسناد و مدارک یکی از نتایج بالقوه فعالیتی است که باعث ایجاد دانش می شود. مهندسی نیازمندی های 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 3.2 عمل می کند: انطباق قانونی، حفظ اطلاعات ارزشمند، تسهیل ارتباطات و پشتیبانی از فرایندهای فکری.

تصور غلط - داستان های کاربر کافی هستند: داستان های کاربر یکی از روش های محبوب برای استخراج نیازمندی های ذی نفعان هستند. با این حال، آنها برای شروع ارتباطات و نه نمایش مشخصات کامل، در نظر گرفته شده اند. مسیر یک نیاز نامشخص به سمت یک نیازمندی کامل در عمل "3C - کارت (Card)، ارتباط (Communication)، تأیید (Confirmation)" خلاصه شده است. از طریق ترکیبی از داستان های کاربر با سایر تکنیک های مهندسی نیازمندی های تأیید شده مانند نمودارهای زمینه (Context diagrams)، نمونه سازی اولیه (Prototyping)، موردهای کاربرد (Use cases) و سفرهای کاربر (User journeys)، می توان به تصویر کامل تری از نیازمندی ها دست یافت.

تصور غلط - مستندسازی بی ارزش است، فقط کد دارای ارزش ماندگار است: گرچه کاملاً ممکن است در برخی از پروژه های دارای فرآیند سنگین، مشخصات بیش از حد به عنوان یک مسئله مطرح باشد، اما نتیجه گیری اینکه همه مستندسازی ها بی ارزش هستند درست نیست. مستندات نیازمندی ها، درست مانند طراحی، آزمایش یا مستندات عملیاتی، در زمینه های خاصی جایگاه خود را دارد. مستندات همگی به یک اندازه معتبر و محصولات کاری ضروری هستند که از فرآیند توسعه برای هر محصول نرم افزاری پایدار حاصل می شود.

تصور غلط - نرم افزار قابل استفاده تنها راه تأیید نیازها است: بیانیه چابک برای "نرم افزار کارکننده ارجح بر مستندسازی جامع" ارزش می نهد. وقتی صحبت از اعتبارسنجی نیازمندی ها می شود، یک نتیجه گیری غلط از این عبارت این است که اعتبارسنجی نیازمندی ها بر اساس نرم افزار قابل استفاده بر اعتبارسنجی مستندات مرتبط با نیازمندی ها ارجحیت دارد. نرم افزار به عنوان ابزاری برای تأیید نیازمندی ها در صورتی قابل قبول است که هزینه ها و ریسک های مربوط به چنین رویکرد اعتبارسنجی در مقایسه با نتیجه آن، قابل قبول باشد. اگر هزینه ها و یا ریسک ها زیاد باشد، مهندسی نیازمندی ها به فراهم کردن چندین ابزار می پردازد که امکان بازخورد سریع و تأیید نیازمندی ها، قبل از نوشتن یک خط کد را فراهم می کند، به عنوان مثال: موکاپ های رابط کاربر یا استوری بوردها.

دام- نیازمندی ها را به عنوان نوعی اطلاعات یک شکل در نظر بگیرید: یک اشتباه معمول مرتبط با مهندسی نیازمندی ها در تمام زمینه ها / روش های پیاده سازی، در نظر گرفتن "نیازمندی" به عنوان نوعی یک شکل از اطلاعات است. بر اساس این برداشت اشتباه، اغلب مستندات نیازمندی ها به عنوان اتلاف وقت در نظر گرفته می شوند، زیرا نیازمندی ها به سرعت تغییر می کنند و به محض نوشتن بی اعتبار می شوند. نیازمندی ها یک نوع اطلاعات یکنواخت نیستند. نیازمندی ها را می توان در سطوح مختلف جزئیات، انتزاع و فرمت ها بیان کرد. به عنوان مثال، چشم انداز یا اهداف سیستم برای سیستم، نیازمندی هایی در سطح بالایی از انتزاع و به طور معمول با یک عمر طولانی هستند. نمونه اولیه قابل اجرا ابزاری برای تأیید مجموعه ای از نیازمندی ها و یا ایجاد الزامات جدید است.

دام- از دست دادن تصویر بزرگ: روش های چابک اغلب بد فهمیده می شوند و به گونه ای اجرا می شوند که فقط به موضوعاتی بپردازند که همین الان در مقابل تیم قرار دارند. از دید یک توسعه دهنده، این ممکن است یک اصل مفید تلقی شود، زیرا توسعه دهنده می تواند انرژی ذهنی خود را بر روی کار مورد نظر متمرکز کند و موضوعات طولانی مدت حواس او را پرت نمی کند. با این حال، اگر همه فقط روی کارهای حاضر تمرکز کنند، تصویر بزرگ و چشم انداز طولانی مدت از بین می رود. روش های پایدار چابک در جلسات اختصاصی به چشم اندازهای بلند مدت و میان مدت می پردازند (به عنوان مثال جلسات پالایش، جلسات طراحی نقشه راه یا ورکشاپ های طراحی چشم انداز). یک دام مرتبط، ادامه دادن راهکار بدون تعریف کامل مشکل کسب و کار است (اغلب به عنوان نیاز نام برده می شود).

دام - ارائه بیش از اندازه اطلاعات به ذی نفعان: محصولات کاری مهندسی نیازمندی ها می توانند شامل تراکم بالایی از اطلاعات باشند و در تیم چابک، بسیار سریع ایجاد می شوند. این رویکرد غالباً در پروژه های "پیشرفته" یا با فناوری بالا مورد استفاده قرار می گیرد، جایی که به سختی می توان متخصصان موضوعی (Subject matter experts) را پیدا کرد. با این حال، انجام یک تکرار کامل برای کار بر روی محصولات کاری مهندسی نیازمندی ها، بار بازبینی زیادی را برای ذی نفعان ایجاد می کند و آن ها را ملزم به خلاصه کردن مشخصات طولانی می کند. نتایج بهتر ممکن است با ترکیب خوبی از مشخصات و توسعه (نمونه سازی اولیه) حاصل شود.

دام - تشریح هر مبحث به صورت تدریجی و تکرار شونده: نباید هر مبحث نیازمندی های سیستم با جزئیات کامل (Fine-grained) و تکرار شونده (Iterative) توسعه یابد. مباحثی با پیچیدگی اضافی (Additive complexity) (Meyer2014] 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 مطلقاً کامل (Meyer2014] 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 ایده های نوآورانه و یا تحول آفرین را تشویق نکند، زیرا یک محصول کاری خاص (به عنوان مثال نرم افزار یا ویژگی/ عملکرد نرم افزار) پس از تعریف شدن تنها در همان قسمت از برنامه (Locally) بهبود می یابد (به عنوان مثال رفع خطاها یا افزودن عناصر گذشته). اگر چه بیانیه چابک صریحاً از تغییر استقبال می کند، اما فرایندهای تدریجی چابک به طور معمول از نوآوری مداوم برای محصولات و خدمات پشتیبانی می کنند. نوآوری های بنیادی یا تحول آفرین از طریق در نظر گرفتن ایده های متعدد و ترکیب مجدد ایده های موجود پدیدار می شوند [LiOg2011]. 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 10 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 در 1.5 استفاده کرد.

مهم ترین و اصلی ترین دام- چابک و تغییر فرهنگ با هم همخوانی ندارند: ارزش های چابک باعث ایجاد تغییر در نحوه عملکرد سازمان ها، از بین رفتن مالکیت برخی از محصولات قابل تحویل و مسئولیت جمعی می شوند. چابک، بازنگری مستمر در رفتار تیم

را برای بهبود روش کار خود ارتقا می دهد: تغییر مداوم تیم و سرانجام کل سازمان اجتناب ناپذیر است. چنین تغییرات فرهنگی (سازمانی) هم به زمان و هم به افراد واجد شرایط نیاز دارد تا تیم ها را در مسیر جدید هدایت کنند.

1.5 RE@Agile و کار مفهومی (L1)

توسعه چابک در جهان مهندسی نرم افزار برای رفع چالش هایی که از دنیای خارج از مهندسی نرم افزار نمایان شدند، به وجود آمد (1.1 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 (1.2 را مشاهده کنید). در ادامه، ما سه روش را به عنوان نمونه ای برای چابکی در کارهای مفهومی ارائه خواهیم داد.

تفکر دیزاین ([LiOg2011] ، [DsCh2015]) 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 می دهد تا به درک افراد مرتبط با مسئله قابل حل بپردازد.
- تعریف: در این مرحله، تیم با استفاده از بیان مجدد مسئله، درک مشترکی درباره جزئیات مسئله ای که باید حل شود، به دست آورد.
- ایده پردازی: در این فاز، تیم بر ایده پردازی تمرکز دارد. هدف در اینجا توسعه ایده نیست. در عوض تیم تا آنجا که ممکن است تعداد بسیاری از ایده ها را توسعه می دهد. در پایان این مرحله، تیم امیدوارکننده ترین ایده ها را برای نمونه سازی اولیه انتخاب می کند.
- نمونه سازی اولیه: در این مرحله، تیم نمونه های اولیه بسیار ساده ای (نه لزوماً نرم افزاری!) را از ایده های توسعه یافته ایجاد می کند. اصل در اینجا این است که نمونه اولیه باید تا حد ممکن واقع بینانه و ارزان باشد.
- تست: در این مرحله، تیم نمونه اولیه را با مشتریان واقعی تست می کند تا از ایده های آن ها بازخورد بگیرد. یک اصل بنیادی برای مرحله آزمون "نشان دهید، نگوئید" (show, not tell) است، به این معنی که نمونه اولیه باید کاملاً گویا باشد تا کاربر بتواند بازخوردی واقعی ارائه دهد.

مراحل تفکر دیزاین مقیاس پذیر هستند و می توانند در پروژه هایی از چند روز تا چند هفته انجام شوند. به علاوه، مراحل، توالی دقیقی را تعریف نمی کنند. هر زمان که لازم باشد، تیم می تواند 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 فرآیند تفکر دیزاین، مجموعه ای از نمونه سازی اولیه است که نشان دهنده راهکارهای معتبر و ابتکاری برای مسئله در ابتدا تعریف شده است. از این رو، می توان از تفکر دیزاین برای توسعه ایده هایی برای نرم افزارهای با ارزش استفاده کرد و در نتیجه از اصل اول چابک پشتیبانی می کند. همانطور که در بالا گفته شد، تیم چند عملکردی و محیط کار، از عناصر اصلی فرآیند هستند که با اصل پنجم چابک مطابقت دارد. هدف اصلی فاز نمونه اولیه تولید نمونه های ارزان قیمت و سبک است که با اصل سادگی چابک مطابقت دارد.

دیزاین اسپرینت [KnZK2016] یک فرآیند پنج روزه برای توسعه ایده ها بر اساس اصول طراحی، نمونه سازی اولیه و آزمون با مشتریان نهایی است. از منظر مهندسی نیازمندی ها، دیزاین اسپرینت ترکیبی از تکنیک های استخراج و اعتبارسنجی است. پایه این روش نحوه کار زمان-ثابت (Time-boxed) است. در این روش، هر روز به یکی از فعالیت های زیر اختصاص داده شده است: به اشتراک گذاری دانش تیم، ارائه ایده ها، تصمیم گیری در مورد اینکه کدام ایده ها را در نمونه اولیه ایجاد کنید، ساخت ایده های انتخابی

در قالب نمونه اولیه و در آخر، آزمودن ایده ها با مشتریان واقعی. در اینجا، تفاوت مهم در مقایسه با توسعه چابک این است که نمونه اولیه نیازی به نرم افزار ندارد. دانستن این نکته مهم است که اصطلاح "اسپرینت" به اسپرینت اسکرام اشاره نمی کند.

استارتاپ ناب [Ries2011] 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" [Ries2011]. مفهوم مهم دیگر استارتاپ ناب، چرخش (Pivot) است، "یک اصلاح ساختار یافته برای آزمون فرضیه اساسی جدید در مورد محصول، استراتژی و موتور رشد" [Ries2011]. 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 می دهد که چابک واقعاً چیزی بیش از اسکرام است. نمونه هایی از تفکر دیزاین، دیزاین اسپرینت و استارتاپ ناب نشان می دهد که رویکردهایی برای کار مفهومی در RE@Agile وجود دارد که تفکر چابک را دارند و بنابراین با سازمانی که می خواهند نرم افزار را به روش چابک توسعه دهند سازگاری کامل دارند. این رویکردها و سایر رویکردها نباید به عنوان شکل جدیدی از رویکرد آشنایی یا تفکر مقدماتی (Upfront thinking)، در نظر گرفته شوند. آن ها نه تنها این قابلیت را دارند، بلکه باید در چارچوب توسعه چابک (به عنوان مثال در یک یا چند تکرار) برای طراحی جنبه های اختصاصی یک سیستم استفاده شوند. از سوی دیگر، می توان آن ها را به عنوان فعالیت هایی که مقدم بر توسعه چابک هستند (به عنوان مثال یک مرحله سریع تفکر مقدماتی (Upfront thinking))، استفاده کرد. بدین ترتیب، این رویکردها به غلبه بر پتانسیل محدود نوآوری در توسعه چابک کمک می کنند.

2 مبانی (L2) RE@Agile

مدت زمان: 2 ساعت و نیم

اصطلاحات: مالک محصول، بک لاگ محصول، بک لاگ اسپرینت، اپیک ها، داستان های کاربر، نقشه های داستان

اهداف آموزشی

- 2.1.1 هدف آموزشی دانستن نمونه هایی از روش های چابک
- 2.2.1 هدف آموزشی دانستن اسکرام به عنوان یک مثال: نقش ها ، فرآیندها ، مصنوعات و ارتباط آن ها با مهندسی نیازمندی ها
- 2.2.2 هدف آموزشی دانستن مسئولیت های یک مالک محصول اسکرام
- 2.2.3 هدف آموزشی دانستن مفهوم بک لاگ محصول و تفاوت های آن با مستند مشخصات نیازمندی های محصول.
- 2.3.1 هدف آموزشی درک تفاوت بین مهندس نیازمندی های کلاسیک و مالک محصول اسکرام
- 2.4.1 هدف آموزشی درک دلایل خوب برای این مورد که چرا مهندسی نیازمندی ها باید بخشی از یک روند مداوم باشد
- 2.5.1 هدف آموزشی دانستن توسعه مبتنی بر ارزش (به عنوان مثال اولویت بندی نیازمندی ها)
- 2.5.2 هدف آموزشی دانستن اینکه ارزش، مدیریت ریسک و فرصت است
- 2.5.3 هدف آموزشی دانستن نمونه هایی از ارزش در سازمان های انتفاعی و غیر انتفاعی
- 2.6.1 هدف آموزشی دانستن نحوه ساده سازی فرآیند تعریف محصول و چگونگی تعریف حداقل محصول پذیرفتنی
- 2.7.1 هدف آموزشی دانستن ارزش فرآیندهای مداوم و منحنی یادگیری آنها

2.1 رویکردهای چابک (مرور کلی) (L1)

متدهای بسیاری برای به اشتراک گذاری ارزش های بنیادین چابک توسعه یافته است. این واحد آموزشی برای دیدن تنوع رویکردها، نمای کلی برخی از آن ها را به شما ارائه می دهد. این لیست جامعیت نداشته و روش ها را از نظر مهندسی نیازمندی ها مورد بحث قرار می دهد.

کریستال (Crystal)، خانواده ای از متدها است که توسط آلیستر کابرن توسعه داده شده است [Cock1998]. وی پیشنهاد می کند که هر پروژه به مدل فرآیند متناسب با خود نیاز دارد. بسته به اندازه، پیچیدگی و اهمیت آن، آلیستر نقش ها، فعالیت ها و محصولات کاری کافی را پیشنهاد می کند. کریستال کلیر (Crystal Clear) - روشی برای پروژه های کوچک و با اهمیت کمتر - بسیار شبیه XP است. کریستال نارنجی (Crystal Orange) و کریستال قرمز (Crystal Red) کمی فرم گرایی بیشتری برای کنار آمدن با پروژه های بزرگ تر ایجاد می کنند. از دیدگاه نیازمندی ها، (در کنار تمام موارد دیگر) آلیستر پیشنهاد می کند تا با مدل های مورد کاربرد (Use cases) و موکاپ های (Mock-ups) با مستندسازی و رسمیت کم (Low-ceremony) کار شود.

توسعه ناب [Popp2003] (Lean Development) و کانبان (Kanban) بر اساس اصولی هستند که برای اولین بار در دهه 1940 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 7 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 [Scrum2020] چارچوبی برای توسعه و نگهداری راهکارهای ارزشمند به عنوان یک محصول در محیط‌های پیچیده است. این چارچوب بر توسعه تکرارشونده و تدریجی مبتنی بر تجربه‌گرایی و تفکر ناب متمرکز است. این چارچوب تنها سه نقش اساسی (در اسکرام به عنوان مسئولیت نام برده شده اند [Scrum2020]) را مشخص می‌کند: یک مالک محصول (برای مدیریت بک لاگ محصول، به عنوان مثال تعریف چشم انداز و تمام نیازمندی‌های مرتبط به یک محصول)، توسعه دهندگان برای توسعه این نیازمندی‌ها در اسپرینت‌های کوتاه (اصطلاحی برای Iteration‌ها در اسکرام) و یک اسکرام مستر برای راهنمایی و تسهیل‌گری پیاده‌سازی 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) [Beck2003] 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 (برنامه نویسی اکستریم) [Beck2004]: XP بر برقراری ارتباط مستقیم بین مشتری و یک برنامه نویس ("مشتری در محل" که درست کنار برنامه نویس نشسته است، دائماً در مورد نیازمندی‌ها بحث می‌کند و به ارائه بازخوردهایی در قالب ویژگی‌های قابل پیاده‌سازی می‌پردازد) تأکید می‌کند.

2.2 اسکرام (به همراه پرکتیس‌های خوب) به عنوان مثال (L1):

اسکرام محبوب‌ترین و پذیرفته‌شده‌ترین چارچوب چابک است. اسکرام یک چارچوب سبک وزن است که به افراد، تیم‌ها و سازمان‌ها کمک می‌کند تا از طریق راه حل‌های تطبیقی برای مشکلات پیچیده، خلق ارزش کنند. راهنمای اسکرام [Scrum2020]، به ارائه تعریفی از اسکرام و مولفه‌های اساسی آن می‌پردازد.

در این متن، بر اساس تعریف [Scrum2020]، موارد زیر وجود دارند:

- سه مسئولیت (اسکرام مستر، مالک محصول و توسعه دهنده). این سه در مجموع به عنوان تیم اسکرام نیز شناخته می‌شوند.
- پنج رویداد (اسپرینت، برنامه ریزی اسپرینت، اسکرام روزانه، بررسی اسپرینت، بازاندیشی اسپرینت).
- سه مصنوع 1 و تعهدات مرتبط آن (بک لاگ محصول، بک لاگ اسپرینت، اینکریمنت)

اسکرام هیچ روش مهندسی را توصیه نمی‌کند.

اسکرام پیشنهاد می‌کند محصولات را به صورت تکرارشونده و تدریجی در مجموعه‌ای از اسپرینت‌ها شامل تکرار (Iteration) در بازه زمان-ثابت یک ماهه یا کمتر توسعه دهید. هر اسپرینت منجر به یک اینکریمنت قابل استفاده می‌شود که یک پله‌ی رو به جلوی 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 صرف نظر کند.

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 پذیر می 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 یک لاگ محصول (لیستی مرتب و پویا از همه مواردی که ممکن است در محصول مورد نیاز باشد) بر گرفته شده است. یک 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)، یک 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 یک لاگ محصول بیشتر باشد و زمان اجرای آن نزدیکتر باشد (مثلاً در یکی از دو اسپرینت بعدی)، توصیف آن عنصر از یک لاگ محصول دقیقتر است. مالک محصول در قبال یک لاگ محصول که در ساختار فعلی آن توسط هدف واقعی محصول هدایت می شود، مسئول است. برای جزئیات بیشتر به 3.1.1، مستند مشخصات (Specification Documents) در مقابل یک لاگ محصول (Product Backlog) مراجعه کنید.

برنامه ریزی اسپرینت که برای انتخاب عناصری که در اسپرینت بعدی اجرا خواهند شد انجام می شود، به نتایج زیر منتهی می شود: هدف اسپرینت (چرایی) و یک لاگ اسپرینت (چیستی، برای مثال انتخاب آیتم ها و تجزیه آنها برای اسپرینت) و برنامه اسپرینت (چگونگی دستیابی).

متعاقباً، تیم اسکرام شروع به پیاده سازی اینکریمنت محصول می کند. توسعه دهندگان مسئول تبدیل موارد انتخاب شده به نتایج ملموس هستند. آن ها با مالک محصول برای اصلاح موارد یک لاگ محصول و به اشتراک گذاشتن بازخورد درباره پیاده سازی فعلی، همکاری می کنند. در جهت ارزیابی پیشرفت و به روز رسانی یک لاگ اسپرینت، توسعه دهندگان هر روز در طول جلسه اسکرام روزانه همگام می شوند.

کار توسعه دهندگان با توجه به "تعریف انجام شده" هدایت می شود: تعریفی از آنچه برای کامل شدن اینکریمنت باید به دست بیاید. "تعریف انجام شده" به ذینفعان کمک می کند تا بدون شک پیشرفت محصول را درک کنند.

وقتی زمان-ثابت اسپرینت به اتمام می رسد (بین 1 هفته تا 1 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. یک لاگ محصول به همین ترتیب به روز می شود.

اسپرینت با رویداد بازاندیشی اسپرینت به پایان می رسد که طی آن تیم اسکرام این موارد را بررسی می کند که چگونه می تواند کارآمدتر شود، چه چیزی خوب کار می کند و چه چیزی می تواند بهبود یابد. بلافاصله پس از پایان بازاندیشی اسپرینت، اسپرینت فعلی جمع بندی شده و اسپرینت بعدی شروع می شود.

پرکتیس های خوب (Good practices):

حتی اگر اسکرام با تکنیک های مهندسی نیازمندی ها همراه نباشد، جامعه چابک به ارائه برخی اقدامات خوب می پردازد که مناسب اسکرام هستند.

راهنمای اسکرام از اصطلاح آیتم های یک لاگ محصول برای موارد ذکر شده در یک لاگ محصول استفاده می کند. این یک اصطلاح عمومی برای شناسایی هر نوع اطلاعات در مورد محصول در حال توسعه است، اما بسیاری از موارد موجود در محصول در واقع نیازمندی ها هستند یا می توانند برای تبدیل شدن به نیازمندی مورد استفاده قرار گیرند.

جامعه چابک موارد زیر را پیشنهاد می کند:

- تجزیه نیازمندی ها به: اپیک، ویژگی های اختیاری و داستان های کاربر (سطح دانه بندی)، [CPREALAGILE2022] را مشاهده کنید.
- تمایز بین نیازمندی های عملکردی (توانایی)، نیازمندی های کیفی (رفتار) و قیدها (محیط)
- گروه بندی نیازمندی ها با استفاده از تم ها (Themes)

یک پرکنتیس خوب (اضافی) برای توصیف مشخصات بک لاگ محصول ایجاد شده است: بک لاگ باید "DEEP" (جزئیات مناسب (Detailed)، تخمین زده (Estimated)، در حال تکامل (Emergent)، اولویت بندی شده (Prioritized)) باشد [Cohn2004]. در اسپرینت، آراستن (Grooming) و پالایش (Refinement) بک لاگ که بخشی اساسی در فرآیند توسعه چابک است، وجود دارند.

با استفاده از "تعریف انجام شده" (DoD)، تیم درک مشترکی راجع به کامل شدن یک آیتم از بک لاگ محصول و آماده شدن آن برای تبدیل به بخشی از اینکریمنت قابل استفاده را توسعه می دهد [Scrum2020].

یک پرکنتیس خوب دیگر، استفاده از قاعده INVEST است [Wake2003]. این کلمه اختصاری، معیارهای زیر را پوشش می دهد:

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

مهندسی نیازمندی ها معیارهای مربوطه را برای نیازمندی های خوب ارائه می دهد (

[CPREFL2022] را مشاهده کنید):

- توافق شده (agreed)
- شفاف (unambiguous)
- ضروری (necessary)
- همخوان (Consistent)
- قابل تایید (Verifiable)
- امکان پذیر (feasible)
- قابل ردیابی (traceable)
- کامل (complete)
- قابل درک (understandable)

2.3 تفاوت ها و اشتراک های میان مهندسی نیازمندی ها و مالکان محصول (L2)

پس از بحث در مورد اصول چارچوب اسکرام، بیایید نقش مهندس نیازمندی های 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 قابل درک و اعمال نیست.

[CPREFL2022]:

- استخراج نیازمندی ها
- مستندسازی نیازمندی ها
- اعتبارسنجی و مذاکره نیازمندی ها
- مدیریت نیازمندی ها

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

- اطمینان از اینکه تیم ارزش ثابت (تجاری) را ارائه می کند: این بدان معناست که مالک محصول باید بینش بلندمدت محصول (هدف محصول، چشم انداز محصول) را با نیازهای کوتاه مدت متعادل کند، باید آیتم های بک لاگ محصول را طبق معیارهای تعریف شده اولویت بندی کند و نتایج را در پایان هر تکرار (اسپرینت) به همراه ذینفعان مورد بازرسی قرار دهد.
- مدیریت تمامی ذی نفعان: مالک محصول در مورد ارسال نیازمندی های همخوان به تیم پاسخگو است. او باید نیازمندی ها را از تمامی ذینفعان جمع آوری کند و مطمئن شود که مغایرتی با یکدیگر ندارند. هر گونه اختلاف بین ذینفعان باید حل و فصل شود تا توسعه دهندگان از چنین اختلافاتی رهایی پیدا کنند.
- اقلامی را که دارای بیشترین ارزش (تجاری) هستند، به طور مستمر به تیم ارائه دهید: دانه بندی این نیازها باید به اندازه کافی کوچک باشد که در یک تکرار (Sprint) قرار گیرند. برای هر گونه سوال پس از جلسه برنامه ریزی اسپرینت، مالک محصول باید در دسترس باشد تا به سرعت شفاف سازی کند.

با مقایسه این دو نقش، معلوم می شود که هر دو، مهندسی نیازمندی و مالکان محصول، باید وظایف اصلی استخراج، مستند سازی، اعتبارسنجی و مدیریت نیازمندی ها را (همراه با سایر ذی نفعان) انجام دهند. با این حال، معمولاً نمادها (Notations) و ابزارهای استفاده شده در محیط های چابک کمتر رسمی هستند:

- برای مثال، کارت های داستان (Story cards) به جای مستندات نیازمندی ها
- مکالمه بیشتر و نوشتن کمتر
- تأکید بیشتر بر وضعیت فعلی نیازمندی ها، تأکید کمتر بر نسخه بندی (Versioning) و تاریخچه (History)

نقش مالک محصول، درحالی که همچنان مسئولیت کلی در مورد نیازمندی های با کیفیت بالا را حفظ می کند، نسبت به مهندس نیازمندی های سنتی گسترده تر است، زیرا او مسئول موفقیت محصول به عنوان یک کل است و به طور مداوم به جمع آوری نظرات از کسب و کار می پردازد و به همین ترتیب، اقدام به به روزرسانی و اولویت بندی بک لاگ می کند.

با توجه به فعالیت های مهندسی نیازمندی ها، می توان به طور خلاصه بیان کرد که مالک محصول مسئولیت اجرای آنها را بر عهده دارد. او می تواند توسط افراد با تجربه در مهندسی نیازمندی ها پشتیبانی شود یا خودش این وظایف را انجام دهد. با این حال، مسئولیت نتیجه مهندسی نیازمندی ها و همچنین مسئولیت بک لاگ محصول بر عهده مالک محصول است.

2.4 مهندسی نیازمندی ها به عنوان یک فرآیند مداوم (L2)

بنابراین بیش از اینکه مهندسی نیازمندی ها یک مرحله مشخص در طول توسعه در چابک باشد، یک فعالیت تکرار شونده و مداوم است. هدف این نیست که قبل از شروع طراحی و اجرا، همه نیازمندی ها استخراج و تحلیل شوند. نیازمندی ها و محصولات هر دو به صورت تکرار شونده و تدریجی ایجاد می شوند.

بنابراین تا زمانی که خود محصول ادامه داشته باشد، مهندسی نیازمندی ها به عنوان یک فعالیت مداوم در نظر گرفته می شود. با این وجود، این فرآیند نتایج میانی کاملاً مشخصی را به همراه دارد: نیازمندی های پیش بینی شده که بیشترین ارزش (تجاری) را مبتنی بر معیارهای تعریف شده نوید می دهند، باید "آماده برای اجرا" (Ready for implementation) باشند (به معنای "تعریف آماده" (Definition of ready) که در بالا توضیح داده شد). سایر نیازمندی هایی که فوریت کمتری دارند، تنها پس از تکمیل موارد ضروری اصلاح می شوند.

"فرآیند مداوم" (Continuous process) مانع از اجرای برخی فعالیت های مهم پیشین نمی شود. حتی اگر نیازمندی ها بر اساس "نیاز به دانستن" مشخص شوند، برخی از جنبه های این نیازمندی ها وجود دارند که باید در اوایل چرخه زندگی مورد توجه قرار گیرند. به عنوان مثال می توان به تعریف چشم اندازها یا اهداف، شناخت ذینفعان و تعیین دامنه محصول اشاره کرد. شروع اجرا بدون انجام چنین فعالیت هایی میزان خطر را بطور قابل توجهی بالا می برد.

2.5 توسعه مبتنی بر ارزش (L1)

متدهای چابک تلاش می کنند تا به طور مداوم، ارزش (تجاری) را به کاربر نهایی برسانند. اغلب ارزش می تواند مستقیماً با اصطلاحات مالی، افزایش سهم بازار یا رضایت مشتری بیان شود. این روش اغلب در سازمان های انتفاعی استفاده می شود، اما نحوه تعریف ارزش برای سازمان های غیر انتفاعی کمتر مشخص است. در اینجا، معیارهایی مانند میزان استفاده یا شاخص شادی (Happiness index) در رابطه با یک محصول (به عنوان مثال نرخ کلیک در وب سایت ها یا کمک های مالی یک سازمان غیر انتفاعی) ممکن است بیشتر مرتبط باشند.

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

همان طور که در بخش بعدی توضیح داده شده است، به منظور تعیین اینکه کدامیک از نیازمندی ها به ارزش مطلوب منجر می شوند، متدهای چابک اغلب برای رسیدن به کمینه محصولات پذیرفتنی (MVPs) یا کمینه محصولات قابل عرضه (MMPs) تلاش می کنند.

2.6 سادگی به عنوان مفهومی ضروری (L1)

در یک دنیای پیچیده، سادگی راهی برای پذیرش پیچیدگی با فرآیند زیر است:

- ایجاد یک پاسخ ساده و ناقص بالقوه (Potentially incomplete) به یک مسئله، در نتیجه ایجاد یک مقدار کوچک از ارزش،
- به دست آوردن توانایی یادگیری بیشتر در مورد زمینه (Context)، بر اساس تجربه دنیای واقعی،
- تطبیق و تکرار در ایجاد و ارائه ارزش با سرعت پایدار،
- صرفه جویی در منابع از یک ایده غیر سودآور، 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 می شود که سادگی به معنای "کیفیت پایین" نیست بلکه به معنای "حداقل دامنه" (Minimal scope) یا "حداقل خدمات" (Minimal service) است - اما همیشه با کیفیت بالا. کیفیت قابل مذاکره نیست.

بیانیه چابک سادگی را به عنوان «هنر به حداکثر رساندن میزان کار انجام نشده» توصیف می کند. این به معنای پرهیز از هر گونه تلاشی برای کاهش حجم کار نیست، بلکه بیشتر به معنای شفاف بودن در مورد کاری که باید انجام شود، ایجاد ارزش و ارائه اهداف خاص است. اگر تیم متوجه نیازمندی های ناهمسو شود، آنگاه تعداد کار بی ارزش را به حداکثر خواهد رساند - که باید از آن جلوگیری کرد تا هزینه ها و زمان را بدون دلیل موجه از دست ندهد.

اغلب، دو نوع محصولات ساده تعریف بدین صورت تعریف می شوند: MVP ها و MMP ها.

- کمینه محصول پذیرفتنی (MVP) مفهومی از استارتاپ ناب است ([Ries2011] را مشاهده کنید) و به عنوان کوچکترین محصولی تعریف می شود که می تواند 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 امکان پذیر می کند و بسیاری از استارتاپ ها با آن هماهنگ هستند.
- هدف از کمینه محصولات قابل بازاریابی (MMP) چیزی فراتر است. مسئله نه تنها ارائه بازخورد زود هنگام و در نتیجه هدایت مراحل بعدی نیازمندی ها است، بلکه ارزش آفرینی سریع است. بسیاری از محصولات را می توان در یک "نسخه 1" ساده مورد استفاده قرار داد بدون اینکه از قبل از همه ویژگی ها و کیفیت های مورد نظر برخوردار باشد، بنابراین از این طریق درآمد کسب می کند تا هزینه بهبود مستمر محصول پرداخت شود. در اینجا اغلب فرایند استارتاپ ناب (Lean Startup) (ایجاد، اندازه گیری، یادگیری) به صورت تکرار شونده مورد استفاده قرار می گیرند.

2.7 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 در مورد اینکه آیا روند توسعه برای آن ها به خوبی پیش رفته یا نیاز به بهبود دارد، بحث کند.

در این فرآیند 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 می شود به سوالاتی مانند این پاسخ دهند: چه چیزی خوب کار کرد؟ 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 به سرعت (Velocity) خود نگاه کرده و سرعت برنامه ریزی شده برای تکرار بعدی را تغییر داده یا تنظیم کند.

این امر همچنین در فرآیند نیازمندی ها اعمال می شود. پیامدهای این بینش ها باید شامل انطباق کوتاه مدت (Short-term adaptation) با مراحل بهبود فرآیند باشد.

3 محصولات کاری و تکنیک ها در RE@AGILE (L2)

مدت زمان: 2 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

- 3.1.1 هدف آموزشی دانستن تفاوت میان مستندات مشخصات سنتی (Traditional specification documents) و بک‌لاگ ها (Backlogs)
- 3.1.2 هدف آموزشی دانستن ارزش چشم‌اندازها و اهداف
- 3.1.3 هدف آموزشی دانستن ارزش افزوده استفاده از مدل‌های زمینه در RE
- 3.1.4 هدف آموزشی دانستن نحوه تفاوت قائل شدن میان سه نوع از نیازمندی ها
- 3.1.5 هدف آموزشی درک سطوح مختلف جزئیات (Granularity) در نیازمندی ها
- 3.1.6 هدف آموزشی درک فرمت های مختلف مشخصات برای انواع مختلف محصول کاری در فرآیندهای چابک (به عنوان مثال متنی (Textual) در مقابل مبتنی بر الگو (Template-based) و نموداری (Diagrammatic))
- 3.1.7 هدف آموزشی درک ارزش اصطلاحات، واژه نامه ها و مدل های اطلاعاتی
- 3.1.8 هدف آموزشی درک مشخصات نیازمندی ها و محدودیت های کیفی در فرآیندهای مهندسی نیازمندی های چابک
- 3.1.9 هدف آموزشی دانستن معیار پذیرش و تناسب
- 3.1.10 هدف آموزشی درک کاربرد استفاده از تعریف آماده و تعریف انجام شده در فرآیندهای مهندسی نیازمندی های چابک
- 3.1.11 هدف آموزشی دانستن تفاوت میان نمونه اولیه، اینکریمنت و اسپایک (Spike)
- 3.1.12 هدف آموزشی دانستن انواع محصول کاری در فرآیندهای RE@Agile (مدل زمینه، اپیک، داستان کاربر، بک‌لاگ، نقشه راه، نیازمندی، تعریف انجام شده، تعریف آماده)
- 3.2.1 هدف آموزشی درک چگونگی استخراج نیازمندی ها در مهندسی نیازمندی های چابک
- 3.2.2 هدف آموزشی دانستن چگونگی ایجاد و نگهداری بک‌لاگ ها در فرآیندهای مهندسی نیازمندی های چابک.
- 3.2.3 هدف آموزشی دانستن چگونگی اعتبارسنجی و مذاکره در مورد نیازمندی ها در RE@Agile
- 3.2.4 هدف آموزشی دانستن نحوه مدیریت نیازمندی ها در RE@AGILE

3.1 محصولات کاری در RE@Agile (L2)

3.1.1 مستند مشخصات (Specification Documents) در مقابل بک‌لاگ محصول (Product Backlog)

نمی‌توان تنها به ذکر نیازمندی ها 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 [Scrum2020]. فارغ از هر متدولوژی، این مجموعه نیازمندی به عنوان محصول کاری اصلی مهندسی نیازمندی ها، تحلیل گران کسب و کار، مالکان محصول یا هر شخص مسئول برای مهندسی نیازمندی ها در نظر گرفته می‌شود. متدهای مختلف، انواع و نام های مختلفی را برای چنین مجموعه نیازمندی ها پیشنهاد می‌دهند.

مهندسی نیازمندی ها اغلب این محصول کاری را نتیجه فرآیند استخراج، مشخصات نیازمندی های کاربر، مشخصات نیازمندی های سیستم یا مشخصات نیازمندی های نرم افزار، بسته به اینکه چه کسی آن را می‌نویسد و جزئیات آن را پوشش می‌دهد، نام گذاری می‌کند. این مورد لزوماً شامل یک مستند نیست، بلکه فقط مجموعه ای از نیازمندی ها به هر شکل فیزیکی (کاغذ (Paper)، مخزن (Repository)، پایگاه داده (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، 2.1 را مشاهده کنید) تعریف می‌کنند و همچنین بک‌لاگ اسپرینت را برای آن دسته از نیازمندی هایی که برای تکرار بعدی انتخاب شده اند (اسپرینت در اسکرام)، تعریف می‌کنند [Scrum2020]. مجدداً، فرم فیزیکی بک‌لاگ دارای اهمیت نیست. ممکن است بک‌لاگ 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 2.2 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 نیازمندی ها مستند می شوند، حتماً 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. داشتن 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.

3.1.2 چشم انداز و اهداف

هر فرآیند توسعه باید براساس چشم اندازها یا اهدافی که قابلیت محصول را تعریف می کنند، هدایت شود که در صورت پیاده سازی، به معنای موفقیت آمیز بودن راه حل است. داشتن چنین چشم اندازها یا اهدافی که در اسرع وقت مورد توافق کلیه ذینفعان مربوطه باشد، برای هر فعالیت مربوط به نیازمندی های سیستم یا محصول مربوطه از اهمیت بالایی برخوردار است.

در فرآیندهای توسعه چابک معمولاً از اصطلاح "چشم انداز محصول" استفاده می شود تا تأکید شود که هر نتیجه فرآیند توسعه باید دارای یک ارزش مشخص مربوط به چشم انداز محصول باشد. برای تعریف این ارزش، ممکن است لازم باشد ابتدا به وضوح مشخص کنیم که ارزش های یک شرکت چیست.

اهداف ذینفعان مختلف می تواند متناقض باشد، این بدین معنی است که ذینفعان مربوطه باید برای رسیدن به یک چشم انداز یا مجموعه اهداف توافق شده مذاکره کنند. از سوی دیگر، این مورد می تواند بیانگر این باشد که انواع مختلفی از محصول (به عنوان مثال یک آیفون کوچک و بزرگ)، یا حتی به طور کلی محصولات مختلف (آیفون و آی پد) مورد نیاز است.

توسعه چابک اغلب از اهداف با جزئیات متفاوت مانند اهدافی برای افق های زمانی مختلف (Differernt time horizons) یا فواصل برنامه ریزی (Planning intervals) استفاده می کند. به عنوان مثال، ممکن است اهداف سالانه جهت مذاکره در مورد تحویل دادنی ها (زمان و محتوا)، اهداف سه ماهه برای برنامه ریزی اسپرینت و اهداف تکرار/اسپرینت برای تکرار/اسپرینت بعدی وجود داشته باشند [High2009].

چشم اندازهای بلند مدت محصول و اهداف کوتاه مدت برای تأکید بر مهم ترین دستاوردهایی که باید در یک بازه زمانی خاص به دست آیند مفید هستند و به همسویی کلیه ذینفعان در یک "مأموریت مشترک" کمک می کنند. همه این اهداف را می توان در یک جدول زمانی در یک نقشه راه مفید نشان داد.

چشم انداز یا اهداف محصول، انتزاعی ترین شکل نیازمندی ها هستند و بدون پالایش بیشتر نمی توان آنها را توسعه داد. آنها درک و راهنمایی جامعی برای کل فرآیند توسعه چابک ارائه می دهند. 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 مجموعه ای از اهداف، شاخصی برای از دست دادن ارزش باشد. در نتیجه، بیانیه چشم انداز محصول و اهداف مورد توافق ذینفع، محصولات کاری بسیار مهمی برای موفقیت در

فرآیندهای توسعه چابک است، زیرا آن ها بدون اینکه بی جهت خلاقیت توسعه دهندگان را محدود کنند، چارچوبی را برای کلیه فعالیت های توسعه تعیین می کنند.

3.1.3 مدل زمینه (Context model)

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

نیازمندی های سیستم یا محصول اغلب با در نظر گرفتن فرضیات مربوط به محیط مشخص می شوند. مدل های زمینه یک روش ساختار یافته برای مستند سازی 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، نیازمندی های مشخص شده فقط در صورت صحت فرضیات مستند شده در مدل های زمینه درست تلقی خواهند شد، به این معنی که مدل های زمینه، به نمایش زمینه عملیاتی درست سیستم یا محصول می پردازند.

مدل های زمینه، محصولات کاری قدرتمندی هستند، زیرا به وضوح بین سیستم یا محصولی که قرار است ساخته شود و زمینه آن، به عنوان مثال شامل سیستم های مجاور و کاربران انسانی ()

[CPREFL2022] را مشاهده کنید)، تمایز ایجاد می کنند. با استفاده از این تمایز، عملکرد (functionality) را می توان تخصیص داد.

این امر اجازه می دهد تا میان مسئولیت های سیستم یا محصول و مسئولیت های مجاور یا کاربران انسانی در زمینه که در حال همکاری در طول یک عملیات برای تحقق چشم انداز کلی هستند، تمایز قائل شویم. بنابراین، از مدل های زمینه می توان برای شفاف سازی و مشخص نمودن رابط های خارجی (External interfaces) سیستم یا محصول مورد استفاده نیز استفاده کرد.

چنین مدل هایی را می توان با استفاده از قالب های مختلف، مانند نمودارهای زمینه پیشنهاد شده برای تجزیه و تحلیل سیستم ساختاری،

استفاده از دیاگرام های مورد کاربرد (Use case diagram)، نمودارهای تعریف بلوک سیس.ام.ال (SysML block)

(definition diagram)، نمودارهای مولفه های 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

رابط های خارجی با اشخاص یا سیستم های 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 جعبه ای و خطی دستی (Simple hand-drawn box and lines

diagrams) نیز می توانند عملی و مفید باشند.

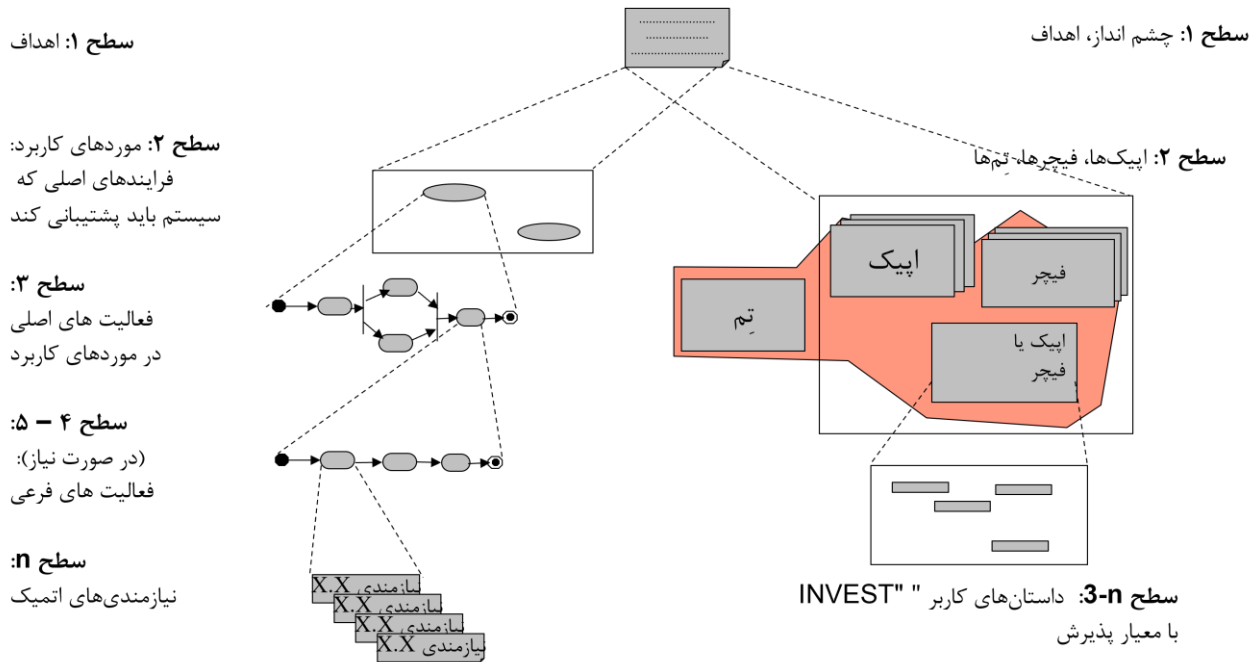
مستقل از فرم مستندات، یک مدل زمینه یک محصول کاری بسیار با ارزش است و در طی فرایند توسعه چابک توصیه می شود. این

مدل، زمانی که رابط های خارجی (یعنی مرز بین دامنه و زمینه) باید با سیستم های مجاور در ارتباط باشند، به تعیین محدوده ای (در محدوده حوزه) می پردازد که تحلیل گران در آن در تصمیم گیری آزاد هستند.

3.1.4 نیازمندی ها

بنابراین، نیازمندی ها باید بر اساس چشم انداز و اهداف ضبط شده و محدود به مدل زمینه باشند. به طور معمول، بین سه نوع نیازمندی تمایز وجود دارد: نیازمندی های عملکردی (Functional requirements)، نیازمندی های کیفی (Quality requirements) و محدودیت ها (Conflicts).

ذی نفعان اغلب نیازهای خود را در سطوح مختلفی از جزئیات اعلام می کنند: از نیازهای درشت دانه، مانند اهداف کلی کسب و کار، تا نیازهای ریز دانه با جزئیات عملکرد سیستم مورد انتظار. نیازمندی های عملکردی و کیفی (**Fehler! Verweisquelle konnte nicht gefunden werden.** می توانند (و باید) مورد بحث و مستند سازی در سطوح مختلف انتزاع قرار گیرند.



شکل 1: دانه بندی نیازمندی ها: تجزیه مورد کاربرد و انواع چابک، موردهای کاربرد، مشخصات مورد کاربرد و دیگرام های فعالیت

یک روش معمول از روش های غیر چابک 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 دانه بندی، هر مورد کاربرد برای شرح مراحل انجام این عملکرد شرح داده شده است. چندین گزینه با توجه به سطح دانه بندی در اینجا موجود است:

1. شرح کلامی (Verbal description) (متن ساده (Plain text))
2. الگوهای مورد کاربرد (جریان نیمه ساختار یافته یا داستانی (Semi-structured format or narrative flows))
3. دیگرام فعالیت یو.ام.ال (UML Activity diagram) (یا هر نوع مدل کاربردی گرافیکی دیگر مانند نمودار جریان داده (Data flow diagram)، BPMN، نمودار زمینه (Context diagram)، نمودار حالت (State diagram)، مدل شی کسب و کار (Business object model) و غیره)

به عنوان مثال، سطح 3 در شکل 1 موردهای کاربرد تشریح شده با نمودارهای فعالیت یو.ام.ال (UML Activity diagram) را نشان می دهد.

بر اساس پیچیدگی مسئله، می تواند سطح بیشتری از جزئیات فراهم شود، به عنوان مثال تجزیه فعالیت ها به فعالیت های فرعی (سطح 4 در شکل 1 را مشاهده کنید) یا با مشخصات متنی نیازمندی ها از فعالیت های واحد (Individual activities) (سطح 5 در شکل 1 را مشاهده کنید).

بنابراین نیازمندی های دانه درشت 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 تجزیه و تحلیل و تجزیه نیازمندی از بالا به پایین است. این دیدگاه تا حدی آرمانی همیشه واقع بینانه نیست. در واقع ممکن است نیازمندی ها ابتدا در سطح جزئیات راهکار، با اهداف کلی‌تر که بعداً تعیین می‌شوند، بیان شوند؛ بنابراین روند دنیای واقعی ممکن است از بالا به پایین، از پایین به بالا یا ترکیبی از این ها باشد.

نکته اصلی این است که متدها و تکنیک هایی وجود دارند تا از این مباحث در سطوح مختلف انتزاعی شان پشتیبانی کنند و با افزایش دانش کلی نیازمندی ها، به ایجاد یک سلسله مراتب معنی دار از نیازمندی ها بپردازد.

یک گزینه برای توصیف موردهای کاربرد با پیچیدگی کم می تواند استفاده از چند جمله برای شفاف کردن گام های مورد نیاز برای اجرا باشد. برای موردهای کاربرد با پیچیدگی متوسط، ممکن است گزینه‌های دیگری مناسب باشند، زیرا الگوهای مورد کاربرد می توانند شرح مراحل پیچیده تر فرآیند را که بخشی از مورد کاربرد خاصی هستند، فراهم کنند. این قالب نیمه ساختار یافته به درک نیازمندی های ثابت شده در یک مورد کاربرد با یک راهنمای مشخص کمک می کند. با این وجود، الگوی مورد کاربرد در صورت وجود فعالیت‌های موازی و پیچیدگی زیاد با چندین سناریو، محدودیت‌هایی دارد.

در نهایت، نمودارهای فعالیت، به مهندس نیازمندی ها اجازه می دهند تا از یک قالب گرافیکی که در آن چندین سناریو قابل نمایش است، استفاده کند. دیاگرام می تواند چندین جریان موازی و متوالی از مراحل را نشان دهد. این فرم از توضیحات مورد کاربرد اجازه می دهد تا جریان های پیچیده مورد استفاده در یک نمودار شرح داده شوند.

موضوعی که همواره منجر به بحث می شود، زمانی است که نیازمندی با جزئیات کافی توصیف می گردد. تمرکز اصلی در اینجا بر روی همکاری در تیم است. یک نیازمندی زمانی با جزئیات کافی توصیف می شود که تیم (به ویژه افرادی که نیازمندی را پیاده سازی می کنند، به عنوان مثال توسعه دهندگان) متوجه شده باشند که چه چیزی از آن ها خواسته شده. توصیه می شود 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: اپیک ها، تم ها، فیچرها و داستان های کاربر

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

در اینجا، اهداف سطح بالای کسب و کار و عملکرد پیچیده به عنوان اپیک یا فیچر ثبت شده و براساس تم ها دسته بندی می شوند. چنین موضوعاتی ممکن است به قدری پیچیده باشند که به بیشتر از یک اسپرینت برای توسعه توسط تیم اسکرام نیاز داشته باشند [Griff2015].

اینگونه مباحث پیچیده سپس به داستان های کاربر کوچک تر تجزیه می شوند. قبلاً معیار برای داستان کاربر "خوب"، مورد بحث قرار گرفت (2.2). آن ها باید از "تعریف آماده" [Kron2008] نظیر تحقق معیار INVEST [Wake2003]، 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 بالا به پایین برای پالایش اپیک ها (Refining epics) به فیچرها و یا داستان های کاربر را دنبال می کند، یا اینکه داستان های کاربر ابتدا به صورت جداگانه با تم های مشترک جمع آوری می شوند و اپیک هایی که بعداً از بالا به وجود می آیند، به ماهیت پروژه و ذینفعان مرتبط بستگی دارد. در هر صورت چابکی، محصولات کاری را برای بحث و

اولویت بندی در مورد تصویر کلی (Big picture) و نیازمندی های آماده (Development-ready requirements) برای توسعه فراهم می کند.

3.1.6 مدل های گرافیکی و توضیحات متنی

مستقل از دانه بندی، همیشه بحث بر سر انتخاب نماد (Notation) در مورد نیازمندی های عملکردی وجود دارد. این نمادها را می توان به صورت متنی نیمه ساختاریافته (مانند داستان های کاربر منطبق با الگو)، یا به عنوان یک زبان ساده طبیعی یا رسمی (مانند Gherkin) بیان کرد که به روشنی بیان می کند راهکار باید چه کاری انجام دهد.

از آنجایی که این موضوع به خوبی شناخته شده است که زبان طبیعی گاهی اوقات به اندازه لازم دقیق و بدون ابهام نیست، بسیاری از نمادهای گرافیکی برای غلبه بر این ابهام یا نشان دادن وابستگی متقابل، برای مثال میان داده ها و فرایند، ایجاد شده اند. مثال ها شامل دیاگرام های فعالیت یو.ام.ال (UML Activity diagram)، دیاگرام های BPMN، نمودارهای جریان (Flow charts)، نمودارهای حالت (State chart)، دیاگرام های توالی (Sequence diagrams) و غیره می شوند.

اغلب این نمادهای مبتنی بر گرافیک، بر جنبه های مختلفی تاکید دارند. نمودارهای فعالیت یا نمودارهای BPMN برای فرآیندهای نسبتاً خطی که مراحل متوالی، گزینه ها یا حلقه ها را نشان می دهند، مناسب هستند. برعکس، نمودارهای حالت هر زمان که رویدادهای ناهمگام (Asynchronous events) بتوانند جریان را تحت تأثیر قرار دهند، عالی هستند. نمودارهای توالی برای "بیان مشخصات به همراه مثال" (Specification by example) مناسب هستند، زیرا آن ها حتی در صورتی که کامل نباشند، سناریوهای مشخصی از تعاملات را نشان می دهند.

در مدل سازی فرایندها و داده ها، یا داده ها و تعاملات، یا فرایندها و رابط ها نکات مثبت و منفی وجود دارد. به دلیل اینکه این موارد به هم مرتبط هستند (داده از فرایند پشتیبانی می کند)، در نتیجه هرگز نمی تواند یک وضعیت وجود داشته باشد که فقط پیشنهاد می کند تا از این مدل یا آن مدل استفاده شود. چنین رویکردی کاملاً بر ضد نتیجه خواهد بود. درحالی که بسیاری از ذی نفعان درک آسان تری از نیازمندی های متنی دارند، اما این نوع از نیازمندی ها به راحتی قابل تفسیر یا برداشت اشتباه توسط خوانندگان یکسان هستند. مدل های گرافیکی رسمیت بیشتری را فراهم می کنند بنابراین از تفسیرهای مختلف یا درک اشتباه جلوگیری می کنند. انتخاب سبک باید توسط اهداف کلیدی مهندسی نیازمندی ها تعیین شود، اطمینان از درک مشترک بین همه ذینفعان از یک طرف و محافظت کافی در برابر ناقص بودن و درک های اشتباه از طرف دیگر.

3.1.7 تعریف اصطلاحات، واژه نامه ها و مدل های اطلاعاتی

نیازمندی های عملکردی، بدون درک دقیق تمام اصطلاحات استفاده شده در جمله ها و مدل های گرافیکی، ناقص هستند.

بنابراین، مجموعه ای از کلیه اصطلاحات مربوط به کسب و کار که در محصول کاری نیازمندی ها به کار رفته است، به عنوان یک محصول کاری ضروری در نظر گرفته می شوند، اما در هنگام تمرکز بر نیازمندی های کسب و کار، به راحتی نادیده گرفته می شوند. شکل این محصول کاری به صورت حداقلی، لیستی از اصطلاحات و اختصارات کسب و کاری (توصیف شده به صورت متنی) است که معمولاً برای جستجوی آسان تر به ترتیب حروف الفبا طبقه بندی می شوند. برخی اوقات، این مورد را دیکشنری، واژه نامه یا لیست تعاریف می نامند.

هنگامی که کسب و کار بسیار پیچیده است، می توان این واژه نامه را با گروه بندی اصطلاحات ساده در کلاس ها (یا موجودیت ها) و نمایش نمای کلی تمامی اصطلاحات در قالب گرافیکی، ساختاردهی کرد. به چنین محصولات کاری مدل های داده (Data models)، مدل های اطلاعات (Information models)، دیاگرام های موجودیت-رابطه (Entity-relationship diagrams) یا نمودار کلاس یو.ام.ال (UML Class Diagram) گفته می شود. علاوه بر تعریف اصطلاحات، این مدل ها همچنین شامل ارتباطات مربوطه بین این موجودیت ها (Entities)، یعنی روابط ایستا (Static relationship) میان اصطلاحات هستند.

همانگونه که در بالا ذکر شد، بک لاگ محصول اغلب متشکل از اپیک ها، فیچرها و داستان های کاربر است که بر عملکرد مورد نیاز تأکید می کند و تأکید بر تعاریف اصطلاحات کسب و کار را کمرنگ می کند. با این وجود، حتی در رویکردهای چابک، درک صحیحی از اصطلاحات کسب و کار برای ایجاد درک مشترک و همسو از اصطلاحات مورد استفاده در محصولات کاری نظیر اپیک ها، فیچرها و داستان های کاربر لازم است.

3.1.8 نیازمندی های کیفی و قیدها

علاوه بر نیازمندی های عملکردی (مشخص کردن عملکردهای سیستم یا محصول مورد نیاز برای ارائه)، نیازمندی های کیفی و قیدها برای موفقیت سیستم یا محصول در حال توسعه از اهمیت اساسی برخوردار است. به طور سنتی، نیازمندی ها و قیدهای کیفی در چارچوب اصطلاح "نیازمندی های غیر عملکردی" (Non-functional requirements) گنجانده می شوند [CNYM2000]. کارشناسان مهندسی نیازمندی ها برای دهه ها، بر اهمیت این نیازمندی های "غیر عملکردی" تأکید 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 [Glinz2026]، استفاده می کند.

نیازمندی های کیفی، مرتبط با ویژگی های خاصی است که یک سیستم یا محصول برای ارائه نیاز دارد، به عنوان مثال، می توان از عملکرد (Performance)، قابلیت اطمینان (Reliability)، ایمنی (Safety)، امنیت (Security) یا قابلیت استفاده (Usability) نام برد [ISO25010]. 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 (CPREFL2022] 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 نیازمندی های کیفی به وسیله بازسازی مجدد (Refactoring) در سیستم های موجود، بسیار دشوار است. بنابراین در نظر گرفتن چنین جنبه هایی در مراحل اولیه بسیار ارزشمندتر است.

قیدها (Constraints) به تعریف محدودیت های کلی در فضای راهکار سیستم یا محصول مورد توسعه می پردازند. قیدها به اشکال مختلف وجود دارند: قیدهای سازمانی (قیدهای بودجه، برنامه های زمانی فشرده، روند توسعه تجویزی و غیره)، قیدهای فنی (نیاز به سیستم خاص DB، استفاده از زبان برنامه نویسی خاص، چارچوب های انتخاب شده و غیره) یا قیدهای خاص محیط (Environnement) که سیستم در آن کار خواهد کرد (استانداردها (Standards)، هنجارها (Norms)، مقررات (Regulations) و غیره).

مشابه نیازمندی های کیفی، یادگیری خیلی دیر در مورد قیدهای کلیدی می تواند بسیار پر هزینه باشد، زیرا بسیاری از این جنبه ها را نمی توان به صورت تدریجی اضافه کرد. تصمیم گیری در زمینه برنامه ریزی 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 نیازمندی های عملکردی، ممکت است نیازمندی های کیفی و قیدها در سطوح مختلف انتزاع وجود داشته باشند و باید در بک لاگ محصول ثبت شوند، برای تیم قابل مشاهده باشند و در هر تکرار مورد آزمون قرار گیرند. بنابراین، ممکن است این امر که محدودیت ها را به "تعریف انجام شده" (Definition of Done) اضافه کنیم و اعتبارسنجی آن ها را در قالب آزمون های رگرسیون (Automated regression test) خودکار انجام دهیم، مفید باشد.

3.1.9 معیار پذیرش (Acceptance criteria) و معیار تناسب (Fit criteria)

انواع نیازمندی ها از ارزش محدودی برخوردارند، اگر تحقق آن ها قابل اعتبارسنجی، بررسی یا آزمودن نباشد. بنابراین، هر یک از نیازمندی ها به مجموعه معیارهایی نیاز دارند که بتوان به وسیله آن ها، نیازمندی ها را مورد آزمون قرار داد تا برآورده شدن آن ها مورد بررسی واقع شود. علاوه بر این، این معیارها با توصیف دقیق آنچه انتظار می رود به درک (و تشویق بازخورد) کمک می کنند.

نوع معیارهای استفاده شده با سطح جزئیات نیازمندی مطابقت دارد:

- در سطوح انتزاع بالاتر (چشم انداز، اهداف و اپیک ها)، معیارهای موفقیت [SAFE] معمولاً به این دلیل تعریف می شوند که اندازه گیری تنها در صورت فراهم بودن یا عدم وجود قابلیت یا توانایی لازم امکان پذیر است.
- به منظور کسب مقبولیت، می توان در سطوح انتزاع پایین از معیارهای پذیرش برای توصیف چگونگی آزمودن راهکار مرتبط با نیازمندی، استفاده کرد.

در متدهای غیر چابک و چابک این موضوع مورد توافق است که نیازمندی ها باید قابل تایید (Verifiable) باشند. متدهای غیر چابک اغلب از اصطلاحاتی مانند "معیارهای کیفیت" (Quality criteria) یا "معیارهای تناسب" (Fit criteria) یا "موردهای آزمون" (Test cases) قدیمی استفاده می کنند. در چابک، اصطلاحات "معیارهای پذیرش" (Acceptance criteria) (برای داستان های کاربر) یا "معیارهای موفقیت" (Success criteria) (برای اپیک ها یا تم ها) بیشتر رایج است.

3.1.10 تعریف آماده (Definition of Ready) و تعریف انجام شده (Definition of Done)

در حالی که معیارهای پذیرش و تناسب، به نیازمندی های کسب و کار تعلق دارند و آنها را تکمیل می کنند، محصولات کاری، "تعریف آماده" (DoR) و "تعریف انجام شده" (DoD)، از فرایند رسمی توسعه پشتیبانی می کنند و از کیفیت نیازمندی ها و اینکریمنت محصول اطمینان حاصل می کنند. تعریف تمام شده (DoD) بخشی رسمی از راهنمای اسکرام است [Scrum2020] و به عنوان یک تعهد به اینکریمنت تشریح شده است. تعریف تمام شده به عنوان یک دروازه کیفیت (Quality gate) برای فرآیند توسعه چابک عمل می کند، در حالی که DoR یک پرکنش خوب است که باید به ایجاد نیازمندی های ارزشمند کمک کند و از اضافه بار تیم با نیازمندی های نامشخص جلوگیری کند. DoR کمک می کند تا نیازمندی ها به جزئیات مناسب برسند و اطلاعات کافی را برای مذاکره میان مالک محصول/مهندس نیازمندی ها و توسعه دهندگان ارائه می دهد.

3.1.11 نمونه اولیه در مقابل اینکریمنت ها (Prototype vs Increments)

روش دیگر کار با نیازمندی ها، استفاده از نمونه های اولیه است، زیرا بسیاری از ذینفعانی که به سیستم یا کالایی نیاز دارند، نمی خواهند مستندات برای تعریف محصول بنویسند یا بخوانند - آنها موفقیت فوری می خواهند. برای آن دسته از افراد، بهترین راه برای درک نیازمندی های آن ها و دریافت بازخورد، نمایش عملکردها یا قابلیت های سیستم در قالب یک سیستم در حال اجرا است.

یکی از راه های انجام این کار استفاده از محصولات حداقلی و تدریجی (Minimal and incremental products) است. مهندسی نیازمندی ها دو نوع را پیشنهاد می کند: اینکریمنت های "افقی" محصول (Horizontal product increments) برای نشان دادن انواع مختلف برای اعتبار سنجی ("انجام کارهای بیشتر") و اینکریمنت "عمودی" محصول برای تأیید درست بودن رویکرد توسعه ("انجام کار درست").

با ارائه تعامل مستقیم با سیستم کاری، نمونه های اولیه می توانند برای به دست آوردن بازخورد بسیار مفید باشند. به عنوان مثال مشخص کردن ویژگی های کاربردی، مانند زمان واکنش (Reaction time) دشوار است اما هنگام استفاده از نرم افزار قابل استفاده (Working software)، شناسایی آن آسان است. با این حال، نمونه های اولیه نیز ممکن است مایوس کننده باشند - مایوس کننده برای کاربران، به این دلیل که معتقد هستند که توسعه به اتمام رسیده است و مایوس کننده برای توسعه دهندگان، زیرا اغلب این نمونه های اولیه دور ریخته می شوند تا بعدها با فناوری بهتر جایگزین شوند.

متدهای چابک سعی دارند با ایجاد سریع اینکریمنت با کیفیت خوب، سیستم یا محصول "واقعی"، از دور ریختن نمونه های اولیه جلوگیری کنند. چابک برای تکرارهای کوتاه تلاش می کند تا محصولات قابل مشاهده ای را نشان دهد که به نوبه خود برای کسب اطلاعات بیشتر در مورد نیازمندی ها استفاده می شوند. با توجه به اینکه هدف این نیست که کد توسعه یافته را دور ریخته شود، بازسازی مجدد (Refactoring) یک به روش چابک است و به عنوان پاسخی به تغییر نیازمندی های عملکردی یا کیفی بر اساس بازخورد کاربر در نظر گرفته می شود.

اصطلاح "اسپایک" (Spike) در چابک برای اشاره به یک تکرار توسعه (Development iteration) است که با هدف درک صحیح بخشی از پیچیدگی (به عنوان مثال معماری سیستم) و در نتیجه کاهش ریسک استفاده می شود. این اصطلاح اگرچه به طور دقیق تعریف نشده است، اما می تواند به اعتبارسنجی یک تسک یا تکرار کامل (Whole iteration) اشاره داشته باشد. نمونه سازی اولیه یک تکنیک معتبر است که در آن برخلاف سایر تکرارهای چابک، هدف اصلی آن به جای کد قابل استفاده، به دست آوردن دانش است.

3.1.12 خلاصه محصولات کاری

همانطور که در پاراگراف های گذشته ذکر شد، برخی از محصولات کاری برای توسعه موفقیت آمیز بسیار مهم هستند:

- شروع با چشم اندازها و اهداف، یک پرکنیس خوب است.
- شناسایی و دانستن مهم ترین ذی نفعان، همیشه یک پرکنیس خوب است.
- تعیین صریح دامنه و محدود کردن آن از زمینه، یک پرکنیس خوب است.

حتی اگر مهندسی نیازمندی های غیر چابک و مهندسی نیازمندی های چابک اصطلاحات مختلفی به کار ببرند، بر ضروری بودن درک عملکرد و همچنین اصطلاحات کسب و کار و ضبط نیازمندی های عملکردی (از جمله عملکرد و داده ها) توافق می کنند.

علاوه بر این، متدهای چابک و غیر چابک باید نیازمندی های کیفی و قیدها را شناسایی کنند، زیرا این امر ممکن است به شدت بر تصمیمات طراحی تأثیر بگذارد. بی توجهی به آن ها یا بسیار دیر آموختن درباره آن ها، ممکن است منجر به کار مجدد و همچنین محصولی شود که انتظارات مشتری را برآورده نکند.

مهندسی نیازمندی های خوب تضمین می کند که هیچ موضوع مرتبطی فراموش نمی شود. بنابراین متدهای غیر چابک اغلب بر مستند کردن بسیاری از زمینه های بالقوه مورد علاقه مربوط به نیازمندی های یک سیستم اصرار دارند.

رویکردهای چابک از محصولات کاری کمتر رسمی استفاده می کنند و مستندات از دست رفته را جایگزین حلقه های ارتباط مستقیم و بازخورد سریع می کنند که از طریق توسعه تدریجی محصول یا نمونه های اولیه امکان پذیر است.

اصل دوم مانیفست چابک که اغلب نقل می شود [AgileMan2001]، دقیقاً همان چیزی را که مورد بحث ما قرار گرفت را توضیح می دهد:

„... از طریق این کار، ما به این ارزش رسیده ایم: ... نرم افزار کارکننده ارجح بر مستندات جامع ...”

ما متقاعد شده ایم که بررسی عمده آنچه باید به صورت مکتوب بیان شود، آنچه می توان در مورد آن بحث کرد و آنچه که می توان با نمونه اولیه یا به صورت اینکریمنت نشان داد، برای هر سازمانی بسیار مثر ثمر است. بهترین نتایج زمانی حاصل خواهد شد که مستندات، ارتباطات و نمونه سازی/توسعه تدریجی با توجه به محدودیت ها و فرهنگ شرکت متعادل شود. فصل 4 بیشتر، موضوع تعادل بخشیدن به فعالیت های مقدماتی (Upfront activities) (و معماری) با فعالیت های تکراری (Iterative activities) را پوشش خواهد داد.

3.2 تکنیک های RE@Agile (L2)

در 3.1 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 [CPREFL2022]، این فعالیت ها را به

روش زیر ساختاردهی می کند:

- استخراج نیازمندی ها
- مستندسازی نیازمندی ها
- اعتبارسنجی و مذاکره نیازمندی ها
- مدیریت نیازمندی ها

در ادامه، زیر بخش های ارائه شده این فعالیت ها را از منظر چابک به بحث می کشد.

3.2.1 استخراج نیازمندی ها

مهندسی نیازمندی ها در توسعه چابک بر پایه ارتباطات نزدیک میان تمامی ذی نفعان (از جمله توسعه دهندگان) برای استخراج نیازمندی ها، بنا شده است. ارتباطات غیررسمی و مستقیم بین اعضای تیم می تواند به خودی خود، ترکیب خوبی از مصاحبه و توفان فکری در نظر گرفته شود. تکنیک هایی مانند "مشتری در محل" (Onsite customer) متد XP، می توانند به همان اندازه در قالب جلسات بین مالک محصول و ذینفعان (به طور معمول توسعه دهندگان). هدف در همه موارد، کسب بینش بیشتر از آنچه واقعاً مورد نیاز است، می باشد.

مهندسی نیازمندی ها بیش از اینکه به طور معمول در زمینه توسعه چابک بحث کند، به فراهم کردن طیف وسیعی از تکنیک ها برای کشف و استخراج نیازمندی ها می پردازد این موارد شامل تکنیک های Q/A (نه تنها مصاحبه، بلکه پرسشنامه)، تکنیک های مشاهده، تکنیک های مبتنی بر محصول کاری (استفاده مجدد System archeology، Reuse) و غیره)

[CPREFL2022] و تکنیک های خلاقیت مانند توفان فکری یا تفکر دیزاین است. این تکنیک ها، از ایده ارائه نیازمندی ها در قالب داستان کاربر پشتیبانی می کنند که مبنایی برای بحث ساختار یافته هستند و نسخه ای برای اجرا تلقی نمی شوند.

همانطور که در بخش **Fehler! Verweisquelle konnte nicht gefunden werden.** ذکر شد، نمونه های اولیه و اینکریمنت محصول از جمله روش هایی برای کسب اطلاعات بیشتر در مورد نیازمندی ها هستند. به محض این که اینکریمنت محصول در جلسه بررسی اسپرینت یا جلسه دمو ارائه شود، ممکن است ایده های جدیدی ارائه شوند که می توانند مستقیماً به بک لاگ محصول وارد شده و توسط مالک محصول اولویت بندی شوند.

مهندسی نیازمندی ها در توسعه چابک می تواند با مطالعه تکنیک های مختلف استخراج و تصمیم گیری حساب شده در مورد انتخاب ترکیبی از تکنیک ها، از مزایای مهندسی نیازمندی های غیر چابک در پروژه ها و توسعه محصولات بهره مند شود. در حالی که برقراری ارتباط نزدیک میان ذینفعان و بازخورد سریع از طریق اینکریمنت محصول، ایده بسیار خوبی است، اما استخراج چیز دیگری بیش از این است. به عنوان مثال، تنها بهره مندی از ارتباط کلامی در زمانی که صدها یا هزاران ذی نفع وجود داشته باشند، کافی نیست. هنگام تلاش برای کشف نیازمندی ها یا نیازمندی ها نوآورانه، در جایی که ذینفعان حتی از امکان آن آگاه نیستند، ممکن است تکنیک های خلاقیت مورد نیاز باشد.

هنگام کار با محدودیت زمان بندی دقیق (یعنی وقت کافی برای بحث های فشرده)، ممکن است تکنیک هایی مانند کارت های برفی (Snow cards) بسیار کارآمد باشند. هنگام کاوش در فن آوری های جدید، اسپایک ها (جدول زمانی، اینکریمنت های ساده برای کشف راه حل های بالقوه) یک ایده عالی به حساب می آیند.

3.2.2 مستندسازی نیازمندی ها

در چابک، نیازمندی ها درون بک لاگ سازماندهی می شوند. ممکن است نیازمندی ها در قالب کارت های داستان (Story cards) دارای ارزش و اولویت ثبت شوند. ممکن است این موارد در نگاشت های داستان (Story maps) سازماندهی شده یا به داستان های ساده تری تجزیه شوند. اصل مدنظر در مورد کارت ها این است که اندازه کارت، موجب محدود شدن آن چه روی آن نوشته شده می شود و به تمرکز روی جزئیات اصلی کمک می کند.

با این وجود، مستندسازی نیازمندی‌ها همچنان یک فعالیت مهم در جهت تقویت ارتباط بین همه ذینفعان به حساب می‌آید. اگر جزئیات به صورت 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. با توجه به 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.

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

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

1. **مستندات برای اهداف حقوقی (Legal):** دامنه‌ها یا زمینه‌های خاص پروژه (به‌عنوان مثال نرم‌افزار در بخش مراقبت‌های بهداشتی (Healthcare) یا هواپیمایی (Avionics)) برای اخذ تأییدیه قانونی نیاز به مستندسازی اطلاعاتی خاص (به‌عنوان مثال 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 جدا نشدنی از محصول باشند.

2. **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 به‌عنوان مثال 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 (به‌عنوان مثال "چرا ما تصمیم گرفتیم این مورد را اجرا نکنیم؟").

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

3. **مستندات برای اهداف ارتباطی (Communication):** ارتباط موثر 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). علاوه بر این، اطلاعات گاهی اوقات به‌قدری پیچیده هستند که ممکن است ارتباط مستقیم، ناکارآمد یا گمراه‌کننده باشد. به‌عنوان مثال، نمونه اولیه کاغذی یا نموداری از یک الگوریتم پیچیده می‌تواند بعدها مورد بازخوانی قرار گیرند. گاهی اوقات ذینفعان به سادگی، 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 می‌کنند.

اصل ایجاد مستندات برای اهداف ارتباطی این است: در صورتی که ذی‌نفعان یا توسعه دهندگان ارزشی در وجود مستندسازی ببینند، یک مستند میتواند به عنوان وسیله ارتباطی اضافی ایجاد شود. هنگامی که ارتباط موفقیت‌آمیز باشد، این مستند بایگانی می‌شود. هنگامی که ارتباط موفقیت‌آمیز است، این مستند بایگانی می‌شود.

4. **مستندات برای اهداف تفکر (Thinking):** جنبه ای که اغلب از نوشتن یک مستند فراموش می‌شود این است که نوشتن همیشه وسیله ای 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. به عنوان مثال، نوشتن یک مورد کاربرد (Use case) نویسنده را مجبور می‌کند تا در مورد تعاملات مشخص بین سیستم و بازیگران از جمله، به عنوان مثال، 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 قابل توجهی بهره مند شوند. به طور خلاصه می‌توان گفت که مستندسازی نیازمندی‌ها به خودی خود هدف نیست، بلکه باید ارتباط میان ذی‌نفعان را، به‌ویژه بین درخواست‌کننده (Requester) (که اغلب مالک محصول به‌عنوان جایگزین او انتخاب می‌شود) و توسعه دهندگان را تسهیل کند.

3.2.3 اعتبارسنجی و مذاکره نیازمندی‌ها

در حالی که مهندسی نیازمندی‌ها بر اعتبارسنجی نیازمندی‌ها از طریق متدهایی نظیر بررسی (Review)، رویدادهای بررسی فرآورده (Walkthrough)، بازرسی‌ها (Inspections) یا خواندن مبتنی بر چشم انداز (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 بحث برانگیز یا متناقضی است که در آن می‌توان از تکنیک‌های مذاکره و حل تعارض استفاده کرد.

از آنجا که 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.

اعتبار سنجی رسمی (Formal validation) نیز با نشان دادن نتایج سریع به صورت اینکریمنت‌های یکپارچه محصول، کاهش می‌یابد. اگر این اینکریمنت پاسخگوی تمام شرایط ذینفعان نباشد، دلتا (Delta) به عنوان نیازمندی جدید به بک لاگ محصول وارد می‌شود و با سایر آیتم‌های بک لاگ ارزیابی و اولویت بندی می‌شود.

با این وجود، بررسی‌های مربوط به بک لاگ محصول، بحث در مورد ارزش کسب‌وکار، بحث در مورد ریسک‌ها و مذاکرات فوری در مورد نیازها، 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 توسعه دهندگان (و در صورت در دسترس بودن، ذینفع) با هم کار می کنند تا سطح جزئیات نیازمندی را با استفاده از اصول پالایش مداوم (Continuous refinement) پیدا کنند.

در اولین نسخه های راهنمای اسکرام، جلسه پالایش بک لاگ محصول تنها به صورت غیر مستقیم ذکر شده بود. در نسخه جدید [Scrum2020]، پالایش (Refinement) به صراحت به عنوان فعالیتی برای استخراج، مستندسازی و اعتبارسنجی نیازمندی ها ذکر شده است. این فرآیند پالایش، که جلسه پالایش بخشی ضروری از آن است، مشکلات احتمالی را زود آشکار می کند و بعدها، زمان مورد نیاز برای جلسه برنامه ریزی اسپرینت را کاهش می دهد.

3.2.4 مدیریت نیازمندی ها

در RE سنتی، مدیریت نیازمندی ها مربوط به کلیه فعالیت ها برای رسیدگی به نیازمندی ها در طول زمان است. این مورد شامل مدیریت نسخه (Version management)، مدیریت تغییر (Change management)، مدیریت پیگیری (Configuration management)، قابلیت ردیابی (Traceability) و همچنین افزودن ویژگی هایی مانند وضعیت (Status)، تخمین ها (Estimation)، اولویت ها (Priorities)، پیوند به نیازمندی های مورد تعارض و افراد مشارکت کننده در ضبط، بررسی، امضا، پیاده سازی یا آزمون نیازمندی می شود.

همانطور که قبلاً بحث شد (3.1.1 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.

فعالیت هایی که در بک لاگ ها در قالب مدیریت نیازمندی ها انجام می شوند، شامل موارد زیر هستند:

1. اولویت بندی نیازمندی ها: تعیین ارزش (تجاری) نیازمندی ها برای تصمیم گیری در مورد زمان اجرای آن ها. هرچه ارزش کسب و کار بالاتر باشد، اولویت نیازمندی بالاتر است، زیرا پروژه های چابک سعی می کنند ابتدا عناصری را با بالاترین ارزش تجاری ارائه دهند. عوامل موثری که ارزش کسب و کار و همچنین اولویت بندی حاصل از آن را تعیین می کنند در سطح Practitioner و Specialist ماژول RE@Agile پوشش داده شده است [CPREALAGILE2022].
2. تخمین نیازمندی ها: تعیین کنید که چه مقدار کار برای تحقق آن ها انجام می شود. تخمین های بیش از حد بزرگ پیام واضحی به مالک محصول هستند که باید کار بیشتری انجام شود تا بتواند آن ها را به تعریف آماده (DoR) برساند (2 را مشاهده کنید). بنابراین او باید آیتم مربوطه را به واحدهای کوچکتر تقسیم کند تا تخمین ممکن شود.

این بدان معنا نیست که سایر فعالیت های مربوط به جنبه های تاریخی مدیریت نیازمندی ها نمی توانند در چابک انجام شوند. این فعالیت ها معمولاً در خارج از آیتم های بک لاگ ثبت می شوند.

در تصمیم گیری در مورد مناسب بودن فعالیت های مدیریت نیازمندی ها در یک شرایط خاص، مهندس نیازمندی ها باید به دنبال ایجاد تعادل میان به حداقل رساندن هزینه های اضافی، امکان تحویل به موقع راهکارهای قابل استفاده و نیازمندی های بلندمدت سازمان مانند انطباق قانونی (Legal compliance)، مستندات عملیاتی (Operational documentation) یا تحویل آن به اعضای جدید توسعه دهندگان باشد.

نتیجه گیری

فعالیت های نیازمندی ها نظیر استخراج (Elicitation)، مستندسازی (Documentation)، اعتبارسنجی (Validation) و مذاکره (Negotiation) و همچنین مدیریت (Management) نیازمندی ها به طور کلی باید در توسعه چابک انجام شوند. تکنیک های ترجیحی برای استخراج و مستندسازی ممکن است در توسعه چابک و غیر چابک متفاوت باشند، اما یادگیری از یکدیگر بهترین راه حل را نشان می دهد، زیرا این امر نه تنها موجب ترکیب قدرت ها می شود بلکه باعث کاهش هدر رفت (Waste) یا سربرار (Overhead) نیز خواهد شد. این روش کار نشان دهنده پنج ارزش اسکرام (تعهد (Commitment)، تمرکز (Focus)، صراحت

(Openness)، احترام (Respect) و شجاعت (Courage) و نماینده آن ها در سه ستون اسکرام (شفافیت (Transparency)، بازرسی (Inspection) و سازگاری (Adaptation) است [Scrum2020].

به خصوص صراحت و احترام، هنگام وارد کردن استاد کاری مهندسی نیازمندی ها (RE craftsmanship) به دنیای چابک و آوردن ایده ها و اصول چابک به استاد کاری مهندسی نیازمندی ها، بسیار مفید هستند.

در این واحد آموزشی، شما آموخته اید که حتی در RE@Agile، به جز داستان های کاربر، محصولات کاری دیگری در بک لاگ محصول وجود دارند و مهم ترین فعالیت های مهندسی نیازمندی ها را نباید فراموش کرد - اما ممکن است با تاکید و روش های مبتنی بر اصول چابکی ذکر شده در 1، انجام شوند.

4 جنبه های سازمانی (L2) RE@Agile

مدت زمان: 1 ساعت و نیم

اهداف آموزشی

- هدف آموزشی 4.1.1 درک تعامل بین ساختار سازمانی و RE@Agile
- هدف آموزشی 4.2.1 دانستن تعامل با ذینفعان در فرایندهای مهندسی نیازمندی های چابک
- هدف آموزشی 4.2.2 دانستن اینکه چگونه ارتباط و همکاری می تواند به بهبود نتایج کمک کند
- هدف آموزشی 4.2.3 دانستن نقش مدیریت در چابک
- هدف آموزشی 4.3.1 دانستن انگیزه برای مقیاس پذیری (Scaling)
- هدف آموزشی 4.3.2 دانستن ابعاد سازماندهی تیم ها
- هدف آموزشی 4.3.3 دانستن رویکردهای سازماندهی ارتباطات میان تیم ها
- هدف آموزشی 4.3.4 دانستن چارچوب های نمونه برای مقیاس پذیری
- هدف آموزشی 4.3.5 دانستن تأثیرات اصلی مقیاس پذیری در RE
- هدف آموزشی 4.4.1 دانستن معیارهایی برای تصمیم گیری در سطح مقدماتی (Upfront) در مقابل مهندسی مداوم نیازمندی ها
- هدف آموزشی 4.4.2 دانستن مقدار مناسب جزئیات برای آیتم های بک لاگ
- هدف آموزشی 4.4.3 دانستن ارزش اعتبارسنجی در RE@AGILE
- هدف آموزشی 4.4.4 دانستن چرخه به روزرسانی مناسب برای بک لاگ محصول
- هدف آموزشی 4.4.5 دانستن چگونگی یافتن زمان مناسب چرخه توسعه

4.1 تاثیر سازمان بر RE@Agile (L2)

ریشه چابک در تولید و کنترل فرآیند تجربی (Empirical process-control) است [TaNo1986] 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 [Conw1968]. انجام این کار باعث افزایش علاقه به موضوعاتی مانند مدیریت چابک (ACP/PMI) را مشاهده کنید) و سازمان های چابک ([Appel2011]، [Denn2015] را مشاهده کنید) شده که منجر به بحث در مورد جنبه هایی از چابک می شود که اغلب فراتر از توسعه نرم افزار) است.

ایده های قرار دادن مشتری در مرکز توجه (Customer at the center)، تیم های خود سازمانده (Self-organizing)، فعال کردن (Enabling) و توانمندسازی (Empowering) افراد و پیشرفت مداوم (Continuous improvement)، در دنیای وسیع کسب و کار طنین انداز 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 می دهد تا یک پیاده سازی اسکرام متناسب با نیازهای منحصر به فرد شرکت شما مقیاس پذیر شود.

در دنیای توسعه نرم افزار، بسیاری از پیاده سازی های فرایندهای توسعه چابک در مفهومی سازی (Conception) شکست می خورند، زیرا سایر اعضای سازمان قادر به تغییر در جهت پشتیبانی از تیم های چابک نبوده اند.

چرا چنین تغییر سازمانی لازم است؟

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

1. سازمان (یا بخشی از سازمان) تقاضا دهنده باید بتواند نیازمندی های به اندازه کافی خوب را ارائه دهند (معنی خوب به اندازه کافی دقیق (Detailed enough) است اما خیلی دقیق نیست - با توجه به تعریف آماده (Defintion of Ready)) تا روند توسعه را با سرعت ثابت ادامه دهد. این مورد، در ترکیب با نیازمندی های در حال تغییر مداوم، نیاز به جریان تقاضای مداوم (Continuous demand flow) دارد.
2. بخش منابع انسانی باید درک کند که چه افرادی را برای پشتیبانی صحیح از تیم های چابک استخدام کند. می توان نمونه های بدی از پیشنهادات شغلی را برای اسکرام مستر یافت که در آن بر داشتن مهارت برنامه نویسی و گواهینامه تاکید شده است که این نشان دهنده آن است که اصول سه نقش اسکرام به درستی درک نشده اند.

در 4.2 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 لزوماً از اصول چابک پیروی نکنند.

تأثیر در سازمان در مواردی که بیش از یک تیم چابک برای حل یک مسئله پیچیده مورد نیاز است، در 4.3 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 کاملاً چابک در اینجا به صراحت مورد توجه قرار نگرفته است، زیرا از حوصله بحث در مورد RE خارج است.

4.4 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.

4.2 توسعه چابک در محیط غیر چابک (L1)

4.2.1 تعامل با ذی نفعان 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 می دهد. این بدان معنا است که مشتری در کل چرخه تولید محصول درگیر است، جایی که بازخورد تحویل تدریجی (Incremental deliveries) به طور فعالانه درخواست می شود و نیازمندی های جدید مطابق با نیازمندی های کسب و کار را برآورده می کند.

با مشارکت مداوم مشتری، RE نیز به یک روند مستمر تبدیل می شود (2.4 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 یک 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 می تواند ماهیتی مستقیم و غیررسمی داشته باشد، جایی که فقط نتایج یا تصمیمات ثبت می شوند.

شناخت عوامل خارج از کنترل توسعه دهندگان (به عنوان مثال، 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 موثرتر، یا با دعوت از نمایندگان مشتری به جلسات برنامه ریزی و بررسی منظم، یا با انجام جلسات فشرده با زمان ثابت (Time-boxed)، (برای مثال تفکر دیزاین (Design Thinking) یا دیزاین اسپرینت (Design Sprint)، 1.5 را مشاهده کنید) در مرحله اولیه با ورودی ضبط شده در یک لاگ محصول، برنامه ریزی شود.

4.2.2 سازمان مبتنی بر محصول در مقابل سازمان مبتنی بر پروژه

چابک در طول زمان، روابط مستقیم، باز و غیر سلسله مراتبی درون سازمانی و انعطاف پذیری را دقیقاً در تکامل محصولات، ارتقا می بخشد. سازمانهای بزرگتر، به طور سنتی بر اساس ساختارهای مدیریتی بالا به پایین (Top-down management)، برای برنامه ریزی (Planning) و قابلیت پیش بینی (Predictability)، اهمیت زیادی قائل هستند. به عنوان مثال، برنامه ریزی پروژه و منابع، واقعیت هایی هستند که سازمان ها نمی توانند به راحتی از آن ها منصرف شوند.

به طور سنتی، توسعه نرم افزار اغلب مبتنی بر پروژه بوده است، به این معنی که به عنوان مجموعه ای از تعهدات موقت برای تولید محصولات، خدمات یا نتایج منحصر به فرد انجام می شود ([PMI] را مشاهده کنید). گروه هایی از پروژه های مرتبط با اهداف مشترک غالباً برنامه (Program) نامیده می شوند، در حالی که برنامه ریزی و کنترل کل پروژه ها و برنامه ها در داخل سازمان به عنوان مدیریت پورتفولیو (Portfolio Management) نامیده می شود.

رویکرد چابک که به ریشه های خود در توسعه محصول وابسته است، بیشتر محصول محور است. بک لاگی از پیشرفت های محصول (Product improvements) حفظ می شود و این موارد، در یک فرایند تکرار شونده بهبود مستمر اجرا می شوند چابک به تنهایی تاریخ پایان را به این ترتیب تعریف نمی کند. اصولاً تا زمانی که پیشرفت هایی باید انجام شود و یا منافع قابل تحقق باشند، در صورتی که منافع بیشتر از تلاش/هزینه ها (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 پروژه های اضافی نیز برای بهبودهای بعدی ایجاد شده اند) اما اختلاف دیدگاه و اصطلاحات ممکن است منشا تنش و سوء تفاهم میان توسعه نرم افزار چابک و سازمان های غیر چابک شوند.

بیک روش برای حل چنین تنش هایی این است که در حالی که توسعه نرم افزار به صورت کاملاً چابک انجام می شود، عملکرد مدیریت پورتفولیو و مدیریت برنامه، سطح بالاتری از برنامه ریزی و کنترل را فراهم کنند که از رویکردهای هر دو جهان استفاده می کند (4.3 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 نیازمندی ها، مفاهیم و روش های لازم را برای تمایز میان نیازمندی های موجود در این سطوح مختلف انتزاع، با نیازمندی های سطح کسب و کار در پورتفولیو و برنامه ها و با نیازمندی های نرم افزاری مشتق شده و با جزئیات مناسب برای پیشبرد توسعه، فراهم می کند. رویکرد نیازمندی ها از رویکرد با جزئیات که به عنوان دستوراتی دقیق برای توسعه استفاده می شود، به نیازمندی هایی تغییر می یابد که به عنوان یک مبنای مشترک برای بحث و هماهنگی به موقع و به همان اندازه که برای سطح فعلی برنامه ریزی لازم است، مورد استفاده قرار می گیرد.

4.2.3 نقش مدیریت در زمینه چابک

دیسپلین ها و تیم ها (Disciplines & Teams): کارمندان بخش های فناوری اطلاعات به طور سنتی توسط دیسپلین سازماندهی می شدند: توسعه دهندگان، آزمون کنندگان، مهندسان نیازمندی ها، تحلیلگران کسب و کار، مدیران پروژه و غیره. تیم های پروژه متشکل از این مهارت های مختلف، برای مدت زمان مشخصی از انجام کار، گرد هم می آمدند. چابک و به ویژه اسکرام، ایده تیم های چند عملکردی (Cross-functional) بیشتر را ترویج می دهند. جدا از نقش های تخصصی مالک محصول و اسکرام مستر، تمامی اعضای تیم باید توانایی بازی در ظرفیت های مختلف را داشته باشند، به طوری که هر فرد در صورت نیاز، از فعالیت های مهندسی نیازمندی ها یا فعالیت های آزمون پشتیبانی کند همچنین به همین دلیل، این اعضای تیم معمولاً در اسکرام "توسعه دهندگان" نامیده می شوند. هدف این تیم این است که بتواند نیازمندی های مشتری را به طور کامل و بدون در نظر گرفتن عناصر فنی یا سازمانی ارائه دهد. این را می توان با داشتن صلاحیت های RE در تیم توسعه (راه حل ترجیحی) یا خارج از توسعه دهندگان (که در واقع اغلب به عنوان پشتیبانی از مالک محصول استفاده می شود) 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 [Conway's Law] [Conw1968]، 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

تیم های چند عملکردی که همه قابلیت ها برای انتقال کل مراحل از فرانت اند (Frontend) به بک اند (Backend) را به ارث می برند، می توانند به راحتی مقیاس پذیر شوند - اما ایجاد آن ها آسان نیست و کاری وقت گیر است.

مالک محصول در مقابل مدیر پروژه: همانگونه که در قبل ذکر شد (را مشاهده کنید)، مالکان محصولات با بر عهده گرفتن مسئولیت اولویت های کاری در تیم های توسعه محصول، مسئولیت مهندسی نیازمندی ها را گسترش می دهند. مالکان محصول باید فعال بوده (دانش) و توان تصمیم گیری در زمینه کسب و کار را داشته باشند. برخی از تصمیماتی که قبلاً توسط مدیران پروژه گرفته شده بود، دیگر در کار با رویکردهای چابک مورد نیاز نیست، زیرا توسعه دهندگان در محدوده سازمان خود، خودسازمانده می شوند و تنها نیاز به کاری جهت انجام دادن دارند (نیازمندی هایی در سطحی از جزئیات که امکان توسعه در یک تکرار را فراهم می کنند) تا کار خود را (تجزیه تسک و تعیین تکلیف در تیم) سازماندهی کنند.

آنچه از مدیران پروژه چابک IT مورد انتظار است، تعیین روشن چشم انداز توسعه چابک و ارتباط واضح پیش شرط های فرهنگی برای موفقیت این رویکرد است.

رسیدن به تعادل، درست در حین برآورده کردن انتظارات از کسب و کار، موضوع ساده ای نیست. برخی از معیارهای موفقیت در زیر در 4.4 مورد بحث و واقع شده است.

مرتبط بودن مهندسی نیازمندی ها (RE): الگوهای قبلی به عنوان مثال برای اسکرام نیز اعمال می شود. افرادی که بر اساس اصول RE کار می کنند، می توانند بخشی از تیم باشند و بنابراین به طور جداگانه نامی از آنها برده نمی شود. از سوی دیگر، می توانند خودشان تیمی را تشکیل دهند که از یک یا چند مالک محصول، در قالب یک تیم پشتیبانی می کند. هر دو روش مزایای خود را دارند و می توانند بدون نقض اصول چابک به صورت ترکیبی مورد استفاده قرار گیرند. RE به ستون فقرات توسعه موفق چابک تبدیل خواهد شد.

4.3 رسیدگی به مشکلات پیچیده با مقیاس پذیری (L1)

4.3.1 انگیزه مقیاس پذیری

ارزش های چابک به طور معمول نمایانگر ارتباط مستقیم، روزانه، غیر سلسله مراتبی توسط تیم های کوچک و نزدیک به هم مانند تیم اسکرام، با تعداد توصیه شده کمتر از 10 نفر (توسعه دهندگان + مالک محصول + اسکرام منستر) هستند. اعضای تیم در حالت ایده آل باید از نظر از نظر فیزیکی در یک مکان بوده و از نظر مهارت های 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 بخش های مختلف کسب و کار باشد که به راحتی در یک تیم واحد جای نمی گیرد.
- مشکلات پیچیده ممکن است شامل طیف وسیعی از متخصصان فنی و دانشی باشد که به راحتی در یک تیم واحد جای نمی گیرد.
- دامنه موردنیاز برای یک تاریخ عرضه (Rollout date) مشخص، فراتر از سرعت قابل دستیابی یک تیم واحد است.
- کارمندان در مشاغل جهانی ممکن است از نظر جغرافیایی توزیع شوند.

اصطلاح چابک مقیاس پذیر (Scaled Agile) برای توصیف شرایطی استفاده می شود که چندین تیم لازم است با هم روی یک محصول / راه حل مشترک اهداف مشترک کار کنند. رویکردهای چابک مقیاس پذیر نیاز به تصمیم گیری در مورد چگونگی سازماندهی تیم ها و چگونگی هماهنگی ارتباط بین تیم ها دارند. هدف، در حین حفظ مزایای چابکی، دستیابی به یک رویکرد مؤثر برای رسیدگی به مشکلات پیچیده است.

4.3.2 رویکردهایی برای سازماندهی تیم ها

سؤالی که در اینجا باید پاسخ داده شود این است که سازمان چگونه می تواند با توجه به ارتباط و همسویی در عین مؤثر بودن (حداکثر اندازه)، خود را در قالب تیم هایی با اندازه مناسب قرار دهد و به آن ها اجازه چندعملکردی بودن دهد. سازماندهی در امتداد خطوط عملکردی (به عنوان مثال یک تیم متخصص در یک حوزه کسب و کاری مشخص، یا حتی در مورد ویژگی در یک حوزه کسب و کار) این مزیت را دارد که دانش کسب و کار را در یک تیم واحد متمرکز می کند. به عنوان مثال، این مورد ممکن است به تسهیل در استخراج نیازمندی ها از طریق کاهش تعداد ذی نفعانی که به صورت مستقیم با تیم ارتباط دارند، منجر شود.

یک نقطه ضعف این رویکرد این است که ارائه قابلیت های انتها به انتها (End-to-End) در یک حوزه کسب و کار، احتمالاً شامل چندین تخصص فنی مختلف مانند طراحی رابط کاربری، موتورهای پردازش، پایگاه داده و پلتفرم های اصلی مانند ERP یا پردازنده های مرکزی (Mainframes) اصلی است که به راحتی می توانند بیش از ظرفیت تیم، فشار وارد کنند.

یک روش جایگزین برای سازماندهی توسعه، خطوط فنی (Technical lines) است و تیم های متخصص در مولفه های فنی (Technical components) یا پلتفرم ها فعالیت می کنند. مزیت تیم های مولفه یا پلتفرم، دانش عمیق در زمینه فناوری مربوطه است. نقطه ضعف، وابستگی هایی است که چنین رویکردی بین تیم هایی که با هم کار می کنند ایجاد می کند تا کل اینکریمنت محصول را طبق برنامه (Schedule) ارائه دهند.

چارچوب های مقیاس پذیر (مانند آن هایی که در **Fehler! Verweisquelle konnte nicht gefunden werden.** 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 به ویژه نقش اساسی در تجزیه اهداف و نیازمندی های کسب و کار به نیازمندی های زیر سیستم تشکیل دهنده دارد که می تواند به تیم های جداگانه اختصاص داده شود و ثانیاً به دنبال توسعه از طریق اطمینان از نتیجه گیری یک راهکار یکپارچه و اطمینان از تحویل ارزش کسب و کار است.

ممکن است ترکیبی از انواع تیم ها با در نظر گرفتن محدودیت های خاص شرکت، منجر به ارائه بهترین و عملی ترین راهکار برای بهره مندی از ایده های تیم های برجسته شود (برای جزئیات بیشتر به [CPREALAGILE2022] مراجعه کنید).

4.3.3 رویکردهایی برای سازماندهی ارتباط

در پاسخ به این سوال که چند تیم می توانند به طور کارآمد با هم کار کنند، می توان بین دو رویکرد مختلف تمایز قائل شد:

1. استفاده از رویکردهای تیم واحد (Single team approaches)
2. معرفی مفاهیم اضافی برای سازماندهی ارتباطات و مسئولیت ها (Communication and responsibilities)

استفاده از رویکردهای تیم واحد: رویکرد اول از این ایده پیروی می کند که محصولات کاری و نقش های اضافی مورد نیاز نیستند و ارتباط بین تعدادی از تیم ها باید با تکنیک های موجود از رویکردهای تیم واحد پشتیبانی شود. نقش ها و محصولات کاری اضافی، مغایر تفکر چابک بوده و منجر به پیچیدگی بیشتری در سازمان می شود. با توجه به اصل اساسی "ساده نگه دارید"، هیچگونه هزینه اضافی بر اساس نقش های جدید ایجاد نمی شود. ارتباط و هماهنگی بین تیم ها معمولاً توسط مالکان محصول تیم ها آغاز می شود اما توسط نمایندگان تیم (Team delegates) انجام می شود. ساختارهایی نظیر انجمن های پرکنیس (Communities of Practice)، در سراسر تیم اعضا را قادر می سازد تا تجربیات خود را به اشتراک بگذارند و فرایندهای کلی را هماهنگ کنند.

معرفی مفاهیم اضافی: روش دوم توصیه می کند که مشکلات بزرگتر را به مشکلات کوچکتر تقسیم کنید و مسئولیت ها را بر اساس نقش های مختلف برای انتزاع های مختلف مدیریت کنید. بنابراین باید محصولات کاری اضافی برای سطوح مختلف انتزاع معرفی شوند (به عنوان مثال اپیک های تجاری، اپیک های معماری، تیم های سرمایه گذاری، ویژگی ها، داستان های کاربر).

بسته به چارچوب مورد استفاده، نقش های دیگر مسئول در سطوح مختلف انتزاع معرفی می شوند (به عنوان مثال مدیران پورتفولیو، مدیران محصول و مالکان محصول ارشد). به دلیل پیچیدگی روز افزون، مصنوعات و نقش های اضافی برای مدیریت برنامه ریزی و ارتباط بین تیم های مختلف و دستیابی به نتایج یکپارچه محصولات کاری هر تکرار (به عنوان مثال نقشه های راه و مدیران انتشار) مورد نیاز است. علاوه بر این، برای ارتقای ارتباط میان نقش های جدید و تاسیس شده، جلسات خاصی باید تشکیل شود. RE به ارائه روش های زیادی می پردازد که می تواند به تقسیم مشکلات بزرگتر و حمایت از نقش های جدید در سطوح مختلف انتزاع (به عنوان مثال مدل سازی زمینه (Context modeling)، مدل سازی 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 کسب و کار (Business Case) پیدا کند.

4.3.4 چارچوب های نمونه برای مقیاس پذیری RE@Agile

چارچوب هایی که از مقیاس پذیری اسکرام و چابک پشتیبانی می کنند: چارچوب های مختلفی وجود دارند که از این رویکردها پشتیبانی می کنند و به دلیل اهمیت روز افزون مقیاس پذیری، این تعداد نیز به سرعت در حال رشد است. مجموعه ای از معروف ترین چارچوب ها را در زیر خواهید یافت:

- **چارچوب چابک مقیاس پذیر (SAFe) (Scaled Agile Framework)**
SAFe یک پایگاه دانش از الگوهای موفق اثبات شده برای پیاده سازی نرم افزاری ناب-چابک و توسعه سیستم در مقیاس سازمانی است. [SAFe]
- **اسکرام مقیاس-بزرگ (LeSS) (Large-Scale Scrum)**
LeSS در واقع استفاده از اسکرام برای مجموعه ای از تیم ها است که به همراه یکدیگر با یک مالک محصول، بر روی یک محصول کار می کنند. [Less]
- **نکس (Nexus)**
Nexus چارچوبی است که به قلب مقیاس پذیری می رود: وابستگی های تیمی متقابل (Cross-team dependencies) و مسائل ادغام [Nexus2015]. [Integration issues]
- **اسکرام در مقیاس (Scrum@Scale)**
Scrum of Scrums تکنیکی است که در آن از اسکرام برای هماهنگ کردن چندین تیم استفاده می شود. [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 [CPREALAGILE2022].
- **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 می دهد تا یک پیاده سازی اسکرام متناسب با نیازهای منحصر به فرد شرکت شما مقیاس پذیر شود. [SatS]

4.3.5 تأثیرات مقیاس پذیری در RE@Agile

لایه های انتزاعی که در بالا مورد بحث قرار گرفتند به این معنی هستند که فعالیت های RE توسط نقش های بیشتری مانند مالک محصول ارشد، مدیر محصول، مدیر پورتفولیو و تحلیلگر کسب و کار مدیریت می شوند. هر نقشی محصولات کاری RE مربوطه را برای حوزه مورد نظر خود ایجاد می کند (اپیک، ویژگی ها، داستان های کاربر و قابلیت ردیابی (Traceability) بین آن ها).

درحالی‌که ارتباط با ذی‌نفعان با نقش‌های مختلف در سطوح مختلف درون سازمان انجام می‌شود، جلساتی اضافی برای فعالیت‌های RE موردنیاز خواهد بود (به‌عنوان مثال زمان‌های داستان (Story times)، زمان‌های دمو (Demo times)، دمو سیستم (System demo)).

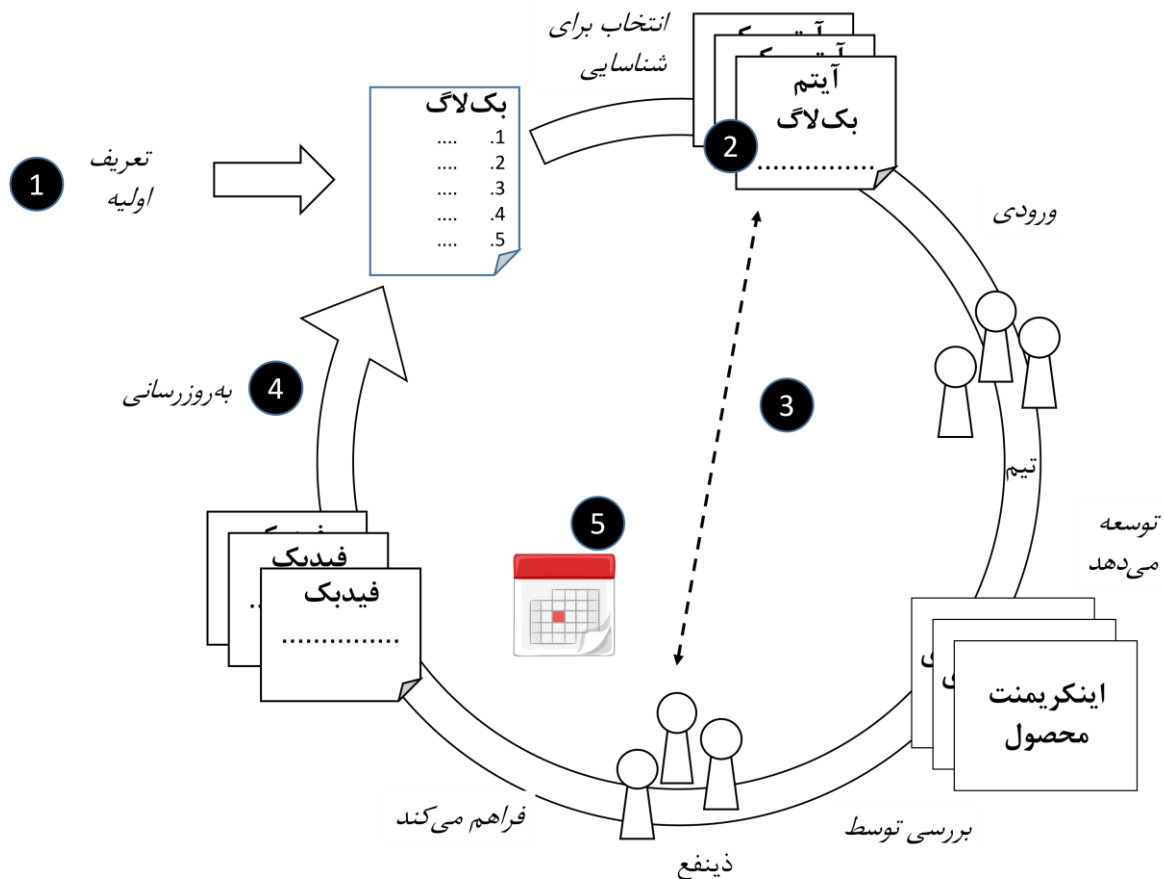
از آنجایی که اسکرام خود به راحتی مقیاس پذیر نمی‌شود، RE نقش مهمی در تعیین چگونگی تجزیه و تقسیم نیازمندی‌های اساسی بین چندین تیم چابک برای زنده نگه داشتن هر روش مقیاس بندی بازی می‌کند. تکنیک‌های RE در سازماندهی تجزیه و تحلیل مسئله و اصلاح نیازمندی‌های درشت دانه (Coarse-grained requirements) به نیازمندی‌های 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 نیمه مستقل چابک فراهم می‌کند.

این امر به ویژه در مورد وابستگی‌هایی که باید در اسرع وقت پیدا و مورد بحث 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 شیفیت زمان ممکن یا زبان‌های مختلف افزایش نمی‌یابد. در بیشتر موارد، چالش اصلی بر اساس فرهنگ‌های مختلف است. از آنجا که ارتباطات یکی از وظایف اصلی مالکان محصول و مدیران محصول است، این موضوع تأثیر مهمی در مهارت‌های مورد نیاز نقش‌های مربوط به RE دارد.

4.4 متعادل سازی مهندسی نیازمندی‌های مقدماتی و مداوم در زمینه مقیاس پذیری (L1)

در سطح کاملاً انتزاعی، روش‌های چابک را می‌توان به عنوان یک فرایند تکرار شونده و مداوم توصیف کرد که در آن سیستم به تدریج بر اساس موارد بک لاگ 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

4.4.1 تعریف اولیه نیازمندی ها

قبل از شروع فرایند توسعه مداوم، باید یک بک لاگ اولیه ایجاد شود ([TaNo1986] را مشاهده کنید). این تعریف اولیه از بک لاگ اغلب به عنوان "تعریف مقدماتی" (Upfront definition) شناخته می شود. گاهی اوقات مهندسی نیازمندی ها به این صورت درک می شود که این تعریف اولیه، مشخصات کامل و دقیق کلیه نیازمندی ها است. ممکن است این مورد برای برخی از مدل ها یا دامنه های فرآیندی باشد، اما همیشه اینطور نیست و در توسعه Agile نسبتاً غیر معمول است.

مینای تعریف اولیه بک لاگ، اطلاعاتی است که برای شروع تکرار اول فرایند توسعه مداوم مورد نیاز 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 نیازمندی ها (شروع بدون دانش کافی) ممکن است به تأخیرهای اضافی، هزینه ها یا به طور بالقوه به شکست پروژه منجر شود. برعکس، شروع خیلی دیر ممکن است تأثیر منفی بر زمان بازاریابی و مناسب بودن نیازهای نهایی کاربران در یک زمان خاص داشته باشد.

مهندسی نیازمندی ها به ما می گوید که باید آن دسته از نیازمندی هایی که می تواند تأثیر زیادی بر معماری، امکان سنجی کلی راحل و یا گزینه های اصلی مربوط به زیرساخت ها و سخت افزارها داشته باشد، شناسایی کرد و هر چه زودتر آن ها را تشریح کرد. چنین مسائلی مربوط به معماری در حقیقت، باید قبل از تکرار اول توسعه استخراج و تحلیل شود. نیازمندی های با اثر کمتر (Lower impact) ممکن است در طول تکرارها، در قالب یک فرایند مداوم مورد اصلاح واقع شوند.

در فرایندهای تکرار شونده و تدریجی چابک، مهندسی نیازمندی ها به یک فرایند مداوم تبدیل می شود که نیازمندی ها را به موقع و با جزئیات کافی برای تغذیه چرخه توسعه ارائه می دهد. این فرایند به یک قیف (Funnel) خرد کردن سنگ شباهت دارد، جایی که

سنگ های بزرگ وارد کیف می شوند و تا زمان اتمام فرایند به اندازه های متوسط خرد می شوند و در نهایت به سنگ های کوچک خرد می شوند.

4.4.2 سطح جزئیات برای آیتم های بک لاگ

سطح جزئیات یک آیتم بک لاگ، آزادی توسعه دهندگان را برای تحقق آیتم بک لاگ محدود می کند ([Pich2010] را مشاهده کنید). تمام جنبه هایی از آیتم بک لاگ که هنوز تعریف نشده اند، به عنوان تصمیماتی برای تیم باقی می ماند که باید به آن اجازه دهد تا در مرزهای مشخص شده کسب و کار، خلاقیت بیشتری داشته باشد.

صلاحیت های موجود توسعه دهندگان می تواند به عنوان یک اصل کلی عمل کند. در صورتی که توسعه دهندگان صلاحیت و تجربه کافی برای تصمیم گیری در مورد جزئیات یک آیتم بک لاگ را دارد (به عنوان مثال، یک متخصص احراز هویت (Authentication) و مجوز (Authorization) در میان توسعه دهندگان است)، تصمیم گیری در مورد جزئیات باید به توسعه دهندگان سپرده شود. همچنین **Fehler! Verweisquelle konnte nicht gefunden werden.** را مشاهده کنید.

4.4.3 اعتبار آیتم های بک لاگ

دانستن اعتبار (Validity) یک آیتم بک لاگ قبل از اجرا، به کاهش کارهای غیرضروری پیاده سازی کمک می کند ([Denn2015] را مشاهده کنید). انتظار برای تشخیص یک نیازمندی غلط یا ناقص در نرم افزار پیاده سازی شده، یک رویکرد بسیار گران قیمت برای اعتبارسنجی نیازمندی ها است.

تعیین اعتبار یک آیتم بک لاگ، ارتباط نزدیکی با سطح جزئیات آن آیتم مدنظر دارد. اعتبار یک آیتم بک لاگ محصول فقط زمانی مشخص می شود که آیتم بک لاگ 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 نرم افزار تحویل شده مقایسه شود.

وقتی می توان اولویت ذینفع مربوط به یک آیتم بک لاگ را با تلاشی قابل قبول تعیین کرد، تأیید چنین شرایطی باید قبل از توسعه آیتم بک لاگ انجام شود.

نمونه های رایج این نوع نیازمندی (از جمله رویکرد اعتبارسنجی نمونه ای (Exemplary validation approach)) عبارتند از: طراحی کلی رابط کاربری (به عنوان مثال با موکاپ های رابط کاربر (UI mockups))، سازوکارهای مجوز (Authorization) و تأیید اعتبار (Authentication) (به عنوان مثال با استفاده از بررسی موارد کاربرد (Use case reviews))، ساختارهای داده ای که باید در سیستم (به عنوان مثال با بررسی مدل داده) و نیازمندی های رابط های سیستم های موجود (به عنوان مثال با بررسی نمودار فعالیت (Activity diagram)).

آیتم های بک لاگی که دارای ریسک بالایی هستند (به عنوان مثال عملکردهای حیاتی کسب و کار (Critical business functionalities))، عملکردهای حیاتی ایمنی (Safety critical functions)، عملکردهای نوآورانه (Innovative functionalities)) یا آیتم هایی که هزینه تست بالایی برای اجرا دارند (به عنوان مثال نرم افزار باید در نمونه اولیه ای گران قیمت مورد آزمون واقع شود)، باید قبل از اجرا، اعتبارسنجی شوند.

4.4.4 بازخورد و به روزرسانی بک لاگ

بک لاگ ها اغلب براساس بازخورد از یک فعالیت بازرسی (Inspection activity) مانند بررسی اسپرینت در اسکرام، به روز می شوند. چنین رویکردی برای مقیاس های کوچک یا نیازمندی های جزئی امکان پذیر است که می توان در یک محیط با یک یا دو تیم توسعه، تأثیر یک تغییر را تحلیل و درک کرد. اصلاح نیازمندی هایی که از پیچیدگی بیشتری برخوردار هستند و یا وابستگی های متعددی در کوتاه مدت دارند، توصیه نمی شود. در چنین شرایطی که اصلاح آیتم های بک لاگ زمان بیشتری می برد، ممکن است ذینفعان در دسترس نباشند و تجزیه و تحلیل اضافی لازم باشد.

عامل دیگری که ممکن است در تغییر آیتم های بک لاگ تأثیر بگذارد، فرایند تصمیم گیری سازمان است. در سازمان هایی که گرفتن تصمیمات مهم ممکن است مدتی طول بکشد (به عنوان مثال شورای مسئول فقط هر سه ماه یکبار تشکیل جلسه می دهد) ، اصل پالایش مداوم (Continuous refinement) باید به شکل جلسات مشخصی با مشارکت همه ذینفعان تأثیرگذار باشد. چنین جلساتی، به مهندسی نیازمندی ها برای آماده سازی و پشتیبانی از تصمیم گیری نیاز دارند.

4.4.5 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 فعالیت های مهندسی نیازمندی های دارد که باید هنگام کار روی مواردی که هنوز در دست توسعه نیستند، اجرا شوند. علاوه بر این، طول تکرار، توالی تحویل نتایج به ذینفعان کسب و کار یا مشتریان در دسترس را برای بررسی تعیین می کند. در نتیجه، تکرارهای کوتاه تر به سه دلیل باعث افزایش حجم کاری ذی نفعان کسب و کار می شوند (برای مثال مقایسه کنید با [Rein1997])

1. ذی نفعان کسب و کار باید در کل تکرار برای کار بر روی آیتم های بک لاگ در دسترس باشند تا ورودی برای تکرار بعدی ایجاد کنند.
2. ذی نفعان کسب و کار باید نتایج ایجاد شده توسط تیم را برای ارائه بازخورد بررسی کنند.
3. ذی نفعان کسب و کار از تعویض های زمینه (Context switches) بین کسب و کار روزانه (Daily business) و کار پروژه (Project work) رنج می برند.

با این که طول های تکرار بیشتر (Longer iteration lengths) موجب کاهش فشار می شوند اما موجب کاهش توانایی تأثیر (Capability of influencing) بک لاگ برای تولید محصول می شوند.

تعریف طول تکرار باید با در نظر داشتن ذی نفعان کسب و کار انجام شود. ذی نفعان کسب و کار به طور معمول به صورت 100٪ برای فعالیت های توسعه در دسترس نیستند، زیرا آن ها وظایف دیگری در سازمانی دارند که سیستم برای آن توسعه یافته است.

به عنوان یک قاعده کلی، زمان تکرار کوتاه تر بازخورد مکرر و فرصت های بیشتری برای کشف زود هنگام خطاها را فراهم می کند، بنابراین تکرارهای کوتاه تر سرعت فعالیت های توسعه را افزایش می دهند. اگر زمان چرخه کوتاه بار غیرقابل قبولی 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 می تواند در زمان چرخه (Cycle time) تأثیر بگذارد، متوسط اندازه و پیچیدگی آیتم های بک لاگ است. آیتم های بک لاگ بزرگ تر یا پیچیده تر، زمان بیشتری را برای درک و تجزیه و تحلیل صرف می کنند. بنابراین زمان چرخه درجهت رسیدگی به آیتم های بزرگتر یا پیچیده تر، می تواند افزایش پیدا کند. به عنوان مثال، اگر سیستم در حال توسعه در مراحل اولیه باشد، ممکن است یک چرخه طولانی تر توصیه شود تا تیم به زمان بیشتری برای درک اولیه سیستم برسد. با این وجود، این عامل باید با هدف استفاده از زمان های چرخه کوتاه تر برای دریافت بازخورد مکرر متعادل شود. تصمیم گیری در مورد داشتن زمان های چرخه طولانی تر یا کوتاه تر باید با هم به عنوان تیمی انجام شود که نیاز به تجزیه و تحلیل با هدف بازخورد زودرس را ارزیابی می کند. باتوجه به 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 امکان پذیر خواهد بود. علاوه بر این، زمان چرخه باید تا حد امکان سازگار باشد تا پیچیدگی کاهش یابد و قابلیت پیش بینی پایدار ایجاد شود. تغییر بیش از حد مدت زمان ممکن است باعث بی ثباتی، کاهش تعهد تیم و به احتمال زیاد مانع از سرعت افزایش محصول شود.

5 تعاریف اصطلاحات، واژه نامه (L2)

واژه نامه CPRE و راهنمای اسکرام (SCRUM Guide) اصطلاحاتی را تعریف می‌کنند که در زمینه RE@Agile Primer مرتبط هستند.

- واژه نامه CPRE برای دانلود در وبسایت IREB در دسترس است: <https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-glossary>
- راهنمای اسکرام (SCRUM Guide) برای دانلود در دسترس است: <https://scrumguides.org/download.html>

- [ACP/PMI] Agile Certified Practitioner: <https://www.pmi.org/certifications/agile-acp>. Last visited June 2026
- [AgileMan2001] :Agile Manifesto<http://agilemanifesto.org/>.Last visited June 2026 .2001 ,
- [AmLi2012]Ambler S.; Lines, M.: Disciplined Agile Delivery: A Practitioner’s Guide to Agile Software Delivery in the Enterprise. IBM Press, 2012
- [Ande2012] Anderson, D.J.: Lessons in Agile Management: On the Road to Kanban. Blue Hole Press, 2012
- [Appel2011] Appelo J.: Management 3.0. Addison-Wesley Professional, 2011
- [Beck2003] Beck, K.: Test-Driven Development by Example. Addison Wesley – Vaseem, 2003
- [Beck2004] Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, 2004
- [CNYM2000] Chung, L.; Nixon, B. A.; Yu, E.; Mylopoulos, J. : Non-Functional Requirements in Software Engineering. Springer Science & Business Media, 2000
- [Cock1998] Cockburn, A.: Surviving Object-Oriented Projects. Addison-Wesley, 1998
- [Cohn2004] Cohn, M.: User Stories Applied: For Agile Software Development. Addison Wesley Professional, 2004
- [Conw1968] Conway, M.: How Do Committees Invent? Datamation 14(4):28–31, 1968. Article available at http://www.melconway.com/Home/Conways_Law.html. Last visited June 2026
- [CPREFL2022] IREB e.V.: Syllabus CPRE Foundation Level<https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-foundation-level-syllabus>.Last visited June 2026 .
- ,[CPREALAGILE2022] IREB e.V.: Syllabus CPRE Practitioner | Specialist <https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-re-agile-syllabus>, version 2.3 .Last visited June 2026 .
- [Denn2015] .Denning, S.: How To Make The Whole Organization Agile <http://www.forbes.com/sites/stevedenning/2015/07/22/how-to-make-the-whole-organization-agile/#658d3f65135b>, 2015. Last visited June 2026
- [Dsch2015] .d.school: An Introduction to Design Thinking – Process Guide <https://web.stanford.edu/~mshanks/MichaelShanks/files/509554.pdf>, 2015. Last visited June 2026

- [Glinz2026] ,Glinz, M.: A Glossary of Requirements Engineering Terminology <https://cpire.ireb.org/en/downloads-and-resources/downloads#cpire-glossary> Last visited June 2026
- [Griff2015] Griffiths, M.: PMI-ACP Exam Prep. Rmc Publications, 2015
- [GoAk2003] Gordijn, J.; Akkermans, J.M.: Value-based Requirements Engineering: exploring innovative e-commerce ideas. Springer, 2003
- [High2009] Highsmith, J.: Agile Project Management: Creating Innovative Products. Addison-Wesley Professional, 2009
- [ISO25010] ISO/IEC Systems and software engineering - Systems and software Quality Requirements and Evaluation. ISO/IEC Standard 25010:2011
- [KnZK2016] Knapp, J.; Zeratsky, J; Kowitz, B.: Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days. Simon & Schuster, 2016
- [Kron2008] Kronfaelt, R.: Ready-ready: the Definition of Ready for User Stories going into Sprint planning. <https://scrumftw.blogspot.com/2008/10/ready-ready-definition-of-ready-for.html>, 2008. Last visited June 2026
- [Less] Large Scale Scrum: <https://less.works/resources/LeSS-brochure.pdf> Last visited June 2026
- [LiOg2011] Liedtka, J.; Ogilvie, T.: Designing for Growth: A Design Thinking Tool Kit For Managers. Columbia Business School Publishing, 2011
- [Martin1991] Martin, J.: Rapid Application Development. Macmillan Coll Div, 1991
- [Meyer2014] Meyer, B.: Agile! – the good, the hype, and the ugly. Springer, 2014
- [MeMi2015] Mesaglio, M., Mingay, S.: Bimodal IT: How to Be Digitally Agile Without Making a Mess, Gartner 2015, <https://www.gartner.com/en/documents/2798217>. Last visited June 2026
- [Nexus2015] Nexus Guide <https://www.scrum.org/resources/nexus-guide>,. Last visited June 2026
- [Patt2014] Patton, J.: User Story Mapping. O'Reilly, 2014
- [Pich2010] .Pichler, R: Make the product backlog deep <http://www.romanpichler.com/blog/make-the-product-backlog-deep/>,. Last visited June 2026
- [PMI] .PMI Project Management Institute. <http://www.pmi.org/>. Last visited June 2026
- [Popp2003] Poppendieck, M.: Lean Software Development: An Agile Toolkit. Addison-Wesley Professional, 2003
- [Rein1997] .Reinerstsen, D. G.: Managing the Design Factory – A Product Developer's Toolkit Simon & Schuster, 1997

- [Ries2011] Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Publishing Group 2011
- [SAFe] SAFe – Scaled Agile Framework. <http://www.scaledagileframework.com>. Last visited June 2026
- [SatS] ./Scrum@Scale framework: <https://www.scrumatscale.com/scrum-at-scale-guide>. Last visited June 2026
- [Scrum2020] :Schwaber, K. & Sutherland, J.: The Scrum Guide: The Definitive Guide to Scrum The Rules of the Game, July 2020. <https://scrumguides.org/download.html>.. Last visited June 2026
- [ShYo2006] Sheppard, J. M.; Young W. B.: Agility literature review: Classifications, training and testing. Journal of Sports Sciences 24(9): 919-932, 2006
- [SofS] ,Scrum of Scrums <https://scrumguide.de/scrum-of-scrums>. Last visited June 2026
.available in German only
- [TaNo1986] Takeuchi, H.; Nonaka, I.: The new new product development game. Harvard Business Review 64(1), January/February 1986, p.137-146
- [Wake2003] ,Wake, B.: Invest in Good Stories and Smart Tasks
<http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>. Last visited June 2026