به نام خدا

متدولوژی‌های ایجاد نرم‌افزار

استاد: دکتر رامان رامسین

تمرین اول

حسین بهبودی

۴۰۲۲۱۱۵۲۹

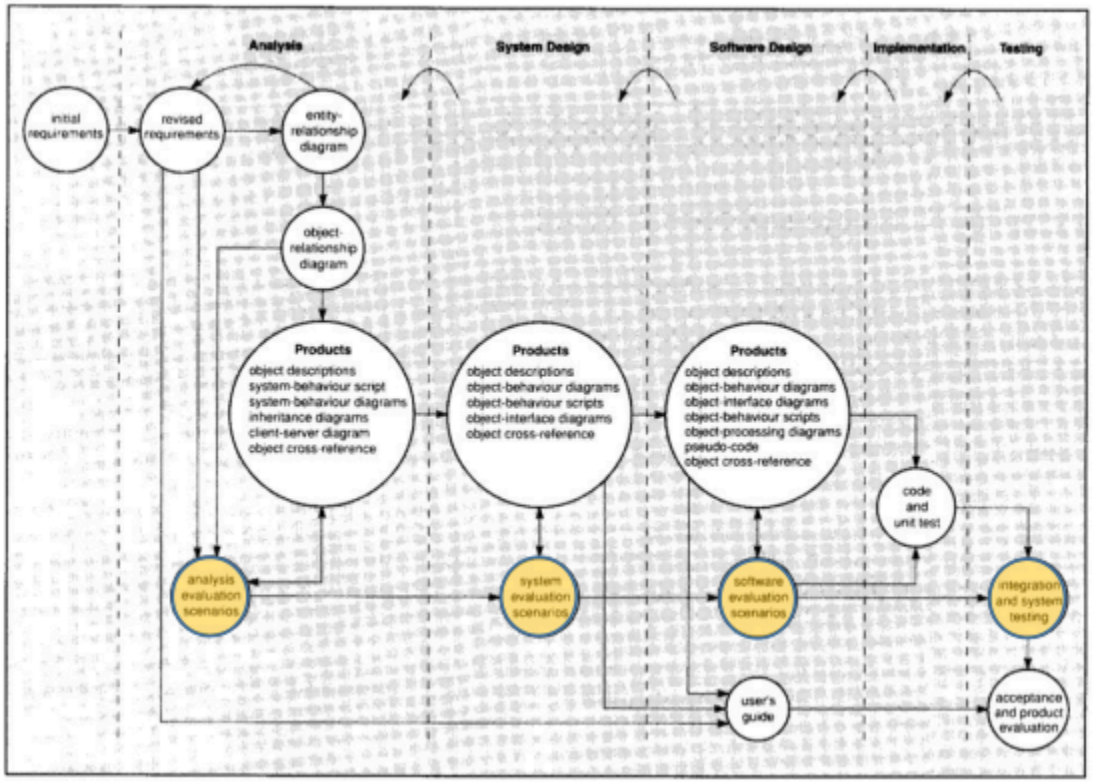
زمستان ۱۴۰۴

فهرست محتوا

مروری کوتاه بر متدولوژی‌های مورد بحث در گزارش

متدولوژی Hodge-Mock

فرآیند متدولوژی: معرفی اجمالی فازها و اهداف آن‌ها



شکل ۱. تصویری از فرآیند متدولوژی

تصویر فرایند متدولوژی Hodge Mock در شکل ۱ آورده شده‌است. طبق این تصویر مشاهده می‌شود که این متدولوژی دارای ۵ فاز کلی است. همچنین به منظور به حداقل رساندن نرخ خطاهای احتمالی دو قابلیت در آن وجود دارد:

1. امکان برگشت به عقب در مراحل وجود دارد؛ در نتیجه در صورت بروز خطا در مدل‌های طراحی در هر مرحله، می‌توان به مرحله‌ی قبلی برگشت و آن را اصلاح کرد.
2. در پایان هر فاز سناریوهایی به منظور ارزیابی مدل‌های بدست آمده در هر فاز مورد ارزیابی قرار داد (در شکل ۱ به صورت دایره‌ای نارنجی رنگ علامت‌گذاری شده‌اند).

در ادامه به بررسی هر فاز می‌شود.

پیش از مرور اجمالی بر هر کدام از فازها، در ابتدا نگاهی بر اهداف کلی، خروجی‌های کلی، جایگاه فاز در قلمروهای مسئله و محصولات بدست آمده از هر کدام از آن‌ها پرداخته می‌شود.

جدول ۱. کلیاتی از فازهای متدولوژی

فاز	جایگاه در قلمروی مسئله	هدف فاز	مصنوعات	سناریوهای ارزیابی در پایان فاز
تحلیل	قلمروی مسئله (ناظر بر چیستی)	شناسایی قلمروی سیستم کشف آبجکت‌های حوزه‌ی مسئله	تحلیل نیازمندی (سند نیازمندی) تحلیل اطلاعات (ERD، ORD، OD، OCR و نمودار وراثت) تحلیل رخدادها (SBD، SBS، بروزرسانی ERD در صورت لزوم و بروزرسانی ORD در صورت نیاز) گذار به طراحی (نمودار CSD)	بر اساس نیازمندی‌ها با هدف اطمینان از پوشش آن‌ها
طراحی سیستم	قلمروی مسئله (ناظر بر چیستی)	درک رفتار سیستم شناسایی آبجکت‌های سیستم شناسایی تعامل بین آبجکت‌های سیستم آبجکت‌های مورد نیاز برای عملیاتی شدن سیستم	OID OBD OBS تکمیل مستندات قبلی در صورت لزوم	تدقیق سناریوها
طراحی نرم‌افزار	قلمروی راه‌حل (تا حد خوبی ناظر بر چگونگی)	تعیین زبان برنامه‌نویسی و سخت‌افزار بدست آوردن اطلاعاتی به منظور پیاده‌سازی آبجکت	تکمیل OD OPD	تدقیق سناریوهای مراحل قبل ساخت تست برای کل سیستم
پیاده‌سازی	قلمروی راه‌حل (ناظر بر چگونگی)	کدنویسی	-	-
آزمون	قلمروی راه‌حل (ناظر بر چگونگی)	آزمون		

فاز اول: تحلیل

تصویر فرایند متدولوژی Hodge Mock در شکل ۱ آورده شده‌است. طبق این تصویر مشاهده می‌شود که این متدولوژی دارای ۵ فاز کلی است. همچنین به منظور به حداقل رساندن نرخ خطاهای احتمالی دو قابلیت در آن وجود دارد:

فعالیت‌های این فاز با هدف اصلی شناسایی قلمروی مسئله (ناظر بر چیستی سیستم) صورت می‌گیرد. عمده‌ی این هدف با شناسایی آبجکت‌های حوزه‌ی مسئله و روابط بین آن‌ها صورت می‌گیرد.

در این فاز، نیازمندی‌ها دریافت شده و در نهایت دو خروجی کلی زیر به دست می‌آید:

1. مدل تحلیل سیستم (ناظر بر آبجکت‌های حوزه‌ی تحلیل و ارتباطات بین آن‌ها)
2. سناریوهایی برای ارزیابی محصولات این فاز که این سناریوها متضمن آن هستند که مدل‌های ایجاد شده در این فاز، تمامی نیازمندی‌ها را پوشش دهند.

در ادامه به بررسی فعالیت‌های اساسی فاز تحلیل پرداخته می‌شود.

در حالت کلی، فاز تحلیل شامل سه فعالیت اساسی است که در ادامه هر کدام از آن‌ها به‌صورت ورودی–فعالیت–فرایندی که در ذیل فعالیت اجرا می‌شود و در نهایت مصنوعات ایجاد شده در پایان فعالیت مورد بررسی قرار می‌گیرد. این سه فعالیت عبارت‌اند از:

1. تحلیل نیازمندی
2. تحلیل اطلاعات
3. تحلیل رخدادها

در ادامه به تفصیل این فعالیت‌ها پرداخته می‌شود. البته بایستی توجه کرد که در پایان بخش تحلیل، یک فعالیت دیگر نیز مطرح شده است که در آن گذار به طراحی گفته می‌شود.

۱) تحلیل نیازمندی

هدف

شناسایی و تبیین نیازمندی‌های عملکردی و غیرعملکردی

توصیف دقیق سیستم از منظر اهداف و قلمرو کاربردی

ورودی

سند نیازمندی (که می‌تواند مهم و حتی ناسازگار باشد)

فرایند اجرا

در قالب پاسخ‌گویی به دو سوال است:

در مورد قلمروی مسئله چه چیزی را باید بدانیم؟ (ناظر بر دانستن Scope سیستم)

از چه منابعی می‌توانیم برای اشراف بر قلمروی مسئله کمک بگیریم؟

مصنوعات

یک سند نیازمندی دقیق و سازگار

۲) تحلیل اطلاعات

هدف

شناسایی آبجکت‌ها و خصوصیات آن

شناسایی ارتباط بین آبجکت‌ها

فرایند

1. شناسایی موجودیت‌ها و ارتباط بین آن‌ها با ERD.
2. ترجمه موجودیت‌ها به آبجکت‌ها و بیان آن‌ها در قالب ORD. بررسی این موضوع که کدام آبجکت می‌تواند توصیف بهتری برای هر موجودیت باشد.
3. برای توصیف دقیق هر آبجکت، از سند آبجکت (اصطلاحا OD) استفاده می‌شود.
4. پس از شناسایی موجودیت‌ها و روابط آن‌ها روی ERD و سپس ترجمه آن‌ها به آبجکت‌ها با استفاده از ORD، سپس توصیف هر آبجکت در قالب OD، اینک نوبت به آن می‌رسد که تعامل بین آبجکت‌ها نیز تشریح شود. این کار با استفاده از یک جدول به نام OCR و مبتنی بر OD صورت می‌گیرد. در این روابط شامل موارد زیر هستند:
    - سرویس‌هایی که توسط هر آبجکت فراهم می‌شود.
    - سرویس‌هایی که توسط هر آبجکت استفاده می‌شود.
    - آبجکت‌هایی که مسئول ارائه‌ی سرویس‌هایی به این آبجکت هستند.
5. در نهایت لازم است روابط ارث‌بری (اعم از روابط Generalization و Specification) در قالب نمودار وراثت (موسوم به نمودار I) نمایش داده شود.

مصنوعات

1. ERD: کاربرد در پردازش رخدادها و تحلیل ارتباطات بین موجودیت‌ها.
2. ORD: نمایش آبجکت‌ها، ویژگی‌هایشان، زیرآبجکت‌ها و رفتارهایشان.
    - با این نمودار، سرویس‌هایی که توسط هر آبجکت ارائه می‌شود مشخص می‌شود.
    - می‌توان مجموعه‌ای از آبجکت‌های مرتبط باهم را در یک گروه قرار داد که به آن گروه Subject گفته می‌شود.
        - استفاده از مفهوم Subject به سادگی به تقسیم‌بندی قلمروی مسئله کمک می‌کند.

- با مفهوم Subject، تا حدی ضعف مدل ERD در نمایش سلسله‌مراتب آبجکت‌ها و موجودیت‌ها جبران می‌شود

3. OD: ویژگی‌های هر یک از آبجکت‌ها را توصیف می‌کند (یک مستند هستانی است).
    - به صورت تکاملی در جریان چرخه‌ی ایجاد تکمیل می‌شود.
    - پارامترهای توصیف‌کننده‌ی هر آبجکت عبارت‌اند از:
        - هدف آبجکت
        - زیرآبجکت‌ها (رابطه‌ی has-a یا به عبارت امروزی رابطه‌ی whole–part)
        - رابطه‌ی is-a (رابطه‌ی والد–فرزندی یا ارث‌بری)
        - سرویس‌هایی که در اختیار سایر آبجکت‌ها قرار می‌دهد (Provide)
        - سرویس‌هایی که از سایر آبجکت‌ها دریافت می‌کند (Required)
        - توصیف هر سرویس
4. OCR: به منظور تشریح ارتباطات بین آبجکت‌ها (سرویس‌هایی که به یکدیگر می‌دهند و یا از یکدیگر دریافت می‌کنند) می‌باشد.
    - این مدل از روی OD ساخته می‌شود.
    - شامل این اطلاعات است: نام کلاس، سرویس‌هایی که ارائه می‌کند، سرویس‌هایی که از آن استفاده می‌کند (به همراه کلاس‌های فراهم‌کننده‌ی آن سرویس).
    - بایستی مشابه با OD، دائما به‌روز نگه‌داری شود.
    - کاربرد اساسی آن در مراقبت و نگه‌داری است.

۳) تحلیل رخدادها

هدف

- شناسایی رفتار آبجکت‌ها در سطح سیستم (خصوصا از دیدگاه خارجی سیستم).
- در این فعالیت، سیستم را به‌مثابه‌ی یک ماشین محرک–پاسخ مشاهده می‌کنیم (به این معنا که سیستم در برابر یک محرک خارجی، پاسخ مناسبی ارائه می‌دهد).
- این فعالیت با هدف اعتبارسنجی سیستم (Validation) نیز صورت می‌گیرد.

مصنوعات ورودی

- رفتارهایی که سیستم بایستی در قبال هر محرک انجام دهد (احتمالا بر مبنای سند نیازمندی که در فعالیت اول این فاز تدقیق شده است، استخراج می‌شود).

فرآیند

1. استخراج فعالیت‌های اساسی سیستم که در راستای برقراری مقصود سیستم صورت می‌گیرد.

- به فعالیت‌هایی اطلاق می‌شود که به‌صورت مستقیم در فعالیت‌های اساسی سیستم مورد استفاده قرار می‌گیرند.

2. استخراج فعالیت‌های حمایتی سیستم که فعالیت‌های اساسی سیستم را پشتیبانی می‌کنند.
3. شرح فعالیت‌های اساسی سیستم و فعالیت‌های حمایتی آن در قالب یک مدل جدولی به نام توصیف رفتاری سیستم (موسوم به SBS).
4. استخراج مدل رفتاری سیستم از دیدگاه خارج از سیستم و بیان آن در قالب یک ماشین حالت به نام مدل نمودار رفتار سیستم (موسوم به SBD) که تغییر وضعیت سیستم را در قبال هر کدام از محرک‌های خارجی توصیف می‌کند.
5. بر اساس فاز تحلیل اطلاعات، اقلام اطلاعاتی که در نتیجه‌ی هر محرک به دست می‌آید به همراه پاسخ مناسب آن باید مشخص شود.
6. برای ساخت SBD لازم است از اطلاعات سند توصیف رفتار آبجکت (OD) استفاده شود.
7. به‌روزرسانی سایر مدل‌های تحلیل (در صورت لزوم):
    - در صورت شناسایی موجودیت جدید، نمودار ERD می‌بایست به‌روزرسانی شود.
    - در صورت شناسایی اقلام اطلاعاتی جدید، افزودن آن‌ها به آبجکت‌های مناسب.
        - نتیجه: به‌روزرسانی ORD در صورت نیاز.

مصنوعات خروجی

- مدل جدولی SBS که فعالیت‌های اساسی و حمایتی سیستم را شرح داده است. برای هر کدام از این دو دسته فعالیت، در جدول دو ستون اساسی وجود دارد:
    - اول: نام فعالیتی که انجام می‌شود.
    - دوم: محرکی که باعث بروز آن فعالیت می‌شود.
- ماشین حالت SBD که در آن تغییر حالات سیستم در قبال محرک‌های بیرونی قابل بیان است.
- به‌روزرسانی سایر مصنوعات (نظیر ERD و ORD) در صورت لزوم.

۴) گذار به طراحی

هدف

- ارزیابی تحلیل با سناریوهای ارزیابی.
- برگشت به فاز قبل در صورت نیاز.
- خروج از فاز تحلیل و ورود به فاز طراحی.
    - غنی‌سازی نمودار ORD.
- به متدولوژی در گذار همواره‌ی درز کمک کند.

مصنوعات ورودی

- نمودار ORD.

فرایند

- افزودن جزئیاتی به نمودار ORD در خصوص نحوه‌ی تعامل آبجکت‌ها با یکدیگر.
- ترسیم یک نمودار سرویس‌دهنده – سرویس‌گیرنده (موسوم به نمودار CSD).

مصنوعات خروجی

- نمودار میزبان – سرویس‌دهنده (موسوم به CSD): بیان رابطه‌ی بین آبجکت‌های سیستم در قالب یک ارتباط کلاینت–سرویسی، حاوی اطلاعات زیر:
  - چگونگی ارتباط بین آبجکت‌ها.
  - انتقال پیام بین آن‌ها.

فاز دوم: فاز طراحی سیستم

در این فاز می‌خواهیم در نهایت بفهمیم که سیستم چگونه قرار است کار کند. برای این منظور، نیازمند دانستن موارد زیر هستیم:

1. دانستن جزئیات رفتار آبجکت‌های سیستم و تعامل بین آن‌ها.
2. شناسایی آبجکت‌هایی که به سیستم این قابلیت را می‌دهند که سیستم کار کند.

تمامی این موارد مستقل از پیاده‌سازی صورت می‌گیرد و در پایان این فاز، سناریوهای ارزیابی فاز قبلی (تحلیل)، تدقیق شده و در فاز جاری در قالب سناریوهای ارزیابی طراحی سیستم مورد استفاده قرار می‌گیرد.

هدف این فاز

درک این موضوع که سیستم قرار است چگونه کار کند (مستقل از روش پیاده‌سازی).

مصنوعات ورودی این فاز

1. OD
2. OCR
3. CSD

فعالیت‌های اساسی این فاز

1. شناسایی آبجکت‌های اساسی که برای پیاده‌سازی سیستم مورد نیاز هستند.
2. تدقیق آبجکت‌هایی که در طی فرآیند تحلیل شناسایی شده‌اند.

3. تعریف واضح و روشن تعامل بین آبجکت‌ها به‌گونه‌ای که منجر به تحقق هدف اساسی سیستم شود.
    - این تعامل می‌تواند با استفاده از یک نمودار به نام نمودار واسط آبجکت (موسوم به OID) صورت گیرد. بر مبنای این نمودار، مصنوعات زیر نیز تولید می‌شوند:
        i. یک ماشین حالت که نشان‌دهنده‌ی وضعیت هر کدام از آبجکت‌ها است (از آن تحت عنوان نمودار رفتار آبجکت موسوم به OBD یاد می‌شود).
        ii. یک سند نیز تحت عنوان توصیف رفتار آبجکت (موسوم به OBS) از هر کدام از آبجکت‌ها ساخته می‌شود و هر یک از رفتارهای آن‌ها را توصیف می‌کند.
4. در صورت لزوم، مدل‌های OD و OCR نیز غنی‌تر می‌گردند (نظیر افزودن ورودی و خروجی هر سرویس).

مصنوعات ایجاد شده در این فاز

- نمودار واسط آبجکت (OID): به منظور نمایش تعاملات یک آبجکت با سایر آبجکت‌ها (از زاویه‌ی نگاه همان آبجکت) مورد استفاده قرار می‌گیرد.
    - از نمودار CSD (همان نمودار نمایش کلاینت–سرویسی آبجکت‌ها) به دست می‌آید.
        - نمودار OID در سطح یک آبجکت و با تفصیل بیشتری نسبت به CSD توصیف می‌کند.
- نمودار رفتار آبجکت (OBD): یک ماشین حالت است که وضعیت یک آبجکت را به همراه روند تغییر وضعیت نشان می‌دهد.
- توصیف رفتار آبجکت (OBS): توصیف متنی از تمامی رفتارهای هر آبجکت را نشان می‌دهد.

فاز سوم: فاز طراحی نرم‌افزار

در این فاز قرار است بفهمیم که سیستم چگونه قرار است پیاده‌سازی شود. برخلاف مراحل قبل، در این مرحله علاوه بر تعیین زبان برنامه‌نویسی و پلتفرم سخت‌افزاری، اطلاعاتی را به منظور پیاده‌سازی به هر آبجکت بیفزاییم.

برای ارزیابی محصولات این فاز، فعالیت‌های زیر را در نظر می‌گیریم:

1. تدقیق سناریوهای ارزیابی دو فاز قبلی.
2. گردآوری اطلاعات جامع (با محوریت سناریوهای ایجاد شده) به منظور تست سیستم در مراحل بعدی.

باید توجه شود که تمامی محصولات این فاز در پیاده‌سازی مورد استفاده قرار می‌گیرند؛ بنابراین لازم است در پایان تمامی فازهای قبلی، مدل‌های ایجاد شده را مرور کرده باشیم.

- فرایند این فاز
  a. در صورت نیاز، به مدل‌های فازهای قبلی جزئیات لازم افزوده شود.
  b. مدل OD باید تکمیل شده و حاوی جزئیات بیشتری شود (نظیر public یا private بودن رفتارها).
  c. بیان جزئیات رفتاری هر آبجکت در قالب نمودار OPD.
- مصنوعات تولید شده در این فاز
  a. نمودار OPD.

فاز چهارم و پنجم: پیاده‌سازی و آزمون

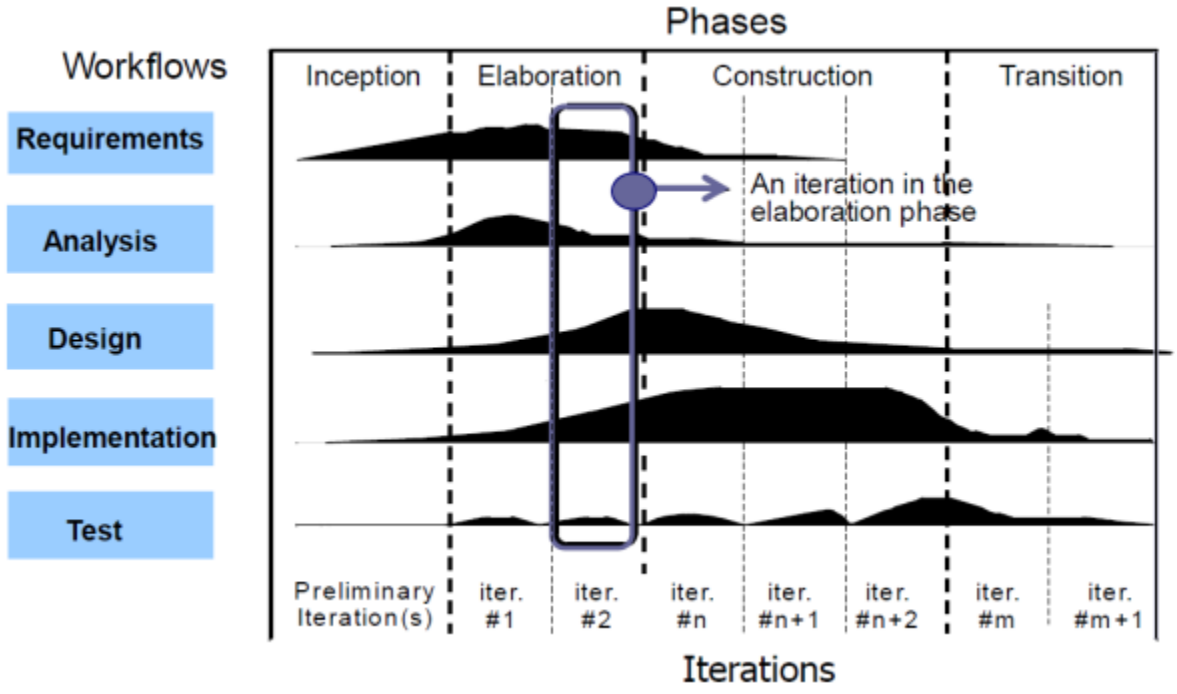
در خصوص این فاز، جزئیات زیادی در مقاله‌ی مربوطه ارائه نگشته است.

متدولوژی USDP

متدولوژی USDP که با نام UP نیز شناخته می‌شود، در سال ۱۹۹۹ توسط رامبا، جیکابسن و بوچ ساخته شد که نسخه‌ی ساده‌شده و غیراختصاصی متدولوژی RUP است. این متدولوژی بر پایه‌ی UML بوده و مبتنی بر Use-Case است. همچنین این متدولوژی در قالب فرایند تکراری–افزایشی اجرا می‌شود. در این راستا، چرخه‌های نرم‌افزار در روی زمان به چهار فاز ترتیبی تقسیم می‌شود که در شکل ۲ قابل مشاهده است. به ترتیب فاز آغاز، فاز تفصیل، فاز ساخت و فاز انتقال، چرخه‌های UP را تشکیل می‌دهند.

چرخه حیات نرم‌افزار در USDP

در شکل ۲ برای هر فاز می‌توانیم تعدادی تکرار را مشاهده کنیم که هرکدام شامل چهار جریان کاری هستند که عبارت‌اند از: نیازمندی‌ها، تحلیل، طراحی، پیاده‌سازی و تست. در ادامه به توضیح فازها و فعالیت‌های آن‌ها می‌پردازیم.



شکل ۲. مدل چرخه حیات USDP

فاز آغاز

این فاز نقطه‌ی عطف چشم‌انداز پروژه است که در آن چشم‌انداز، حوزه‌ی پروژه و توجیه اقتصادی تعریف می‌شوند. این فاز بین چهار هفته طول کشیده و در طی یک یا دو تکرار انجام می‌شود. در این فاز برای امکان‌سنجی، نمونه‌های اولیه‌ای فنی برای اعتبارسنجی تصمیمات تکنولوژی و همچنین نمونه‌های اولیه‌ای مفهومی برای اعتبارسنجی نیازمندی‌های کسب‌وکار ساخته می‌شوند. یک توجیه اقتصادی ساخته می‌شود تا نشان داده شود که پروژه منافع قابل اندازه‌گیری تحویل خواهد داد. در این فاز نیازمندی‌های اساسی ثبت می‌شوند تا کمک کنند حوزه‌ی سیستم مشخص شود. همچنین یکی از مهم‌ترین کارها در این مرحله، تشخیص ریسک‌های حیاتی است. این فاز یک مرحله‌ی مقدماتی است که در آن می‌خواهیم به مواردی پاسخ دهیم؛ از قبیل این‌که هدف پروژه چیست و آیا ارزش تلاش را دارد؟ آیا پروژه از لحاظ تکنولوژی، اقتصادی و زمانی انجام‌پذیر است؟ آیا بهتر است یک سیستم موجود را به‌کار بگیریم یا سیستم جدیدی بسازیم؟ تخمین هزینه‌ها و ریسک‌ها چیست؟ و در نهایت تصمیم بر این است که آیا روند پروژه را ادامه دهیم یا خیر؟ همچنین در این فاز درگیر برنامه‌ریزی و مدیریت پروژه نیز می‌شویم.

پس‌شرط‌ها و مصنوعات Inception

مصنوعات قابل تحویل در این فاز را می‌توان در شکل ۳ مشاهده کرد. مهم‌ترین آن‌ها سند چشم‌انداز است که در آن نیازمندی‌های اصلی، قابلیت‌ها و محدودیت‌ها بیان شده است. همچنین در این فاز باید یک معماری اولیه برای پروژه مشخص شود و پروژه در امکان‌سنجی شده و پروژه بتواند ادامه پیدا کند.

Conditions of satisfaction	Deliverable
The stakeholders have agreed on the project objectives	A vision document that states the project's main requirements, features, and constraints
System scope has been defined and agreed on with the stakeholders	An initial use case model (only about 10% to 20% complete)
Key requirements have been captured and agreed on with the stakeholders	A project glossary
Cost and schedule estimates have been agreed on with the stakeholders	An initial project plan
A business case has been raised by the project manager	A business case
The project manager has performed a risk assessment	A risk assessment document or database
Feasibility has been confirmed through technical studies and/or prototyping	One or more throwaway prototypes
An architecture has been outlined	An initial architecture document

شکل ۳. مصنوعات قابل تحویل در فاز inception

به دلیل آن‌که امکان‌پذیری پروژه و ادامه‌ی آن قبل از آغاز مشخص نیست، منطقی است که این فاز کوتاه باشد تا اگر پروژه رد شود، هزینه و زمان زیادی هدر نرود.

فاز تفصیل

این فاز نقطه‌ی عطف معماری پروژه است که در آن تعریف پروژه دقیق‌تر می‌شود و یک خط پایه‌ی معماری قابل اجرا ساخته می‌شود. همچنین برای ایجاد و استقرار پروژه، برنامه‌ریزی دقیق‌تر انجام می‌شود. هدف فاز قبل فهم چیستی مسئله است، در حالی‌که فاز تفصیل، قلمرو راه‌حل را مدنظر قرار می‌دهد.

در این فاز، ارزیابی ریسک دقیق‌تر می‌شود، صفات کیفیت تعریف می‌شوند، نیازمندی‌های وظیفه‌ای تا حدود ۸۰٪ ثبت می‌شوند، یک برنامه‌ریزی مفصل برای فاز بعدی ایجاد می‌شود و در اثر یک مناقشه شامل منابع، زمان، تجهیزات، افراد و هزینه تنظیم و ارائه می‌شود.

در این فاز، تمرکز روی جریان‌های کاری به این شکل است که در نیازمندی، حوزه‌ی سیستم و نیازمندی‌های دقیق‌تر مشخص می‌شود؛ در تحلیل، چیزی که قرار است ساخته شود مدل می‌شود؛ در طراحی، یک معماری پایدار ساخته می‌شود؛ در پیاده‌سازی، یک خط پایه‌ی معماری قابل اجرا ساخته می‌شود؛ و در تست، خط پایه ساخته‌شده تست می‌شود.

در این فاز، تفصیل ریسک‌های پروژه باید از بین برود؛ معماری و اسکلت‌بندی باید توسط مشتری و کاربران پذیرفته شود؛ ساخت نسخه‌های آزمایشی از اجزای پرریسک فنی برای پیاده‌سازی باید انجام شود؛ تخمین هزینه باید نهایی شده باشد؛ مستندات راهنمای کاربر مقدماتی باید ساخته و تحلیل شده باشد.

پس‌شرط‌ها و مصنوعات Elaboration

در این فاز، یک خط پایه‌ی معماری قابل اجرا ساخته شده و همچنین با ذی‌نفعان، مبنای ادامه‌ی پروژه توافق شده و سند مربوطه ایجاد می‌شود. سایر مصنوعات و اصلاحات در شکل ۴ قابل مشاهده است.

Conditions of satisfaction	Deliverable
A resilient, robust executable architectural baseline has been created	The executable architectural baseline
The executable architectural baseline demonstrates that important risks have been identified and resolved	UML static model UML dynamic model UML use case model
The vision of the product has stabilized	Vision document
The risk assessment has been revised	Updated risk assessment
The business case has been revised and agreed with the stakeholders	Updated business case
A project plan has been created in sufficient detail to enable a realistic bid to be formulated for time, money, and resources in the next phases	Updated project plan
The stakeholders agree to the project plan	
The business case has been verified against the project plan	Business case
Agreement is reached with the stakeholders to continue the project	Sign-off document

شکل ۴. مصنوعات قابل تحویل در فاز elaboration

فاز ساخت

این فاز نقطه‌ی عطف قابلیت‌های عملیاتی اولیه است؛ به‌گونه‌ای که در آن محصول تا حدی ساخته می‌شود که بتوان آن را برای اولین بار به کاربر نهایی ارائه داد.

هدف این فاز آن است که به‌صورت تکراری مصنوعات‌ی که قبلا ساخته شده‌اند را تکامل داده و به سیستم هدف برسانیم. تمامی نیازمندی‌ها (تا ۱۰۰٪ یا باقی‌مانده)، تحلیل و طراحی تکمیل می‌شوند. خط پایه‌ی معماری قابل اجرا که در فاز قبل ساخته شده بود، تکامل پیدا کرده و به سیستم نهایی تبدیل می‌شود.

پس‌شرط‌ها و مصنوعات Construction

محصولاتی که در این فاز ساخته یا تکمیل می‌شوند در شکل ۵ قابل مشاهده‌اند.

Conditions of satisfaction	Deliverable
The software product is sufficiently stable and of sufficient quality to be deployed in the user community	The software product The UML model Test suite
The stakeholders have agreed and are ready for the transition of the software to their environment	User manuals Description of this release
The actual expenditures vs. the planned expenditures are acceptable	Project plan

شکل ۵. مصنوعات قابل تحویل در فاز construction

فاز انتقال

این فاز نقطه‌ی عطف انتشار است که در آن محصول ساخته‌شده به محیط کاربر منتقل می‌شود. این فاز شامل ساخت، تحویل، آموزش و برنامه‌ریزی برای پشتیبانی و نگه‌داری محصول می‌باشد.

هدف این فاز، استقرار نهایی محصول نرم‌افزاری است که در انتهای فاز قبل ساخته شده است. در این فاز، تست بتا و پذیرش انجام شده و ایرادها برطرف می‌شوند. محیط کاربر برای نرم‌افزار جدید

آماده می‌شود و تغییرات لازم در نرم‌افزار انجام می‌شود تا در محیط کاربر عملیاتی گردد و مشکلات پیش‌بینی‌نشده استقرار حل شوند.

در این فاز، مستندات مختلف مانند راهنمای کاربر ساخته می‌شود، به کاربر مشاوره ارائه می‌شود و همچنین بازبینی پساپروژه انجام می‌گیرد.

در این پروژه، جریان کاری نیازمندی معنا ندارد؛ مدل تحلیل در صورت نیاز به‌روزرسانی می‌شود؛ مدل طراحی در صورت بروز مشکلات به‌روزرسانی می‌شود؛ پیاده‌سازی تغییرات نرم‌افزار برای محیط کاربر انجام می‌شود و همچنین تست بتا و تست پذیرش در محیط کاربر انجام می‌شوند.

پس‌شرط‌ها و مصنوعات Transition

در این فاز، محصول نرم‌افزاری ارائه شده و سایر موارد در شکل ۶ قابل مشاهده‌اند.

Conditions of satisfaction	Deliverable
Beta testing is completed, necessary changes have been made, and the users agree that the system has been successfully deployed The user community is actively using the product	The software product
Product support strategies have been agreed on with the users and implemented	User support plan Updated user manuals

شکل ۶. مصنوعات قابل تحویل در فاز transition

نقاط قوت و ضعف

نقاط قوت

- فرایند تکراری–افزایشی.
- فرایند به‌خوبی مستندسازی شده است.
- بر اساس مدل‌سازی وظیفه‌ای، رفتاری و ساختاری قلمروی مسئله و سیستم.
- قابلیت ردیابی از طریق Use-Case پشتیبانی می‌شود.
- تا حد خوبی بدون ابهام است (البته سکسکه‌هایی وجود دارد؛ برای مثال در تبدیل use-caseها به نمودارهای توالی).

- فرایند معماری‌محور که توصیف روند کلی یک طرح معماری را ایجاب می‌کند.
- به قابلیت سفارشی‌سازی پرداخته شده است.
- ایجاد مبتنی بر ریسک با هدف کاهش ریسک‌ها قبل از انجام تکالیف.
- پشتیبانی از انواع مدل‌سازی در تمام سطوح (از قلمروی مسئله تا اشیا و منطق تا فیزیکی).
- زبان مدل‌سازی غنی، به‌ویژه در مدل‌سازی ساختاری و رفتاری.
- تا حدی از روش‌های صوری پشتیبانی می‌کند (از طریق OCL).
- پشتیبانی قوی ابزاری.
- نسبت به RUP بسیار ساده‌تر است.
- برخلاف RUP، تحلیل و طراحی جریان‌های کاری جدا از هم هستند که هرکدام واحدهای کاری و محصولات مخصوص به خود دارند.

نقاط ضعف

- فرایند پیچیده است (البته نسبت به RUP بسیار ساده‌تر است).
- فرایند برای افرادی که تازه‌کار هستند، به‌ویژه به دلیل ماهیت تکراری–افزایشی بودن آن، پیچیده‌تر می‌شود.
- پیکربندی آن دشوار است.
- فاز مراقبت و نگه‌داری ندارد.
- تعداد مدل‌ها زیاد است.
- تعصب به استفاده از UML؛ به‌خصوص این‌که UML بی‌نقص نیست و می‌تواند مشکل ناسازگاری مدل‌ها را تشدید کند.
- فعالیت‌های مدل‌سازی کسب‌وکار، استقرار و مدیریت نقش‌های محوری خود را در قالب جریان‌های کاری از دست داده‌اند.

کلیاتی از فرآیند متدولوژی

جدول ۲. کلیاتی از فازهای متدولوژی

فاز	هدف فاز	فعالیت	توضیحات
آغاز	تعریف نیازمندی‌ها در سطح بالا تحلیل مقدماتی تعیین منابع ارزیابی ریسک	شناسایی	توجیه اقتصادی ایجاد سیستم
		انعقاد پروژه	تعیین مرز سیستم تعیین منابع سیستم
		تعریف رسمی نیازمندی‌ها	تعریف نیازمندی‌ها با سطح کوچکی از فرمالیسم
		ارزیابی ریسک	با هدف سنجش امکان‌پذیری سیستم
ایجاد	تحلیل تفصیلی طراحی تفصیلی و طراحی معماری تست و پیاده‌سازی به صورت همزمان	تحلیل	دقیق‌تر کردن نیازمندی‌های مستخرج مراحل قبل مدل‌سازی قلمروی مسئله با OPD
		طراحی	دقیق‌تر کردن معماری افزودن اطلاعاتی برای پیاده‌سازی سیستم
		پیاده‌سازی مولفه‌های سیستم	کدنویسی آماده‌سازی بسترهای سخت‌افزاری و نرم‌افزاری لازم
استقرار	استقرار سیستم و نشاندن در محیط کاربر مراقبت و نگهداشت	هضم سیستم	وارد کردن سیستم به محیط عملیاتی (آموزش کاربران، مستندسازی، مهاجرت به سیستم جدید و تست پذیرش)
		استفاده و نگهداشت	پشتیبانی سیستم (بدون دست به کد شدن) نگهداشت سیستم
		پایش سیستم	بررسی موارد زیر آیا سیستم کامان نیاز کاربر را برطرف می‌کند؟ آیا سیستم کماکان قابل نگهداشت است؟
		اعلام پایان عمر سیستم	تهیه‌ی یک مستند کوتاه از سیستم تهیه‌ی مستند از درس آموخته‌های پروژه تسویه حساب

زبان‌های مدل‌سازی

تنها زبان مدل‌سازی این متدولوژی، OPD است که همراه با توصیف متنی OPL مورد استفاده قرار می‌گیرد؛ این زبان تمامی وجود رفتاری، ساختاری و وظیفه‌ای را می‌پوشاند.

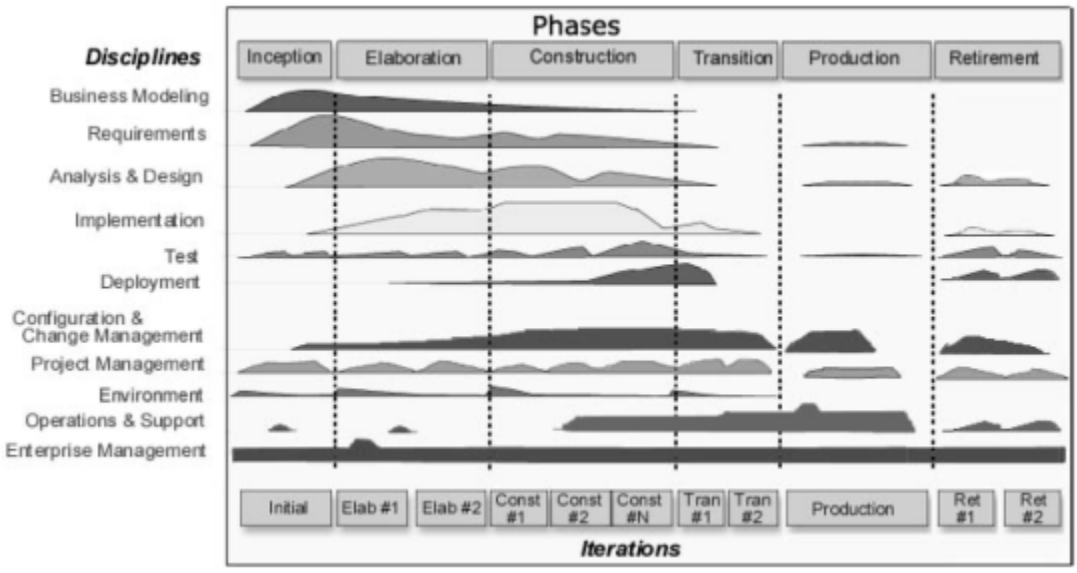
متدولوژی EUP

متدولوژی EUP در سال ۲۰۰۰ توسط امبلر و کنستانتین به‌عنوان یک نسخه‌ی گسترش‌یافته از متدولوژی RUP معرفی شد. ایجادکنندگان این فرایند معتقدند RUP از مشکلات جدی رنج می‌برد که به ادعای آن‌ها در EUP اصلاح شده است. این مشکلات عبارت‌اند از:

- متدولوژی RUP، پشتیبانی از سیستم و بازنشستگی نهایی را پوشش نمی‌دهد.
- متدولوژی RUP به‌طور صریح از ایجاد زیرساخت‌های سطح سازمانی پشتیبانی نمی‌کند.
- طبیعت تکرارشونده‌ی RUP هم یک نقطه‌ی قوت و هم یک نقطه‌ی ضعف است؛ زیرا درک چرخه‌های تکرارشونده برای بسیاری از ایجادکنندگان باتجربه دشوار است.
- رویکرد شرکت رشال برای ایجاد RUP در ابتدا ابزارمحور بود؛ بنابراین فرایند حاصل برای نیازهای ایجادکنندگان کافی نیست.

مدل چرخه‌های EUP در شکل ۷ نشان داده شده است. این مدل با افزودن دو فاز جدید و همچنین گسترش فعالیت‌های برخی از disciplineهای قبلی، RUP را گسترش می‌دهد.

دیدگاه EUP نسبت به مدل‌سازی نیز تا حدی با RUP متفاوت است. در حالی که RUP بر پیروی از UML تاکید دارد، EUP از برخی نمادهای قدیمی‌تر مدل‌سازی نیز استفاده می‌کند. نمونه‌ای از این موارد، استفاده از نمودارهای جریان داده برای مدل‌سازی کسب‌وکار است. علاوه بر این، EUP تاکید می‌کند که Use-Case به‌تنهایی برای مدل‌سازی نیازمندی‌ها کافی نیست؛ به همین دلیل نقش محوری Use-Case را که در RUP دارد، در EUP ایفا نمی‌کند.



شکل ۷. مدل چرخه حیات EUP

بخش‌های زیر به‌طور خلاصه افزودن‌ها و تغییراتی را که EUP نسبت به RUP دارد، توضیح می‌دهند.

فازهای جدید

دو فاز جدیدی که EUP به RUP اضافه کرده است، عبارت‌اند از:

- Production: به‌عنوان فاز پنجم اضافه شده است. تمرکز این فاز بر نگهداری نرم‌افزار در production است تا زمانی که با نسخه‌های جدید جایگزین شود، یا با اجرای مجدد چرخه حیات، بازنشسته و حذف گردد. در این فاز هیچ تکراری وجود ندارد. این فاز تا حدی شبیه به فاز نگهداری در چرخه حیات کلی ایجاد نرم‌افزار است، زیرا بیشتر با عملیات و پشتیبانی از سیستم سروکار دارد. اما برخلاف نگهداری کلاسیک، هر نیازی به تغییر سیستم، حتی رفع یک خطا، منجر به شروع مجدد چرخه ایجاد خواهد شد.
- Retirement: در سال ۲۰۰۲ به‌عنوان فاز ششم اضافه شده است. تمرکز این فاز بر حذف دقیق یک سیستم از production است؛ یا به دلیل اینکه دیگر نیازی به آن نیست، یا در حال جایگزین شدن با سیستم دیگری است. این فاز معمولا شامل موارد زیر است:
  - شناسایی جفت‌شدگی‌های سیستم موجود با سایر سیستم‌ها.
  - طراحی مجدد و اصلاح سایر سیستم‌ها به‌گونه‌ای که دیگر به سیستم در حال بازنشسته شدن وابسته نباشند.
  - تبدیل داده‌های قدیمی موجود.
  - بایگانی داده‌هایی که قبلا توسط سیستم نگهداری می‌شدند و دیگر نیازی به آن‌ها در سایر سیستم‌ها نیست.
  - مدیریت پیکربندی نرم‌افزار حذف‌شده به‌گونه‌ای که در صورت نیاز، امکان نصب مجدد آن وجود داشته باشد.
  - آزمون ادغام سیستم‌های باقی‌مانده به‌منظور اطمینان از اینکه پس از بازنشستگی سیستم، دچار اشکال نشده‌اند.

نظام جدید

دو نظام جدیدی که EUP به RUP اضافه کرده است، عبارت‌اند از:

- عملیات و پشتیبانی: مربوط به مسائل مرتبط با عملیات و پشتیبانی از سیستم است که معمولا با فاز نگهداری در چرخه حیات کلی ایجاد نرم‌افزار مرتبط است. با این حال، این نظام چندین فاز را شامل می‌شود، نه تنها فاز production محدود نیست. در طول فاز ساخت و شاید حتی زودتر در فاز تحلیل، ایجاد جریان‌ها، استانداردهای فرآیندی مربوط به عملیات و پشتیبانی انجام می‌شود. در طول فاز انتقال، بهبود یافته و تکمیل می‌شوند. جای‌گیری این نظام شامل آموزش کارکنان عملیات و پشتیبانی نیز می‌شود. در طول فازهای production و بازنشستگی، این نظام فعالیت‌های کلاسیک نگهداری را پوشش می‌دهد؛ یعنی کارکنان عملیات، نرم‌افزار را فعال نگه می‌دارند، نسخه‌برداری‌های لازم و پردازش‌های دسته‌ای را انجام می‌دهند و کارکنان پشتیبانی با کاربران ارتباط برقرار می‌کنند تا به آن‌ها در کار با نرم‌افزار کمک کنند.
- مدیریت سازمانی: مربوط به فعالیت‌هایی است که برای ایجاد، تکامل و نگهداری مصنوعات بین‌سیستمی سازمان، مانند مدل‌های سطح سازمانی (نیازمندی‌ها و معماری)، فرآیند نرم‌افزار، استانداردها، راهنماها و مصنوعات قابل استفاده مجدد، مورد نیاز است.

نظام تغییر داده‌شده

در EUP تغییرات متعددی در نظام RUP اعمال شده است، مانند موارد زیر:

- نظام آزمون در EUP برای شامل شدن اعتبارسنجی نیازمندی‌ها در طول فاز آغاز گسترش یافته است. این اعتبارسنجی با استفاده از تکنیک‌هایی مانند بازبینی‌ها، بازرسی‌ها و آزمون‌های سناریو انجام می‌شود.
- نظام استقرار در EUP با فعالیت‌های مدل‌سازی استقرار که در RUP بخشی از نظام تحلیل و طراحی بود، تقویت شده است. همچنین، EUP توصیه می‌کند که برنامه‌ریزی استقرار تا حد ممکن در چرخه حیات زودتر آغاز شود. به دلیل این دو تغییر، نظام استقرار در EUP به فازهای آغاز و تفصیل گسترش یافته است.
- نظام محیط به‌روزرسانی شده است تا شامل کارهای لازم برای تعریف محیط production شود.
- نظام مدیریت پیکربندی و تغییر و نظام مدیریت پروژه، به فازهای جدید production و بازنشستگی گسترش یافته‌اند. علاوه بر این، ویژگی‌های جدیدی به نظام مدیریت پروژه اضافه شده است که شامل مدیریت معیارها، مدیریت پیمانکاران فرعی و مدیریت نیروی انسانی می‌شود.

نقاط قوت و ضعف

نقاط قوت

- دارای همان نقاط قوت RUP است.
- به مسائل سطح سازمانی می‌پردازد.
- نگهداری به‌عنوان یک فاز مستقل در نظر گرفته می‌شود.
- به فعالیت‌های پس از اتمام پروژه، هنگام بازنشستگی آن، توجه می‌شود؛ به‌صورت یک فاز جدید به نام فاز بازنشستگی.
- به UML به‌طور سخت‌گیرانه پایبند نیست و از زبان‌های مدل‌سازی دیگر مانند DFD نیز استفاده می‌شود.

نقاط ضعف:

- مشابه RUP، متدولوژی EUP نیز این مشکلات را دارد:
  - بسیار پیچیده است.
  - دارای تعداد زیادی مدل‌های غیرضروری است.
  - پتانسیل بالایی برای ناسازگاری بین مدل‌ها دارد.
  - فرآیند استفاده‌شده در آن گیج‌کننده است.
  - سفارشی‌سازی آن دشوار است.
- EUP با افزودن دو فاز جدید و دو نظام جدید، پیچیدگی بیشتری به RUP اضافه کرده است.
- افزودن فاز نگهداری کافی نیست، زیرا هرگونه تغییری که موردنیاز باشد، منجر به شروع دوباره فرآیند ایجاد خواهد شد.

ارزیابی و مقایسه‌ی دو متدولوژی Hodge-Mock و USDP

ارزیابی متدولوژی‌ها از منظر فرآیندی

معیار اول: صراحت در تعریف

متدولوژی می‌بایست به درستی تعریف شده باشد؛ یک تعریف خوب دارای ویژگی‌های زیر است: صریح و بدون ابهام است.

1. پوشا است.
2. روشن است.
3. منطقی است.
4. صحیح و دارای جزئیات است.
5. سازگار و بدون تناقض است.

متدولوژی Hodge-Mock یک متدولوژی شیءگرا و فرآیندمحور است که تمرکز اصلی آن بر تحلیل چیستی سیستم و شناسایی آبجکت‌ها، رفتارها و روابط آن‌ها پیش از ورود به طراحی و پیاده‌سازی می‌باشد. این متدولوژی با تفکیک چرخه‌ی حیات به فازهای مشخص و تعریف مصنوعات تحلیلی و طراحی در هر فاز، تلاش می‌کند درک دقیقی از سیستم ایجاد کند. مستندسازی در Hodge-Mock صریح بوده و تمامی خروجی‌های فعالیت‌ها به‌عنوان محصولات رسمی متدولوژی شناخته می‌شوند.

در مقابل، USDP یک متدولوژی تکراری–افزایشی، معماری‌محور و مبتنی بر Use-Case است که با تکیه بر UML، چرخه‌ی حیات نرم‌افزار را به فازهای آغاز، تفصیل، ساخت و انتقال تقسیم می‌کند. USDP علاوه بر تمرکز بر چیستی سیستم، به‌صورت هم‌زمان به چگونگی پیاده‌سازی و مدیریت ریسک نیز توجه دارد. در مجموع، USDP نسبت به Hodge-Mock تعریف جامع‌تر و استانداردتری ارائه می‌دهد.

چرخه حیات و واحدهای کاری

در Hodge-Mock، فازهای چرخه‌ی حیات به‌صورت صریح معرفی شده‌اند که شامل: تحلیل، گذار به طراحی، طراحی سیستم، طراحی نرم‌افزار، پیاده‌سازی و آزمون می‌باشند. واحدهای کاری در این متدولوژی مشخص بوده و هر فاز دارای فعالیت‌ها، مصنوعات و سناریوهای ارزیابی اختصاصی است.

ترتیب انجام فعالیت‌ها می‌تواند منعطف باشد، اما انجام تمامی فعالیت‌ها الزامی است. همچنین امکان بازگشت به فازهای قبلی برای اصلاح خطاها وجود دارد.

در مقابل، USDP چرخه‌ی حیات را به چهار فاز آغاز، تفصیل، ساخت و انتقال تقسیم می‌کند که هر فاز شامل چندین تکرار است. در هر تکرار، جریان‌های کاری نیازمندی، تحلیل، طراحی، پیاده‌سازی و آزمون به‌صورت هم‌پوشان اجرا می‌شوند. هدف هر فاز در USDP به‌وضوح مشخص شده و وابستگی آن به مدیریت ریسک و تحویل تدریجی محصول بیشتر از Hodge-Mock است.

تولیدکنندگان (نقش‌ها)

در Hodge-Mock، نقش‌ها و افراد دخیل در فرآیند به‌صورت صریح تعریف نشده‌اند و تمرکز اصلی متدولوژی بر فعالیت‌ها و مصنوعات است. به همین دلیل، مسئولیت‌ها بیشتر به‌صورت ضمنی و وابسته به ساختار فعالیت‌ها قابل برداشت هستند.

در مقابل، USDP یک متدولوژی نقش‌محور محسوب می‌شود که در آن نقش‌های مختلف (تحلیل‌گر، طراح، توسعه‌دهنده، آزمون‌گر، مدیر پروژه و ...) و ارتباط آن‌ها با فعالیت‌ها و محصولات به‌طور دقیق مشخص شده است. از این نظر، USDP شفافیت بیشتری نسبت به Hodge-Mock دارد.

زبان مدل‌سازی

در Hodge-Mock، زبان مدل‌سازی خاصی الزام نشده است، اما مجموعه‌ای از مدل‌ها و نمودارهای اختصاصی (مانند OBD، OID، SBD، SBS، OCR، OD، ORD، ERD و OPD) به‌کار گرفته می‌شوند که در عمل یک زبان مدل‌سازی اختصاصی را شکل می‌دهند. استفاده از زبان‌های برنامه‌نویسی خاص نیز در این متدولوژی اجباری نیست.

در مقابل، USDP استفاده از UML را به‌عنوان زبان مدل‌سازی اصلی الزام می‌کند و استفاده از سایر زبان‌های مدل‌سازی مجاز نیست. این موضوع باعث استانداردسازی بالاتر مدل‌ها در USDP نسبت به Hodge-Mock می‌شود.

محصولات

در Hodge-Mock، تمامی محصولاتی که در جریان فعالیت‌ها ایجاد می‌شوند به‌صورت صریح به‌عنوان مصنوعات قابل تحویل معرفی شده‌اند. این محصولات شامل مدل‌های تحلیلی، رفتاری و طراحی بوده و در هر فاز تکمیل یا به‌روزرسانی می‌شوند.

در USDP نیز محصولات هر فاز به‌صورت دقیق مشخص شده‌اند. برای مثال:

- در فاز آغاز: سند چشم‌انداز، مدل اولیه Use-Case، توجیه اقتصادی و ارزیابی ریسک
- در فاز تفصیل: معماری پایه‌ی قابل اجرا، مدل‌های UML تکمیل‌شده و برنامه‌ی پروژه
- در فاز ساخت: محصول نرم‌افزاری پیکربندی‌شده، نتایج تست و مستندات
- در فاز انتقال: نسخه‌ی نهایی محصول، مستندات راهنمای کاربر و برنامه‌ی پشتیبانی

هر دو متدولوژی به محصولات توجه دارند، اما USDP آن‌ها را ساخت‌یافته‌تر دسته‌بندی می‌کند.

تکنیک‌ها و قواعد

در Hodge-Mock، فعالیت‌ها و اهداف آن‌ها مشخص شده‌اند، اما جزئیات مربوط به تکنیک‌های اجرایی و قواعد عملی به‌صورت محدود بیان شده و بیشتر به منابع تکمیلی ارجاع داده می‌شود.

در مقابل، USDP برای فعالیت‌های مختلف نظیر استخراج نیازمندی‌ها، طراحی معماری، مدیریت ریسک، آزمون و مدیریت پروژه، تکنیک‌ها و رویه‌های عملی مشخصی ارائه می‌دهد. از این نظر، USDP راهنمایی عملی‌تری نسبت به Hodge-Mock فراهم می‌کند.

موارد مربوط به فعالیت‌های چتری

در Hodge-Mock به فعالیت‌های چتری مانند مدیریت پروژه، تضمین کیفیت و مدیریت ریسک اشاره‌ی صریحی نشده است و این فعالیت‌ها بیشتر به‌صورت ضمنی در ساختار فازها حضور دارند.

در مقابل، USDP توجه ویژه‌ای به فعالیت‌های چتری دارد. مدیریت ریسک از فاز آغاز شروع شده، تضمین کیفیت از طریق تست و اعتبارسنجی مداوم انجام می‌شود و مدیریت پروژه در تمامی فازها نقش فعالی ایفا می‌کند. بنابراین USDP در این معیار عملکرد بهتری دارد.

چگونگی تعریف

تعریف Hodge-Mock بیشتر فرآیندمحور و متمرکز بر تحلیل چیستی سیستم است و دیدگاه غالب آن شیءگرایی و رفتار آبجکت‌ها می‌باشد.

در مقابل، USDP با ترکیب دیدگاه‌های فرآیندمحور، نقش‌محور و محصول‌محور، تعریف جامع‌تری از توسعه‌ی نرم‌افزار ارائه می‌دهد. به همین دلیل، در معیار چگونگی تعریف نیز USDP نسبت به Hodge-Mock کامل‌تر ارزیابی می‌شود.

معیار دوم: پوشش چرخه‌ی عمر ایجاد نرم‌افزار

چرخه‌ی عمر ایجاد نرم‌افزار در حالت عام شامل سه فاز کلی تعریف، ایجاد و مراقبت و نگهداری می‌باشد. در این بخش، نحوه‌ی پوشش این فازها در متدولوژی‌های Hodge-Mock و USDP مورد بررسی و مقایسه قرار می‌گیرد.

فاز تعریف (Definition)

کاوش قلمرو مسئله و مدل‌سازی آن

در Hodge-Mock، فاز تحلیل و فعالیت‌های وابسته به آن، بخش عمده‌ای از کاوش و مدل‌سازی قلمرو مسئله را پوشش می‌دهند. در این متدولوژی، با انجام فعالیت‌هایی مانند تحلیل نیازمندی‌ها، تحلیل اطلاعات و تحلیل رخدادها، مرز سیستم مشخص شده و قلمرو مسئله به‌تدریج کاهش می‌یابد. مدل‌سازی قلمرو مسئله عمدتا با شناسایی کلاس‌ها، روابط بین آن‌ها و رفتار سیستم انجام می‌شود.

در مقابل، USDP این فعالیت را از فاز آغاز شروع کرده و در فاز تفصیل به‌صورت کامل‌تری ادامه می‌دهد. در این متدولوژی، مدل‌سازی قلمرو مسئله به‌صورت تدریجی و تکرارشونده انجام شده و از مدل‌های متنوع UML برای نمایش جنبه‌های مختلف سیستم استفاده می‌شود. در نتیجه، USDP در این بخش پوشش ساخت‌یافته‌تری نسبت به Hodge-Mock دارد.

استخراج نیازمندی‌ها

در Hodge-Mock، اولین فعالیت تحلیلی، تحلیل نیازمندی‌هاست که در قالب فعالیت‌های فاز تحلیل انجام می‌شود. اگرچه این متدولوژی به فعالیتی مجزا و مستقل برای استخراج نیازمندی‌ها اشاره نمی‌کند، اما نیازمندی‌های سیستم در جریان تحلیل به‌صورت ضمنی شناسایی و مستند می‌شوند.

در مقابل، USDP استخراج نیازمندی‌ها را یکی از فعالیت‌های محوری خود می‌داند. در فاز آغاز، نیازمندی‌های اصلی استخراج شده و در فاز تفصیل تکمیل می‌شوند. تمامی نیازمندی‌ها در USDP مبتنی بر Use-Case تعریف شده و قابلیت ردیابی بالایی دارند. بنابراین، USDP در این بخش صریح‌تر و نظام‌مندتر عمل می‌کند.

تحلیل امکان‌پذیری

در Hodge-Mock فعالیت مشخص و مستقلی برای تحلیل امکان‌پذیری و بررسی ریسک‌های پروژه تعریف نشده است. تنها می‌توان گفت که در پایان فاز تحلیل و با ارزیابی سناریوهای طراحی، امکان‌پذیری سیستم به‌صورت غیرمستقیم مورد بررسی قرار می‌گیرد.

در مقابل، USDP تحلیل امکان‌پذیری را یکی از اهداف اصلی فاز آغاز می‌داند. در این فاز، ریسک‌های فنی، اقتصادی و زمانی شناسایی شده و با استفاده از نمونه‌های اولیه و تحلیل‌های فنی، تصمیم‌گیری درباره‌ی ادامه یا توقف پروژه انجام می‌شود. در پایان فاز تفصیل نیز انتظار می‌رود تمامی ریسک‌های اصلی برطرف شده باشند.

فاز ایجاد (Development)

طراحی معماری

در Hodge-Mock، طراحی معماری به‌صورت تدریجی و در جریان تحلیل و طراحی انجام می‌شود. در این متدولوژی، در مرحله‌ی گذار از تحلیل به طراحی، با ترسیم نمودار CSD و سپس استخراج مدل OID، ساختار کلی سیستم مشخص شده و معماری سیستم شکل می‌گیرد. این معماری در طول فازهای بعدی تکمیل و اصلاح می‌شود.

در مقابل، USDP یک متدولوژی معماری‌محور است. در فاز آغاز، یک معماری اولیه با هدف امکان‌سنجی ایجاد می‌شود و در فاز تفصیل، معماری پایه‌ی قابل اجرا (EAB) ساخته و تثبیت می‌گردد. در نتیجه، طراحی معماری در USDP جایگاه مشخص‌تر و رسمی‌تری نسبت به Hodge-Mock دارد.

طراحی تفصیلی

در Hodge-Mock، طراحی تفصیلی در فاز طراحی نرم‌افزار انجام می‌شود. در این مرحله، جزئیات بیشتری به مدل‌های قبلی افزوده شده و اطلاعات لازم برای پیاده‌سازی سیستم فراهم می‌شود. استفاده از مدل OPD برای بیان رفتار دقیق آبجکت‌ها از ویژگی‌های این فاز است.

در مقابل، در USDP طراحی تفصیلی بخشی از جریان کاری طراحی بوده و در فازهای تفصیل و ساخت انجام می‌شود. در این متدولوژی، طراحی Use-Caseهای دارای ریسک، ایجاد مدل‌های متعدد UML و تکمیل معماری سیستم از جمله فعالیت‌های اصلی طراحی تفصیلی محسوب می‌شوند.

برنامه‌نویسی

در Hodge-Mock توضیح صریحی درباره‌ی فعالیت برنامه‌نویسی ارائه نشده است و این فعالیت به‌صورت ضمنی پس از تکمیل طراحی نرم‌افزار در نظر گرفته می‌شود.

در مقابل، در USDP برنامه‌نویسی یکی از فعالیت‌های اصلی فاز ساخت است که طی تکرارهای متعدد انجام می‌شود. علاوه بر این، در فاز تفصیل نیز پیاده‌سازی نمونه‌های اولیه برای کاهش ریسک انجام می‌گیرد.

آزمون

در Hodge-Mock به آزمون به‌عنوان یکی از فازهای نهایی اشاره شده است، اما جزئیات فعالیت‌های آزمون به‌صورت دقیق تشریح نشده‌اند.

در مقابل، USDP آزمون را در تمامی فازها مدنظر قرار می‌دهد. در فاز تفصیل، معماری پایه آزمون می‌شود؛ در فاز ساخت، آزمون‌های مکرر انجام می‌گیرد؛ و در فاز انتقال، آزمون‌های آلفا، بتا و پذیرش اجرا می‌شوند.

استقرار

در Hodge-Mock اشاره‌ی صریحی به فعالیت استقرار سیستم نشده است.

در مقابل، در USDP استقرار بخشی از فاز انتقال محسوب می‌شود و شامل آماده‌سازی محیط کاربر، انتقال سیستم و انجام تنظیمات نهایی است.

فاز مراقبت و نگهداری (Maintenance)

در Hodge-Mock فعالیت‌های مشخصی برای مراقبت و نگهداری سیستم تعریف نشده است، اگرچه می‌توان از مستندات تولیدشده در فازهای قبلی برای پشتیبانی سیستم استفاده کرد.

در مقابل، USDP نیز فرآیند مستقلی برای نگهداری تعریف نکرده است، اما با استفاده از تکرار چرخه‌های ایجاد و اعمال تغییرات در نسخه‌های جدید، امکان پشتیبانی محدود از سیستم فراهم

می‌شود. با این حال، ماهیت تکرارشونده‌ی USDP برای نگهداری بلندمدت چندان مناسب نیست و معمولا نیاز به فرآیند مکمل وجود دارد.

مقایسه و نتیجه‌گیری

بر اساس تحلیل انجام‌شده، می‌توان نتیجه گرفت که USDP نسبت به Hodge-Mock پوشش کامل‌تری از چرخه‌ی عمر ایجاد نرم‌افزار ارائه می‌دهد؛ به‌ویژه در فازهای تعریف و ایجاد. Hodge-Mock تمرکز اصلی خود را بر تحلیل و طراحی قرار داده و در مراحل پیاده‌سازی، آزمون، استقرار و نگهداری پوشش محدودتری دارد.

در مجموع، USDP در این معیار عملکرد بهتری داشته و چرخه‌ی عمر ایجاد نرم‌افزار را به‌صورت ساخت‌یافته‌تر پوشش می‌دهد.

معیار سوم: پشتیبانی از فعالیت‌های چتری

فعالیت‌های چتری بخش مهمی از فرآیند ایجاد نرم‌افزار را در بر می‌گیرند که بر فعالیت‌هایی نظارت دارند که از تمامی فعالیت‌های چرخه‌ی حیات نرم‌افزار پشتیبانی می‌کنند. این فعالیت‌ها جزو فعالیت‌های ایجاد مستقیم محصول نبوده و عمدتا ماهیت مدیریتی دارند و تمامی فعالیت‌های چرخه‌ی حیات را می‌پوشانند. اساسی‌ترین فعالیت‌های چتری عبارت‌اند از:

1. فعالیت‌های مدیریت پروژه‌ای
2. فعالیت‌های ناظر بر مدیریت ریسک
3. فعالیت‌های ناظر بر تضمین کیفیت

در ادامه، این فعالیت‌ها در متدولوژی‌های Hodge-Mock و USDP بررسی می‌شوند.

مدیریت ریسک

در متدولوژی Hodge-Mock هیچ‌یک از تکنیک‌های رایج مدیریت ریسک نظیر ساخت نمونه‌ی اولیه، برنامه‌ریزی مبتنی بر ریسک، دخالت فعالانه‌ی کاربر، ترخیص زودهنگام، یکپارچه‌سازی مداوم و مشارکت مستمر کاربر به‌صورت صریح بیان نشده‌اند. همچنین برنامه‌ریزی بر اساس ریسک در این متدولوژی دیده نمی‌شود. با این حال، تحلیل مقدماتی به‌عنوان یکی از مصادیق مدیریت ریسک در این متدولوژی وجود دارد. اگرچه ارزیابی ریسک به‌صورت یک فعالیت مستقل و صریح تعریف نشده است، اما این

متدولوژی سازوکارهایی برای کاهش ریسک در اختیار دارد. از جمله‌ی این سازوکارها می‌توان به وجود ملاک‌ها و سناریوهای ارزیابی در پایان هر فاز، رویکرد تکرارشونده–افزایشی (Iterative–Incremental) و امکان بازگشت به فازهای قبلی در صورت شناسایی خطا اشاره کرد. علاوه بر این، صحت‌سنجی و اعتبارسنجی مستمر و پایش مداوم برنامه‌ها، فرآیندها و مصنوعات در طول چرخه‌ی حیات انجام می‌شود که به شناسایی زودهنگام مشکلات کمک می‌کند.

در مقابل، یکپارچه‌سازی مستمر و مشارکت فعالانه‌ی کاربر در این متدولوژی پشتیبانی نشده‌اند.

در متدولوژی USDP، مدیریت ریسک یکی از ارکان اصلی فرآیند محسوب می‌شود. در فاز آغاز، ریسک‌های بحرانی شناسایی شده و تحلیل امکان‌پذیری انجام می‌گیرد. از تکنیک ساخت نمونه‌ی اولیه برای کاهش ریسک استفاده می‌شود و در فاز تفصیل سند ارزیابی ریسک تولید می‌گردد. نیازمندی‌های دارای ریسک طراحی، پیاده‌سازی و آزمون می‌شوند و انتظار می‌رود تا پایان فاز تفصیل تمامی ریسک‌های اصلی پوشش داده شده باشند. ماهیت تکراری–افزایشی USDP نیز نقش موثری در کاهش تدریجی ریسک‌ها ایفا می‌کند.

مدیریت پروژه

در متدولوژی Hodge-Mock به‌صورت صریح به فعالیت‌های مرتبط با مدیریت پروژه پرداخته نشده است و جریان کاری مشخصی برای برنامه‌ریزی، زمان‌بندی و نظارت و کنترل پروژه ارائه نمی‌شود.

در مقابل، در متدولوژی USDP اگرچه برخلاف RUP جریان کاری مستقلی برای مدیریت پروژه تعریف نشده است، اما این فعالیت‌ها در طول فازها انجام می‌شوند. در فاز آغاز، فعالیت‌هایی نظیر برنامه‌ریزی پروژه، ایجاد نمودار گانت و تخمین اقتصادی انجام می‌شود. در فاز تفصیل، برنامه‌ی پروژه دقیق‌تر شده و برای تکرارهای فاز ساخت برنامه‌ریزی تفصیلی صورت می‌گیرد. برنامه‌ها در طول اجرای پروژه به‌صورت مستمر بازبینی و اصلاح می‌شوند.

تضمین کیفیت

تضمین کیفیت یکی از مهم‌ترین فعالیت‌های چتری است که باید در تمام طول چرخه‌ی عمر نرم‌افزار مورد توجه قرار گیرد.

در متدولوژی Hodge-Mock، تضمین کیفیت از طریق مرور و بازبینی مکرر مصنوعات و وجود سناریوهای اعتبارسنجی در پایان هر فاز انجام می‌شود. همچنین صحت‌سنجی و اعتبارسنجی مستمر در طول چرخه‌ی حیات وجود دارد که نقش مهمی در شناسایی زودهنگام خطاها ایفا می‌کند. در این

متدولوژی به Design By Contract اشاره نشده است و مکانیزم مشخصی برای تضمین کیفیت مبتنی بر قرارداد تعریف نشده است.

در مقابل، در متدولوژی USDP تضمین کیفیت به‌صورت ساخت‌یافته‌تری دنبال می‌شود. فرآیند تکراری–افزایشی باعث می‌شود در هر تکرار فعالیت‌های صحت‌سنجی و اعتبارسنجی انجام شود. از آن‌جا که USDP مبتنی بر Use-Case است، رهگیری مستقیم نیازمندی‌ها تا طراحی، پیاده‌سازی و آزمون امکان‌پذیر می‌شود. همچنین در فاز تفصیل معیارهای کیفیتی نظیر نرخ کشف اشکال و تراکم اشکال تعریف شده و پایش می‌شوند که در تضمین کیفیت محصول نهایی موثر است.

مقایسه و نتیجه‌گیری

بر اساس بررسی انجام‌شده، می‌توان نتیجه گرفت که متدولوژی USDP در پشتیبانی از فعالیت‌های چتری، به‌ویژه در حوزه‌های مدیریت ریسک، مدیریت پروژه و تضمین کیفیت، عملکرد کامل‌تر و ساخت‌یافته‌تری نسبت به Hodge-Mock دارد.

در حالی که Hodge-Mock فعالیت‌های چتری را به‌صورت ضمنی و غیرصریح پوشش می‌دهد و تمرکز آن بیشتر بر بازبینی فازبه‌فاز و کاهش ریسک از طریق اعتبارسنجی است، USDP این فعالیت‌ها را به‌صورت صریح، تکرارشونده و پیوسته در سراسر چرخه‌ی عمر نرم‌افزار دنبال می‌کند.

معیار چهارم: بی‌درزی و همواری انتقال

یکی از اهداف متدولوژی‌ها، بی‌درز و هموار بودن انتقال بین فازها، فعالیت‌ها و مصنوعات است. بی‌درزی انتقال به معنای حفظ پیوستگی مفهومی و اطلاعاتی بین خروجی‌های مراحل مختلف بوده و همواری انتقال نیز به معنای جلوگیری از ایجاد گسست ناگهانی و تولید مصنوعات کاملا جدید و مستقل از مصنوعات قبلی است. وجود عدم بی‌درزی و ناهمواری در انتقال می‌تواند منجر به از بین رفتن یا تضعیف بخشی از اطلاعات و مفاهیم کلیدی در فرآیند ایجاد نرم‌افزار شود.

در متدولوژی Hodge-Mock، در اغلب فعالیت‌ها از زبان و مفاهیم مدل‌سازی خاص هر فاز استفاده می‌شود. این متدولوژی با شناسایی تدریجی کلاس‌ها و ایجاد مدل‌های مبتنی بر آن‌ها تلاش می‌کند تا حدی بی‌درزی را حفظ نماید. با این حال، در برخی گذارها، به‌ویژه در فعالیت‌های تحلیل، مقداری درز انتقال مشاهده می‌شود. در این متدولوژی، گذار از مستندات نیازمندی مبتنی بر تحلیل اطلاعات و مدل‌هایی مانند ERD به مدل‌های شی‌گرا، منجر به شکاف پارادایمی در مدل‌سازی می‌شود. در نتیجه، در برخی موارد شاهد تولید مصنوعات کاملا جدید هستیم که ارتباط آن‌ها با مصنوعات قبلی

به‌صورت مستقیم و هموار برقرار نشده است. همچنین جابجایی بین فعالیت‌ها و فازها به‌طور کامل هموار نیست. با این حال، در Hodge-Mock تلاش شده است که مدل‌های ساخته‌شده در فازهای پیشین به‌صورت تدریجی بازبینی و به‌روزرسانی شوند و مدل‌های جدید نیز بر اساس اطلاعات به‌دست‌آمده از مراحل قبلی ایجاد گردند. این موضوع باعث می‌شود تا حدی شکاف پارادایمی پر شده و انتقال اطلاعات به‌صورت طبیعی‌تری انجام شود. بنابراین، اگرچه Hodge-Mock کاملا بی‌درز و هموار نیست، اما تلاش‌هایی برای کاهش درز انتقال در آن مشاهده می‌شود.

متدولوژی USDP به‌صورت Use-Case Driven عمل می‌کند و تمامی فعالیت‌ها و مصنوعات حول Use-Caseها پیش می‌روند. این رویکرد باعث می‌شود تمامی فعالیت‌ها و محصولات از یک مفهوم مشترک تبعیت کرده و انتقال مفاهیم بین فازها با حداقل گسست انجام شود. در USDP، اگرچه ماهیت Use-Caseها با برخی از نمودارهای UML متفاوت است، اما ارتباط منطقی و قابل رهگیری بین آن‌ها وجود دارد. به‌عنوان مثال، نمودارهای توالی و نمودار فعالیت با استفاده از مفاهیم مشترک، به یکدیگر نگاشت می‌شوند که این امر به کنترل درز انتقال کمک می‌کند. همچنین فرآیند USDP به‌صورت تکراری–افزایشی بوده و فعالیت‌ها در بازه‌های زمانی کوتاه انجام می‌شوند. استفاده‌ی گسترده از UML در تمامی فازها باعث می‌شود زبان مدل‌سازی یکپارچه‌ای در کل فرآیند حفظ شده و مدل‌ها در طول تکرارها تکامل یابند. این ویژگی‌ها موجب می‌شود میزان گسست انتقال در USDP حداقل بوده و انتقال بین فازها به‌صورت هموار انجام شود.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی Hodge-Mock و USDP تا حدی به بی‌درزی و همواری انتقال توجه داشته‌اند. در Hodge-Mock، اگرچه تلاش برای تکامل تدریجی مدل‌ها وجود دارد، اما به دلیل شکاف پارادایمی در گذار بین برخی مدل‌ها و تولید مصنوعات کاملا جدید، بی‌درزی و همواری انتقال به‌طور کامل محقق نشده است. در مقابل، متدولوژی USDP به دلیل Use-Case Driven بودن، استفاده از زبان مدل‌سازی یکپارچه و فرآیند تکراری–افزایشی، انتقالی هموارتر و با درز کمتر بین فعالیت‌ها و فازها فراهم می‌کند. در مجموع، می‌توان نتیجه گرفت که USDP از منظر بی‌درزی و همواری انتقال وضعیت بهتری نسبت به Hodge-Mock دارد، هرچند هر دو متدولوژی تلاش‌هایی برای حفظ پیوستگی مفهومی و اطلاعاتی در فرآیند ایجاد نرم‌افزار انجام داده‌اند.

معیار پنجم: مبتنی بودن بر نیازمندی‌ها

در متدولوژی Hodge-Mock، نیازمندی‌ها در ابتدای فرآیند ایجاد نرم‌افزار گردآوری و ضبط می‌شوند. این نیازمندی‌ها به‌صورت مستقل مدل‌سازی شده و به‌عنوان یکی از محصولات اصلی فازهای ابتدایی فرآیند در نظر گرفته می‌شوند. در ادامه، نیازمندی‌های استخراج‌شده به‌عنوان مبنایی برای مراحل

بعدی ایجاد نرم‌افزار مورد استفاده قرار می‌گیرند و فعالیت‌های بعدی بر اساس این مبنا پیش می‌روند. بدین ترتیب، نقش نیازمندی‌ها در Hodge-Mock بیشتر به‌عنوان نقطه‌ی شروع و مرجع اولیه برای ادامه‌ی فرآیند تعریف می‌شود و چارچوب کلی فعالیت‌های بعدی بر پایه‌ی آن‌ها شکل می‌گیرد.

در مقابل، متدولوژی USDP به‌صورت Use-Case Driven عمل می‌کند و در آن، مبنای تمامی فعالیت‌ها نیازمندی‌ها هستند. در این متدولوژی، رهگیری مستقیم نیازمندی‌ها در طول فرآیند وجود دارد و فعالیت‌ها به‌گونه‌ای سازمان‌دهی می‌شوند که همواره به نیازمندی‌ها قابل ارجاع باشند. علاوه بر این، نیازمندی‌ها به‌صورت تدریجی و در طول فازهای مختلف استخراج می‌شوند؛ به‌طوری که در فاز آغاز حدود ۲۰ درصد، در فاز تفصیل تا ۸۰ درصد و مابقی در فاز ساخت استخراج می‌گردند و سیستم بر اساس این نیازمندی‌ها ساخته می‌شود. این رویکرد باعث می‌شود نیازمندی‌ها نه‌تنها در ابتدای فرآیند، بلکه در سراسر چرخه‌ی ایجاد نرم‌افزار نقش فعال و تعیین‌کننده‌ای داشته باشند.

در متدولوژی Hodge-Mock، نیازمندی‌ها در ابتدای فرآیند استخراج و به‌عنوان مبنایی برای مراحل بعدی استفاده می‌شوند، اما نقش آن‌ها عمدتا به همان مراحل ابتدایی محدود است. در مقابل، USDP به‌صورت Use-Case Driven بوده و نیازمندی‌ها به‌طور مستقیم و مستمر در تمامی فعالیت‌ها و فازها نقش محوری دارند. بنابراین، USDP در این معیار عملکرد بهتری نسبت به Hodge-Mock دارد.

معیار ششم: ارزیابی محصولات متدولوژی

این معیار مشتمل است بر ۳ معیار ریزدانه که همه‌ی آن‌ها ناظر بر محصولات متدولوژی هستند؛ این سه معیار عبارتند از:

1. قابل آزمون بودن محصولات
2. ملموس بدون محصولات
3. قابلیت رهگیری محصولات به نیازمندی‌ها

قابل آزمون بودن محصولات

برای آن‌که محصولات یک متدولوژی قابل آزمون باشند، لازم است دارای ویژگی‌هایی از جمله تعداد قابل کنترل مصنوعات، پیچیدگی قابل مدیریت، وابستگی شفاف بین مصنوعات و مکمل بودن مدل‌ها باشند. افزایش بیش از حد مصنوعات، پیچیدگی زیاد یا وابستگی‌های مبهم می‌تواند فرآیند آزمون را دشوار کرده و قابلیت آزمون‌پذیری محصولات را کاهش دهد.

در متدولوژی Hodge-Mock، تعداد مصنوعات نسبتا زیاد است، اما هر یک از این مصنوعات به‌تنهایی در سطح پیچیدگی بالایی قرار ندارند. مصنوعات به‌صورت تدریجی ایجاد شده و توضیحات آن‌ها در طول فازهای مختلف تکمیل می‌شود. اگرچه بین مصنوعات وابستگی وجود دارد، این وابستگی‌ها معمولا مشخص بوده و مدل‌ها بیشتر نقش تکمیل‌کننده‌ی یکدیگر را ایفا می‌کنند و جایگزین هم نمی‌شوند. در نتیجه، با وجود تعدد مصنوعات، امکان بررسی و آزمون آن‌ها به‌صورت جداگانه تا حد قابل قبولی فراهم است.

در مقابل، در متدولوژی USDP، مدل‌سازی از پیچیدگی بالاتری برخوردار است و مدل‌های متعددی در فازهای مختلف ایجاد می‌شوند. این پیچیدگی و تکثر مدل‌ها، اگرچه به غنای تحلیل و طراحی کمک می‌کند، اما در برخی موارد باعث می‌شود تکمیل و انسجام سایر مدل‌ها دشوار شود. همچنین وابستگی زیاد بین مدل‌ها و نیاز به هماهنگی مستمر میان آن‌ها می‌تواند فرآیند آزمون را پیچیده‌تر کرده و در نتیجه، قابلیت آزمون‌پذیری مصنوعات را کاهش دهد.

با توجه به بررسی انجام‌شده، می‌توان گفت که متدولوژی Hodge-Mock به دلیل ساده‌تر بودن هر یک از مصنوعات، امکان آزمون‌پذیری مناسب‌تری را فراهم می‌کند، هرچند تعداد مصنوعات در آن بیشتر است. در مقابل، در متدولوژی USDP با وجود ساخت‌یافتگی بالاتر، پیچیدگی و وابستگی زیاد بین مدل‌ها می‌تواند فرآیند آزمون را دشوارتر سازد. بنابراین، از منظر قابل آزمون بودن محصولات، Hodge-Mock وضعیت مناسب‌تری نسبت به USDP دارد.

ملموس بودن محصولات

منظور از ملموس بودن محصولات، میزان قابل درک بودن و قابل ارتباط بودن مصنوعات تولیدشده برای ذی‌نفعان مختلف، به‌ویژه کاربران نهایی و مهندسان و ایجادکنندگان نرم‌افزار است. محصول ملموس محصولی است که مخاطب بتواند نقش، کاربرد و جایگاه آن را در فرآیند ایجاد نرم‌افزار به‌خوبی درک کند.

در متدولوژی Hodge-Mock، از دیدگاه کاربران نهایی، محصولات قابل اجرا در جریان متدولوژی تولید نمی‌شوند و کاربر مستقیما با نرم‌افزار اجرایی در طول فرآیند مواجه نیست. با این حال، مصنوعات تولیدشده از نظر نحو (Syntax) و معنا (Semantic) می‌توانند تطابق مناسبی با دامنه‌ی مسئله داشته باشند. به‌ویژه برخی نمودارها مانند نمودارهای کلاینت–سرور و سایر مصنوعات تحلیلی، برای برنامه‌نویسان و مخاطبان فنی ملموس بوده و درک مناسبی از ساختار سیستم ارائه می‌دهند. از منظر مهندسان و ایجادکنندگان نرم‌افزار، محصولات Hodge-Mock دارای کاربردی مشخص، بدون ابهام و تعریف‌شده هستند و جایگاه هر یک از آن‌ها در فرآیند ایجاد نرم‌افزار به‌صورت دقیق مشخص شده

است. این ویژگی باعث می‌شود مهندسان بتوانند نقش هر محصول را در پیشبرد فرآیند به‌خوبی تشخیص دهند و از آن به‌صورت هدفمند استفاده کنند.

در متدولوژی USDP، در طول فازهای مختلف، محصولاتی نظیر سند چشم‌انداز، سند توجیه اقتصادی و سند ارزیابی ریسک تولید می‌شوند که ساختاری ساده داشته و برای کاربران نهایی قابل درک و ملموس هستند. همچنین این اسناد معمولا نیازمند تایید کاربر بوده و نقش مستقیمی در تعامل با ذی‌نفعان ایفا می‌کنند. افزون بر این، برخی مدل‌های UML مانند نمودار کلاس در فاز تحلیل برای کاربران قابل درک هستند؛ در حالی که برخی دیگر، مانند نمودار کلاس در فاز طراحی که شامل جزئیات پیاده‌سازی است، برای کاربران نهایی ملموس نیستند. از دیدگاه ایجادکنندگان و مهندسان نرم‌افزار، مصنوعات و مدل‌های USDP در صورتی که مکمل یکدیگر باشند، صرفا جنبه‌ی تزئینی یا هم‌ریخت نداشته باشند و در راستای انجام فعالیت‌های فرآیند ایجاد نرم‌افزار به کار روند، کاملا ملموس و قابل درک هستند. البته با توجه به سنگین بودن این متدولوژی، استفاده از USDP در پروژه‌هایی که متناسب با این سطح از پیچیدگی باشند، باعث می‌شود محصولات آن برای مهندسان معنادارتر و ملموس‌تر تلقی شوند.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی Hodge-Mock و USDP تا حد خوبی دارای محصولات ملموس هستند. در Hodge-Mock، مصنوعات برای مهندسان و ایجادکنندگان نرم‌افزار به دلیل کاربرد مشخص و جایگاه دقیق در فرآیند، ملموس‌تر هستند، هرچند برای کاربران نهایی به دلیل نبود محصولات اجرایی، میزان ملموس بودن محدودتر است. در مقابل، USDP با تولید اسناد قابل فهم برای کاربران و مدل‌های ساخت‌یافته برای مهندسان، دامنه‌ی گسترده‌تری از ذی‌نفعان را پوشش می‌دهد. در مجموع، هر دو متدولوژی از منظر ملموس بودن محصولات عملکرد قابل قبولی دارند، اما نوع مخاطب در میزان درک و ملموس بودن محصولات نقش تعیین‌کننده‌ای دارد.

قابلیت رهگیری به نیازمندی‌ها

قابلیت رهگیری نیازمندی‌ها به این معناست که کلیه‌ی محصولات و مصنوعات تولیدشده در یک متدولوژی باید به‌صورت شفاف به تک‌تک نیازمندی‌ها قابل ربط و رهگیری باشند. این ویژگی امکان بررسی تحقق نیازمندی‌ها، کنترل تغییرات و ارزیابی صحت فرآیند ایجاد نرم‌افزار را فراهم می‌سازد.

در متدولوژی Hodge-Mock، نیازمندی‌ها مبنای هر یک از گام‌های فرآیند محسوب می‌شوند و این متدولوژی در بستری تکرارشونده–افزایشی (Iterative–Incremental) و مبتنی بر نیازمندی عمل می‌کند. در این رویکرد، سناریوها نقش مهمی در توصیف رفتار سیستم دارند و چندین مدل برای نمایش رفتار و کارکرد سیستم مورد استفاده قرار می‌گیرد. با این حال، اگرچه نیازمندی‌ها در مراحل مختلف

فرآیند حضور دارند، تحقق دقیق هر نیازمندی در قالب زبان مدل‌سازی متدولوژی و از طریق سناریوها به‌صورت صریح و قابل رهگیری مشخص نشده است. به بیان دیگر، ارتباط مستقیم و شفاف بین هر نیازمندی و مصنوعات نهایی به‌طور کامل قابل ردیابی نیست.

در مقابل، متدولوژی USDP به‌صورت Use-Case Driven عمل می‌کند و تمامی فعالیت‌ها، مدل‌ها و مصنوعات بر مبنای Use-Caseها شکل می‌گیرند. از آن‌جا که Use-Caseها نمایش‌دهنده‌ی نیازمندی‌ها هستند، رهگیری مستقیم نیازمندی‌ها در سراسر فرآیند وجود دارد و هر فعالیت، مدل و محصول به‌راحتی به نیازمندی متناظر خود قابل ارتباط است. این ویژگی باعث می‌شود کنترل تحقق نیازمندی‌ها و بررسی تغییرات آن‌ها به‌صورت ساخت‌یافته انجام شود.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی به نیازمندی‌ها توجه دارند، اما USDP از نظر قابلیت رهگیری نیازمندی‌ها عملکرد بهتری نسبت به Hodge-Mock دارد. در حالی که Hodge-Mock نیازمندی‌ها را به‌عنوان مبنای فرآیند در نظر می‌گیرد، رهگیری صریح و مستقیم آن‌ها در محصولات نهایی به‌طور کامل مشخص نیست؛ در مقابل، USDP به دلیل Use-Case Driven بودن، رهگیری شفاف و مستقیم نیازمندی‌ها را در تمامی فعالیت‌ها و مصنوعات فراهم می‌کند.

معیار هفتم: قابلیت جذب مشارکت فعالانه‌ی کاربر

جذب مشارکت فعالانه‌ی کاربر ناظر بر میزان حضور و نقش‌آفرینی کاربر در فرآیند برنامه‌ریزی، بازنگری و ارزیابی ایجاد نرم‌افزار است و یکی از اصول اساسی رویکردهای چابک محسوب می‌شود. این مشارکت معمولا از دو روش اصلی محقق می‌گردد:

1. حضور نماینده‌ای از مشتری در تیم ایجاد نرم‌افزار
2. برگزاری منظم جلسات برنامه‌ریزی و بازنگری با مشارکت کاربر

در متدولوژی Hodge-Mock، نماینده‌ای از مشتری به‌صورت رسمی در تیم ایجاد نرم‌افزار حضور ندارد. اگرچه در جریان فرآیند، جلسات منظم برنامه‌ریزی و بازنگری برگزار می‌شود، اما نماینده‌ی مشتری امکان حضور فعال در این جلسات را ندارد. در نتیجه، تصمیم‌گیری‌ها و بازنگری‌ها عمدتا توسط تیم فنی انجام می‌شود و نقش کاربر در فرآیند ایجاد نرم‌افزار محدود باقی می‌ماند. از این منظر، Hodge-Mock شباهت چندانی به متدولوژی‌های چابک ندارد و میزان مشارکت فعالانه‌ی کاربران در آن پایین ارزیابی می‌شود.

در مقابل، در متدولوژی USDP مشارکت کاربر و ذی‌نفعان در طول فازهای مختلف فرآیند به‌صورت مشخص دیده می‌شود. محصولات ایجادشده در هر فاز برای دریافت تایید به ذی‌نفعان ارائه می‌گردند. در فاز آغاز، اهداف پروژه، نیازمندی‌های اساسی، حوزه‌ی سیستم و تخمین هزینه و زمان با

کاربر مطرح و تایید می‌شوند. در فاز تفصیل، توجیه اقتصادی و برنامه‌ی پروژه مورد بازنگری و تایید قرار می‌گیرد. در فاز ساخت، محصول قابل ارائه به کاربر نمایش داده شده و بازخورد دریافت می‌شود و در نهایت، در فاز انتقال پس از نصب و انجام تمهیدات لازم، تایید نهایی کاربر اخذ می‌گردد. بدین ترتیب، مشارکت کاربر به‌صورت تدریجی و مستمر در طول فرآیند USDP مشاهده می‌شود.

بر اساس بررسی انجام‌شده، می‌توان گفت که USDP از نظر قابلیت جذب مشارکت فعالانه‌ی کاربر عملکرد بهتری نسبت به Hodge-Mock دارد. در حالی که در Hodge-Mock مشارکت کاربر محدود بوده و حضور مستقیمی در فرآیند تصمیم‌گیری و بازنگری ندارد، در USDP با ارائه‌ی منظم محصولات و اخذ تایید در فازهای مختلف، مشارکت کاربر و ذی‌نفعان به‌صورت پیوسته و موثر برقرار می‌شود.

معیار هشتم: قابلیت اجرا و قابلیت اجرا به‌صورت کارا

قابلیت اجرا

قابلیت اجرا ناظر بر میزان امکان‌پذیری به‌کارگیری یک متدولوژی در پروژه‌های واقعی و عملی است و عواملی مانند پیچیدگی فرآیند، تعداد مصنوعات و میزان انطباق‌پذیری با پروژه‌های مختلف را در بر می‌گیرد.

در متدولوژی Hodge-Mock، فرآیند نسبتا پیچیده بوده و تعداد مصنوعات تولیدشده زیاد است. با این حال، این متدولوژی قابلیت پشتیبانی از پروژه‌های پیچیده را دارد و به‌طور خاص برای سیستم‌های کنترل ترافیک هوایی طراحی شده است. این ویژگی نشان می‌دهد که Hodge-Mock از نظر تئوریک و ساختاری قابلیت اجرا در پروژه‌های بزرگ و حساس را داراست، هرچند اجرای آن نیازمند توان فنی و سازمانی مناسب است.

در مقابل، متدولوژی USDP یک متدولوژی سنگین‌وزن محسوب می‌شود که فرآیندی پیچیده دارد و در طی آن مدل‌ها و مستندات مفصل و متعددی تولید می‌شوند. این موضوع باعث می‌شود اجرای USDP به همان شکلی که تعریف شده است، در عمل دشوار باشد و از این منظر، عملکرد آن در معیار قابلیت اجرا کاهش یابد. به همین دلیل، اجرای USDP معمولا نیازمند شخصی‌سازی متناسب با پروژه است که خود فرآیندی ساده نبوده و نیازمند تجربه و مهارت بالاست.

قابلیت اجرا به‌صورت کارا

قابلیت اجرا به‌صورت کارا به میزان بهره‌وری، تمرکز تیم ایجاد نرم‌افزار و حداقل‌سازی عوامل مزاحم در طول اجرای متدولوژی اشاره دارد.

در متدولوژی Hodge-Mock، به‌جز فاز اول، واحدهای کاری پیچیدگی بالایی ندارند و این موضوع به اجرای روان‌تر فرآیند کمک می‌کند. همچنین عواملی که موجب پرت شدن تمرکز ایجادکنندگان شوند در این متدولوژی مشاهده نمی‌شود و اعمال تغییرات خطاخیز در جریان ایجاد نرم‌افزار انجام نمی‌گیرد. Hodge-Mock به فناوری یا ابزار خاصی وابسته نیست که این موضوع انعطاف‌پذیری اجرای آن را افزایش می‌دهد. با این حال، این متدولوژی از نبود یک استراتژی مناسب مدیریت پروژه رنج می‌برد که می‌تواند بر کارایی اجرای آن در پروژه‌های بزرگ تاثیر منفی بگذارد.

در متدولوژی USDP، Use-Case Driven بودن فرآیند و مبتنی بودن مدل‌ها بر نیازمندی‌ها به حفظ تمرکز ایجادکنندگان نرم‌افزار کمک می‌کند. همچنین وجود مدل‌های معماری سیستم و معماری مبنای قابل اجرا نقش مهمی در هدایت فعالیت‌های تیم توسعه دارد. با این وجود، همان‌طور که اشاره شد، فرآیند USDP پیچیده است و برای استفاده‌ی موثر نیاز به شخصی‌سازی متناسب با پروژه هدف دارد. این متدولوژی به ابزار یا فناوری خاصی وابسته نیست، اما در حوزه‌ی مدل‌سازی وابستگی و تعصب شدیدی به UML دارد و در نتیجه، ضعف‌های این زبان مدل‌سازی در خود متدولوژی نیز منعکس می‌شود. در مقابل، USDP از استراتژی‌های نسبتا مناسب مدیریت پروژه برخوردار است که می‌تواند به بهبود کارایی اجرا کمک کند.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی Hodge-Mock و USDP از نظر قابلیت اجرا، توان پشتیبانی از پروژه‌های پیچیده را دارند، اما اجرای آن‌ها بدون چالش نیست. Hodge-Mock اگرچه دارای فرآیند و مصنوعات زیاد است، اما برای پروژه‌های خاص و پیچیده طراحی شده و از نظر کارایی اجرا در واحدهای کاری ساده‌تر، وضعیت قابل قبولی دارد. در مقابل، USDP به دلیل سنگین‌وزن بودن و نیاز به شخصی‌سازی، اجرای دشوارتری دارد، اما بهره‌مندی از رویکرد Use-Case Driven و استراتژی‌های مدیریت پروژه، قابلیت اجرای کارای آن را در صورت استفاده‌ی صحیح افزایش می‌دهد.

معیار نهم: قابلیت مدیریت پیچیدگی

قابلیت مدیریت پیچیدگی ناظر بر آن است که واحدهای کاری یک متدولوژی تا چه حد امکان بخش‌بندی و لایه‌بندی دارند. بخش‌بندی به معنای امکان استفاده‌ی مستقل از اجزای مختلف فرآیند بوده و لایه‌بندی نیز به معنای سازمان‌دهی فعالیت‌ها و مصنوعات در سطوح مختلف انتزاع است؛ به‌گونه‌ای که بتوان ساختارهای بزرگ را به اجزای کوچک‌تر و قابل مدیریت‌تر تقسیم نمود.

در متدولوژی Hodge-Mock، مکانیزم‌های مشخصی برای بخش‌بندی و لایه‌بندی فعالیت‌ها و مصنوعات وجود دارد. به‌طور خاص، تفکیک طراحی سیستم از طراحی نرم‌افزار موجب می‌شود پیچیدگی در سطوح مختلف انتزاع مدیریت شود و هر بخش به‌صورت مستقل مورد بررسی قرار گیرد. این تفکیک باعث می‌شود تحلیل مسائل در لایه‌های مختلف انجام شده و از انباشت پیچیدگی در یک مرحله جلوگیری گردد. از این رو، Hodge-Mock از منظر مدیریت پیچیدگی وضعیت مناسبی دارد.

در متدولوژی USDP، فرآیند ایجاد نرم‌افزار از چهار فاز آغاز، تفصیل، ساخت و انتقال تشکیل شده است و در هر یک از این فازها، چندین تکرار انجام می‌شود. هر تکرار شامل جریان‌های کاری نیازمندی‌ها، تحلیل، طراحی، پیاده‌سازی و آزمون است. وجود این ساختار باعث می‌شود سطوح مختلف انتزاع در فازها و تکرارها دیده شود که نشان‌دهنده‌ی استفاده از تکنیک لایه‌بندی است. همچنین انجام چندین جریان کاری در هر تکرار و تقسیم چرخه‌ی حیات به اجزای مختلف، بیان‌گر بخش‌بندی مناسب فعالیت‌ها در این متدولوژی است. علاوه بر این، اجرای فرآیند به‌صورت تکرارشونده–افزایشی به کنترل و مدیریت تدریجی پیچیدگی کمک می‌کند.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی Hodge-Mock و USDP از منظر قابلیت مدیریت پیچیدگی عملکرد مناسبی دارند. Hodge-Mock با تفکیک روشن بین طراحی سیستم و طراحی نرم‌افزار، مدیریت پیچیدگی را در سطوح مختلف تسهیل می‌کند. در مقابل، USDP با استفاده از ساختار فازی، تکرارها، جریان‌های کاری متعدد و رویکرد تکرارشونده–افزایشی، امکان بخش‌بندی و لایه‌بندی موثری را برای کنترل پیچیدگی فراهم می‌آورد.

معیار دهم: قابلیت‌های گسترش، مقیاس پذیری، پیکربندی و انعطاف

قابلیت گسترش

گسترش‌پذیری ناظر بر آن است که آیا می‌توان به فرآیند یک متدولوژی، قابلیت‌ها یا فعالیت‌های جدیدی اضافه کرد تا متدولوژی برای پروژه‌های خاص یا خاص‌منظوره متناسب‌سازی شود. وجود یک

هسته‌ی قابل توسعه و نقاط مشخص برای افزودن قابلیت‌ها، از شاخص‌های مهم گسترش‌پذیری محسوب می‌شود.

در متدولوژی Hodge-Mock، فرآیند به‌صورت یک هسته‌ی قابل توسعه تعریف نشده است و نقاط مشخصی برای گسترش یا افزودن قابلیت‌های جدید در آن در نظر گرفته نشده‌اند. ساختار فرآیند به‌گونه‌ای است که امکان افزودن فعالیت‌ها یا سازوکارهای جدید به‌صورت نظام‌مند و برنامه‌ریزی‌شده وجود ندارد. از این رو، تطبیق این متدولوژی با پروژه‌های خاص یا خاص‌منظوره از منظر گسترش‌پذیری با محدودیت مواجه است.

در متدولوژی USDP نیز اگرچه فرآیند بسیار بزرگ و پیچیده‌ای ارائه شده است، اما این متدولوژی به‌صورت ذاتی به شکل یک هسته‌ی قابل گسترش طراحی نشده است. برای استفاده‌ی کارا از USDP معمولا لازم است فرآیند از طریق روش‌هایی مانند هرس کردن فعالیت‌ها یا تجمیع برخی اجزا متناسب با پروژه‌ی هدف بازآرایی شود. با این حال، USDP نقاط گسترش مشخص یا مکانیزم‌های رسمی برای افزودن قابلیت‌های جدید تعریف نکرده است و تغییرات اعمال‌شده بیشتر به‌صورت انتخاب و حذف اجزای موجود انجام می‌گیرد تا توسعه‌ی ساخت‌یافته‌ی فرآیند.

بر اساس بررسی انجام‌شده، می‌توان گفت که هیچ‌یک از دو متدولوژی Hodge-Mock و USDP از منظر گسترش‌پذیری وضعیت مطلوبی ندارند. Hodge-Mock فاقد هسته و نقاط توسعه‌ی مشخص است و USDP نیز علی‌رغم پیچیدگی و گستردگی، مکانیزم رسمی برای گسترش فرآیند ارائه نمی‌دهد و بیشتر نیازمند تطبیق از طریق حذف یا تجمیع اجزاست. بنابراین، هر دو متدولوژی در این معیار با محدودیت‌های قابل توجهی مواجه هستند.

مقیاس‌پذیری

یک متدولوژی را مقیاس‌پذیر می‌دانیم اگر بتواند در پروژه‌هایی با اندازه‌های مختلف و همچنین در پروژه‌هایی با سطوح بحرانیت متفاوت به‌کار گرفته شود. یکی دیگر از عوامل موثر بر مقیاس‌پذیری، میزان و کیفیت مدل‌سازی در متدولوژی است؛ به‌گونه‌ای که هرچه مدل‌های تولیدشده بیشتر و غنی‌تر باشند، امکان به‌کارگیری متدولوژی در پروژه‌های بزرگ‌تر و پیچیده‌تر افزایش می‌یابد.

در متدولوژی Hodge-Mock، مکانیزم‌های مشخصی برای تعامل و هماهنگی بین افراد در پروژه‌های نرم‌افزاری در فرآیند متدولوژی تعریف نشده است. این موضوع باعث می‌شود استفاده از این متدولوژی در پروژه‌هایی با اندازه‌های مختلف با محدودیت مواجه شود و نتوان آن را به‌راحتی در پروژه‌های کوچک، متوسط و بزرگ به‌کار گرفت. از سوی دیگر، Hodge-Mock برای پروژه‌ها و سیستم‌های بحرانی مکانیزم‌ها و ساختارهایی در نظر گرفته است، اما به‌کارگیری آن برای سیستم‌هایی

با سطح بحرانیت پایین از نظر هزینه و پیچیدگی به‌صرفه نیست. بنابراین، این متدولوژی در تمامی پروژه‌ها با سطوح بحرانیت مختلف، مقیاس‌پذیر محسوب نمی‌شود.

در مقابل، متدولوژی USDP از مدیریت پروژه و مدیریت ریسک مناسبی برخوردار است و مدل‌سازی در آن نقش جدی و محوری دارد. استفاده از زبان مدل‌سازی غنی UML موجب می‌شود ساختار سیستم در سطوح مختلف انتزاع قابل نمایش باشد و این موضوع به مقیاس‌پذیری متدولوژی کمک می‌کند. از منظر سطوح بحرانیت، UML تا حدی از فرمالیزم با استفاده از OCL پشتیبانی می‌کند، هرچند این فرمالیزم به قدرت زبان‌های فرمان نیست. همچنین مدل‌های ایجادشده در USDP نقش مهمی در انتقال و اشتراک‌گذاری دانش میان اعضای تیم ایفا می‌کنند که این امر به استفاده از متدولوژی در پروژه‌های بزرگ‌تر و پیچیده‌تر کمک می‌نماید.

بر اساس بررسی انجام‌شده، می‌توان گفت که USDP از نظر مقیاس‌پذیری وضعیت بهتری نسبت به Hodge-Mock دارد. در حالی که Hodge-Mock به دلیل نبود مکانیزم‌های تعامل تیمی و تمرکز بر سیستم‌های بحرانی، در پروژه‌ها با اندازه‌ها و سطوح بحرانیت مختلف محدودیت دارد، USDP با بهره‌گیری از مدیریت ریسک، مدل‌سازی غنی و مستندسازی گسترده، امکان استفاده در دامنه‌ی وسیع‌تری از پروژه‌ها را فراهم می‌کند.

قابلیت پیکربندی

قابلیت پیکربندی ناظر بر آن است که آیا می‌توان یک متدولوژی را بر اساس شرایط، اندازه و موقعیت پروژه‌ای پیش از شروع یا در طول اجرای پروژه شخصی‌سازی و تنظیم کرد. وجود چارچوب‌ها، ابزارها یا مکانیزم‌های مشخص برای اعمال این تغییرات، نقش مهمی در افزایش انعطاف‌پذیری متدولوژی دارد.

در متدولوژی Hodge-Mock، وضعیت شکل‌گیری فرآیند ثابت و غیرقابل تغییر است و متدولوژی به‌صورت یک فرآیند از پیش تعریف‌شده ارائه می‌شود. برای پیکربندی این متدولوژی، چارچوب یا ابزار مشخصی ارائه نشده است و امکان اعمال تغییرات ساخت‌یافته برای انطباق با شرایط خاص پروژه وجود ندارد. از این رو، Hodge-Mock از منظر قابلیت پیکربندی با محدودیت جدی مواجه است.

در مقابل، متدولوژی USDP به دلیل پیچیدگی بالا، معمولا نیازمند پیکربندی پیش از شروع پروژه بر اساس موقعیت پروژه‌ای است. این پیکربندی عمدتا از طریق روش‌هایی مانند هرس کردن فعالیت‌ها یا تجمیع برخی اجزای فرآیند و با پشتیبانی ابزارهای مناسب انجام می‌شود. هرچند فرآیند پیکربندی USDP کار ساده‌ای نیست، اما نسبت به متدولوژی‌هایی مانند RUP ساده‌تر شده است.

به‌طور کلی، USDP قابلیت پیکربندی اولیه را داراست و می‌توان آن را متناسب با نیازهای پروژه تنظیم نمود.

بر اساس بررسی انجام‌شده، می‌توان گفت که USDP از نظر قابلیت پیکربندی وضعیت بهتری نسبت به Hodge-Mock دارد. در حالی که Hodge-Mock فرآیندی ثابت و غیرقابل تغییر ارائه می‌دهد، USDP امکان پیکربندی اولیه و تطبیق با شرایط پروژه‌ای را، هرچند با دشواری، فراهم می‌کند.

قابلیت انعطاف‌پذیری

قابلیت انعطاف‌پذیری ناظر بر آن است که آیا می‌توان فرآیند یک متدولوژی را در جریان اجرای پروژه مورد بازنگری و اصلاح قرار داد یا خیر. این معیار میزان امکان تطبیق فرآیند با شرایط جدید، بازخوردها و تغییرات پروژه‌ای را در طول چرخه‌ی ایجاد نرم‌افزار بررسی می‌کند.

در متدولوژی Hodge-Mock، مکانیزم‌هایی برای بازنگری فرآیند متدولوژی در حین اجرای پروژه تعریف شده است. به‌طور خاص، در پایان هر فاز مجموعه‌ای از سناریوهای اعتبارسنجی در نظر گرفته شده است که نتایج فاز مورد ارزیابی قرار گرفته و بر اساس آن امکان بازگشت و اصلاح تصمیم‌ها و فعالیت‌ها وجود دارد. این سازوکار باعث می‌شود فرآیند ایجاد نرم‌افزار تا حدی انعطاف‌پذیر بوده و اصلاحات لازم در ادامه‌ی مسیر اعمال شود.

در متدولوژی USDP، اگرچه میزان انعطاف‌پذیری به اندازه‌ی متدولوژی‌های چابک نیست و جریان کاری مستقلی به‌طور خاص برای بازنگری فرآیند ارائه نشده است، اما سازوکارهای متعددی برای بازخورد و اصلاح در نظر گرفته شده است. در انتهای هر فاز، جلسات بازبینی نقاط عطف برگزار می‌شود و در پایان تکرارها، ارزیابی‌های انتهای تکرار انجام می‌گیرد. همچنین وجود حلقه‌های بازخورد، جلسات مدیریت پیکربندی، بررسی‌های فنی و ارزیابی‌های کسب‌وکار باعث می‌شود علاوه بر محصول، فرآیند نیز تا حدی مورد بازنگری قرار گیرد. این ویژگی‌ها امکان اعمال اصلاحات تدریجی در فرآیند را فراهم می‌آورد.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی Hodge-Mock و USDP از نظر قابلیت انعطاف‌پذیری وضعیت قابل قبولی دارند. Hodge-Mock با استفاده از سناریوهای اعتبارسنجی پایان فاز امکان بازنگری فرآیند را فراهم می‌کند، در حالی که USDP از طریق بازبینی‌های فازی، ارزیابی تکرارها و حلقه‌های بازخورد، انعطاف‌پذیری فرآیند را تا حد مناسبی پشتیبانی می‌نماید.

معیار یازدهم: حوزه کاربرد

معیار حوزه‌ی کاربرد ناظر بر آن است که متدولوژی‌های مورد بحث در چه نوع پروژه‌ها و در چه حوزه‌هایی قابل استفاده هستند. این معیار با توجه به عواملی مانند سطح بحرانیت سیستم، اندازه و پیچیدگی پروژه و نوع سیستم هدف مورد بررسی قرار می‌گیرد.

در متدولوژی Hodge-Mock، حوزه‌های کاربرد مطرح‌شده در مراجع عمدتا شامل سیستم‌های بحرانی است. این متدولوژی برای ایجاد سیستم‌هایی که نیازمند دقت بالا و تحلیل ساخت‌یافته هستند طراحی شده و در عین حال، برای ایجاد یک سیستم اطلاعاتی نیز از نظر مفهومی کفایت می‌کند. با این حال، تمرکز اصلی آن بر پروژه‌هایی است که ماهیت حساس و بحرانی دارند.

در مقابل، متدولوژی USDP دارای فرآیندی سنگین‌وزن است که در آن مدل‌سازی، مستندسازی و فعالیت‌های مدیریتی، از جمله مدیریت سازمانی، نقش پررنگی دارند. این متدولوژی برای پروژه‌هایی با اندازه و پیچیدگی بالا که نیازمند یک روش ساخت‌یافته هستند، مناسب است و می‌تواند انواع مختلف سیستم‌های اطلاعاتی را هدف قرار دهد. اگرچه USDP با استفاده از OCL در UML تا حدی از فرمالیزم پشتیبانی می‌کند، اما این سطح از فرمالیزم برای پروژه‌هایی با بالاترین سطوح بحرانیت کافی نیست و در چنین مواردی استفاده از روش‌های کاملا فرمال توصیه می‌شود. افزون بر این، USDP در حوزه‌ی نگهداری و مراقبت از سیستم‌ها عملکرد ضعیف‌تری دارد.

بر اساس بررسی انجام‌شده، می‌توان گفت که Hodge-Mock بیشتر برای سیستم‌های بحرانی و پروژه‌های خاص‌منظوره مناسب است، در حالی که USDP به دلیل ساختار سنگین و رویکرد جامع خود، گزینه‌ی مناسب‌تری برای پروژه‌های بزرگ و پیچیده و انواع سیستم‌های اطلاعاتی محسوب می‌شود. با این حال، هیچ‌یک از دو متدولوژی برای پروژه‌هایی با بالاترین سطح بحرانیت یا برای فعالیت‌های نگهداری و مراقبت، گزینه‌ی ایده‌آل به شمار نمی‌آیند.

ارزیابی متدولوژی‌ها از منظر زبان مدل‌سازی

معیار اول: پشتیبانی از مدل‌سازی شی‌گرای سازگار، صحیح، دقیق و بدون ابهام

مدل‌سازی شی‌گرای سازگار و صحیح یکی از ارکان اساسی در طراحی و پیاده‌سازی سیستم‌های پیچیده است. این مدل‌سازی باید قادر باشد تا جنبه‌های مختلف سیستم را به‌طور صحیح و دقیق و بدون ابهام بیان کند. این جنبه‌ها شامل مدل‌سازی ساختاری، مدل‌سازی وظیفه‌ای و مدل‌سازی رفتاری هستند.

- مدل‌سازی ساختاری: مدل‌سازی اجزای سیستم و نحوه‌ی تعامل آن‌ها با یکدیگر.
- مدل‌سازی وظیفه‌ای: بیان سرویس‌ها و عملیات‌هایی که سیستم بدون در نظر گرفتن تقدم و تاخیر به بیرون از سیستم ارائه می‌دهد.
- مدل‌سازی رفتاری: توصیف اینکه چه کارهایی با چه ترتیبی در سیستم اجرا می‌شود.

علاوه بر این، متدولوژی باید از سطوح مختلف انتزاع پشتیبانی کند و بتواند مدل‌سازی دنیای واقعی را در نظر بگیرد. همچنین باید از روش‌های صوری پشتیبانی کرده و مدل‌سازی در سطوح مختلف ارتباطی، مانند ارتباط برون‌شی و درون‌شی، را پوشش دهد.

در Hodge-Mock، مدل‌هایی که با یکدیگر اشتراک زیادی داشته باشند وجود ندارند. با این حال، نمودارهای سطح سیستم و نمودارهای سطح کلاس (که هم رفتاری و هم ساختاری هستند) به‌طور مستقیم به یکدیگر وابسته‌اند و اشتراک‌هایی دارند. این متدولوژی از مدل‌سازی در سه جنبه‌ی اصلی پشتیبانی می‌کند:

- مدل‌سازی ساختاری: شامل OD، ORD، ERD و نمودار وراثت
- مدل‌سازی رفتاری: شامل OBD، SBD و OPD
- مدل‌سازی وظیفه‌ای: شامل OID، CDS، SBS، OCR، OD و OBS

در این متدولوژی، مدل‌هایی برای توصیف سیستم مستقل از پیاده‌سازی وجود دارند و تحلیل از دنیای واقعی آغاز می‌شود. مدل‌های وابسته به پیاده‌سازی، بر اساس مدل‌های مستقل به وجود می‌آیند. این متدولوژی از روش‌های صوری پشتیبانی می‌کند و مدل‌سازی درون‌شی و برون‌شی نیز در آن در نظر گرفته شده است.

در USDP، مدل‌سازی به‌صورت سنگین و با استفاده از UML که یک زبان غنی برای مدل‌سازی است، انجام می‌شود. این زبان پشتیبانی مناسبی از مدل‌سازی در سه دیدگاه ساختاری، رفتاری و وظیفه‌ای ارائه می‌دهد و در طول فازهای مختلف، مصنوعات متعددی تولید می‌شوند، مانند نمودار کلاس، نمودار مولفه، نمودار توالی، و use-case. در این متدولوژی، مدل‌سازی از قلمرو مسئله آغاز

می‌شود و انواع مدل‌های تحلیل بدون در نظر گرفتن جزئیات پیاده‌سازی ایجاد می‌شوند. سپس این مدل‌ها به تدریج به سمت قلمرو جواب حرکت کرده و مدل‌هایی با جزئیات پیاده‌سازی ساخته می‌شوند. همچنین، این متدولوژی از سطوح مختلف انتزاع و درشت‌دانی پشتیبانی می‌کند و مصنوعاتی برای مدل کردن سیستم، زیرسیستم‌ها، مولفه‌ها، کلاس‌ها و تعاملات بین اشیا تولید می‌شود. در UML، پشتیبانی از فرمالیزم از طریق OCL وجود دارد، اگرچه به اندازه‌ی روش‌های فرمال دیگر کامل نیست. یک نقطه ضعف مهم USDP به دلیل تعصب بر UML، مشکلات آن در مدل‌سازی کسب‌وکار است که در این متدولوژی به‌طور موثر پوشش داده نشده است و می‌توانست از روش‌های دیگری مانند نمودار جریان داده استفاده شود.

در مجموع، Hodge-Mock از مدل‌سازی ساختاری، رفتاری و وظیفه‌ای پشتیبانی می‌کند، اما اشتراکات زیادی بین مدل‌ها وجود ندارد و وابستگی‌ها به‌طور کامل در نظر گرفته نشده است. به‌طور کلی، این متدولوژی از مدل‌سازی در سطوح مختلف انتزاع و از روش‌های صوری پشتیبانی می‌کند. در مقابل، USDP از UML برای مدل‌سازی در سطوح مختلف انتزاع استفاده می‌کند.

معیار دوم: ارائه راهبردها و تکنیک‌هایی برای رفع ناسازگاری‌ها و مدیریت پیچیدگی

رفع ناسازگاری‌ها و مدیریت پیچیدگی در مدل‌سازی، از ارکان اصلی موفقیت یک متدولوژی است. برای این منظور، متدولوژی باید ابزارها و تکنیک‌هایی ارائه دهد که از بروز ناسازگاری بین مدل‌ها جلوگیری کند و همچنین پیچیدگی‌ها را در سطوح مختلف مدیریت نماید.

در Hodge-Mock، برای جلوگیری از بروز ناسازگاری، روش‌ها و ابزارهایی در نظر گرفته شده‌اند. به‌طور خاص، در پایان هر فاز، اعتبارسنجی انجام می‌شود تا تطابق مدل‌ها و تحلیل‌های صورت‌گرفته با نیازمندی‌ها و اهداف سیستم بررسی گردد. همچنین این متدولوژی راهکارهایی برای مدیریت پیچیدگی در زبان مدل‌سازی دارد. از جمله این راهکارها، استفاده از مدل‌های متعدد است که کمک می‌کند پیچیدگی سیستم به سادگی تقسیم‌بندی شود و هر بخش به‌صورت مستقل مدیریت گردد.

در USDP، مدل‌سازی به‌صورت بسیار جدی و سنگین صورت می‌گیرد و تعداد زیادی مدل در سطوح مختلف ایجاد می‌شود. این حجم بالای مدل‌ها نیازمند توجه ویژه به حفظ سازگاری بین آن‌ها است. برای این منظور، استفاده از ابزارهای مناسب می‌تواند به حفظ سازگاری کمک شایانی کند. ابزارهایی مانند Enterprise Architect، امکاناتی برای بررسی سازگاری بین برخی مدل‌ها ارائه می‌دهند و این کار را با بررسی معانی تعریف‌شده در UML، از جمله بررسی ارتباطات میان کلاس‌ها، مولفه‌ها و بسته‌ها، انجام می‌دهند. همچنین این ابزارها امکان بررسی سازگاری مدل‌های رفتاری (مانند نمودارهای توالی) با مدل‌های ساختاری (مانند نمودار کلاس) را فراهم می‌کنند. با این حال، در صورت استفاده

نکردن از این ابزارها، حفظ سازگاری بین مدل‌ها در این متدولوژی بسیار دشوار است. به‌ویژه که معانی UML در طول زمان برای گسترش حوزه کاربرد آن سبک‌تر شده‌اند و جا برای ناسازگاری افزایش یافته است. در راستای مدیریت پیچیدگی، UML امکاناتی برای لایه‌بندی و جزءبندی مدل‌ها ارائه می‌دهد. این کار از طریق ایجاد نمودارهای بسته در مدل‌های تحلیل و نمودارهای مولفه در مدل‌های طراحی انجام می‌شود. به این ترتیب، سیستم به لایه‌های متعدد و بخش‌های مجزا تقسیم می‌شود و در نهایت به کلاس‌ها می‌رسد، که از این طریق پیچیدگی مدل‌ها مدیریت می‌شود.

با توجه به بررسی‌های انجام‌شده، می‌توان نتیجه گرفت که هر دو متدولوژی Hodge-Mock و USDP از تکنیک‌هایی برای رفع ناسازگاری‌ها و مدیریت پیچیدگی استفاده می‌کنند، اما در مقایسه، عملکرد آن‌ها متفاوت است. در حالی که Hodge-Mock با استفاده از اعتبارسنجی پایان فاز و مدل‌های متعدد، سعی در جلوگیری از ناسازگاری‌ها و مدیریت پیچیدگی دارد، USDP به دلیل مدل‌سازی سنگین و استفاده از ابزارهای خاص برای بررسی سازگاری بین مدل‌ها، از روش‌های پیشرفته‌تری برای حفظ انسجام و مدیریت پیچیدگی بهره می‌برد. با این حال، استفاده نکردن از ابزارها در USDP می‌تواند به چالش‌هایی در حفظ سازگاری منجر شود.

ارزیابی و مقایسه‌ی دو متدولوژی OPM و EUP

ارزیابی متدولوژی‌ها از منظر فرآیندی

معیار اول: صراحت در تعریف

متدولوژی می‌بایست به درستی تعریف شده باشد؛ یک تعریف خوب دارای ویژگی‌های زیر است: صریح و بدون ابهام است.

6. پوشا است.
7. روشن است.
8. منطقی است.
9. صحیح و دارای جزئیات است.
10. سازگار و بدون تناقض است.

متدولوژی EUP در توصیف خود، تمامی موارد مربوط به چرخه‌ی حیات و واحدهای کاری را به‌صورت جامع بیان کرده است. این متدولوژی نسخه‌ای گسترش‌یافته از RUP است که با افزودن دو فاز جدید (استفاده و بازنشستگی) و دو نظام جدید و همچنین گسترش فعالیت‌های برخی نظام‌ها، دامنه‌ی پوشش خود را افزایش داده است. فازهای EUP شامل آغاز، تفصیل، ساخت، انتقال، استفاده و بازنشستگی هستند و هر فاز از چندین تکرار تشکیل می‌شود که در هر تکرار، نظام‌هایی نظیر مدل‌سازی کسب‌وکار، نیازمندی‌ها، تحلیل و طراحی، پیاده‌سازی، آزمون، استقرار، مدیریت پیکربندی و تغییر، مدیریت پروژه، محیط، عملیات و پشتیبانی و مدیریت سازمان حضور دارند. در EUP، نقش‌ها به‌طور کامل و جامع همراه با توصیف وظایف و مصنوعات تولیدی مشخص شده‌اند. علاوه بر نقش‌های RUP، نقش‌های جدیدی که عمدتا مربوط به فعالیت‌های سازمانی هستند نیز اضافه شده‌اند. مدل‌سازی در EUP بسیار جدی و غنی است و مانند RUP از UML برای ایجاد مدل‌های ساختاری، رفتاری و وظیفه‌ای استفاده می‌شود. با این تفاوت که برخلاف RUP، در EUP اجبار مطلق به استفاده از UML وجود ندارد و به‌ویژه در مدل‌سازی کسب‌وکار، استفاده از روش‌هایی مانند DFD یا BPMN مجاز و در توصیف متدولوژی نیز تصریح شده است. همچنین در EUP، توصیف دقیق مصنوعات و جزئیات آن‌ها در هر فاز و تکرار ارائه شده است. علاوه بر مصنوعات RUP، مصنوعات جدیدی عمدتا در حوزه‌های سازمانی و کسب‌وکار اضافه شده‌اند، مانند مدل معماری سازمانی، مستندات نیازمندی‌های معماری کسب‌وکار و مدل فرایندهای کسب‌وکار سازمانی. تکنیک‌ها و رویه‌های عملی برای فعالیت‌ها، به‌ویژه در حوزه‌های

مدیریتی و سازمانی، به‌صورت شفاف بیان شده‌اند و موضوعاتی نظیر مدیریت ریسک، تغییر در طرح سازمان و قواعد حاکم بر چرخه‌ی حیات و محصولات به‌طور صریح مورد توجه قرار گرفته‌اند. علاوه بر این، فعالیت‌های چتری نیز به‌صورت نظام‌های پشتیبان در تمام فازها و تکرارها تعریف شده‌اند، از جمله مدیریت پیکربندی و تغییر، مدیریت پروژه، محیط و مدیریت سازمان که خود مدیریت سازمان شامل هفت نظام مجزا است. در نهایت، EUP علاوه بر توصیف فرآیندمحور، از توصیف‌های محصول‌محور و نقش‌محور نیز برای تکمیل دیدگاه‌ها استفاده می‌کند.

متدولوژی OPM دارای سه فاز اساسی آغاز، ایجاد و استقرار است و مصنوعات تولیدشده‌ی آن به‌صورت انحصاری شامل مدل OPD به همراه توصیف OPL هستند. در این متدولوژی از افراد یا نقش‌های مشخص در ایجاد نرم‌افزار نام برده نشده و تمرکز اصلی بر زبان و مدل‌سازی خاص متدولوژی است. در فاز آغاز، هدف تعریف نیازمندی‌ها در سطح بالا، تحلیل مقدماتی، تعیین منابع و ارزیابی ریسک است و فعالیت‌هایی مانند شناسایی (توجیه اقتصادی)، انعقاد پروژه (تعیین مرز سیستم و منابع)، تعریف رسمی نیازمندی‌ها با سطحی از فرمالیسم و ارزیابی ریسک انجام می‌شود. در فاز ایجاد، تحلیل تفصیلی، طراحی تفصیلی، طراحی معماری، پیاده‌سازی و آزمون به‌صورت همزمان انجام می‌گیرند. فعالیت‌های این فاز شامل دقیق‌تر کردن نیازمندی‌ها، مدل‌سازی قلمرو مسئله با OPD، دقیق‌تر کردن معماری و پیاده‌سازی مولفه‌های سیستم است. در فاز استقرار نیز فعالیت‌هایی نظیر هضم سیستم، استفاده و نگهداشت، پایش سیستم و اعلام پایان عمر سیستم انجام می‌شود که شامل تهیه مستندات نهایی و درس‌آموخته‌ها نیز هست.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP از نظر صراحت در تعریف بسیار قوی‌تر از OPM است. EUP فازها، تکرارها، نقش‌ها، نظام‌ها، مصنوعات و تکنیک‌ها را با جزئیات کامل و از دیدگاه‌های مختلف فرآیندمحور، محصول‌محور و نقش‌محور تعریف می‌کند، در حالی که OPM اگرچه فازها و فعالیت‌ها را مشخص کرده است، اما فاقد تعریف نقش‌ها و تنوع مصنوعات بوده و توصیف آن عمدتا بر زبان و مدل OPD/OPL متمرکز است. بنابراین، از منظر صراحت در تعریف، EUP ساخت‌یافته‌تر، شفاف‌تر و جامع‌تر از OPM ارزیابی می‌شود.

معیار دوم: پوشش چرخه‌ی عمر ایجاد نرم‌افزار

چرخه‌ی عمر ایجاد نرم‌افزار در حالت عام شامل سه فاز کلی تعریف، ایجاد و مراقبت و نگهداری می‌باشد.

چرخه‌ی عمر ایجاد نرم‌افزار در حالت عام شامل سه فاز اصلی تعریف، ایجاد و مراقبت و نگهداری است. یک متدولوژی زمانی پوشش مناسبی از چرخه‌ی عمر دارد که تمامی این فازها را با فعالیت‌ها، مصنوعات و سازوکارهای مشخص در بر گیرد.

در EUP، کاوش و مدل‌سازی قلمرو مسئله به‌صورت جدی و ساخت‌یافته انجام می‌شود. در فاز آغاز، هدف اصلی شناخت چیستی مسئله برای امکان‌سنجی، استخراج نیازمندی‌های سطح بالا و متصور شدن یک معماری اولیه است. در این فاز، حدود ۲۰ درصد نیازمندی‌ها که هسته و سطح بالا هستند استخراج می‌شوند. در فاز تفصیل، مدل‌ها و تحلیل‌ها تکمیل شده و شناخت از قلمرو مسئله گسترش می‌یابد و تا حدود ۸۰ درصد نیازمندی‌ها استخراج می‌شوند و مابقی در فاز ساخت بروز می‌یابند. وجود یک نظام مستقل نیازمندی‌ها و استفاده از تکنیک‌هایی مانند use-case نشان‌دهنده‌ی پوشش کامل فاز تعریف است. در EUP، مدیریت ریسک و امکان‌سنجی نیز در فاز آغاز نقش محوری دارد. در این فاز، تیم خبره طی مدت کوتاهی با تحلیل مسئله و استفاده از نمونه‌های اولیه دورریختنی، امکان‌سنجی فنی و درک نیازمندی‌ها را ارزیابی کرده و معماری اولیه‌ای متصور می‌شود. در پایان فاز تفصیل نیز انتظار می‌رود تمامی ریسک‌های اساسی شناسایی و کنترل شده باشند. فاز ایجاد در EUP شامل فعالیت‌های جدی طراحی، پیاده‌سازی و آزمون است. معماری اولیه که در فاز آغاز ایجاد شده، در فاز تفصیل تکمیل و تثبیت می‌شود و به‌عنوان معماری مبنای قابل اجرا (EAB) مورد استفاده قرار می‌گیرد. طراحی تفصیلی، به‌ویژه برای use-caseهای پرریسک، در فاز تفصیل انجام می‌شود و در فاز ساخت ادامه می‌یابد. پیاده‌سازی نیز در قالب تکرارها، ابتدا برای نمونه‌های اولیه، سپس برای اجزای ریسک‌دار و در نهایت برای کل سیستم انجام می‌شود. آزمون در EUP به‌صورت گسترده در طول چرخه‌ی عمر صورت می‌گیرد. در فاز تفصیل و ساخت، موارد پیاده‌سازی‌شده تست می‌شوند و در فاز انتقال تست‌های آلفا، بتا، کارایی، امنیت و پذیرش انجام می‌شود. همچنین یک نظام مستقل استقرار وجود دارد که از فاز ساخت تا انتقال، مسئول نصب، انتقال به محیط کاربر و آماده‌سازی برای بهره‌برداری است. EUP علاوه بر فازهای کلاسیک، دارای فاز Production است که نقش نگهداری و پشتیبانی را ایفا می‌کند. در این فاز، عملیات و پشتیبانی، نسخه‌برداری، پایش سیستم و ارتباط با کاربران انجام می‌شود. همچنین فاز بازنشستگی برای خروج سیستم از چرخه‌ی حیات در نظر گرفته شده است. بدین ترتیب، EUP کل چرخه‌ی عمر از تعریف تا نگهداری و پایان عمر را به‌صورت جامع پوشش می‌دهد.

در OPM، فاز تعریف شامل فعالیت‌های کاوش قلمرو مسئله، مدل‌سازی آن، استخراج نیازمندی‌ها و تحلیل امکان‌پذیری است. بخشی از کاوش و تعریف نیازمندی‌ها در فاز آغازین انجام شده و بخش تکمیلی آن در فاز ایجاد و در قالب تحلیل‌های تفصیلی ادامه می‌یابد. نیازمندی‌ها در فاز آغازین با سطحی از فرمالیسم بیان می‌شوند و در پایان این فاز، امکان‌پذیری سیستم از منظر ریسک و منابع بررسی می‌گردد. فاز ایجاد در OPM شامل فعالیت‌های برنامه‌نویسی، آزمون و استقرار اولیه است. پس از

پایان این فاز، استقرار نرم‌افزار در محیط کاربر آغاز شده و فاز مستقلی برای استقرار و هضم سیستم در نظر گرفته می‌شود که در آن سیستم وارد محیط عملیاتی شده و مورد استفاده قرار می‌گیرد. در فاز مراقبت و نگهداری، فعالیت‌های مربوط به پایش سیستم و نگهداری آن تا زمان پایان عمر انجام می‌شود. این فاز عمدتا به بررسی عملکرد سیستم در محیط عملیاتی و پشتیبانی از آن اختصاص دارد.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP پوشش بسیار جامع‌تری از چرخه‌ی عمر ایجاد نرم‌افزار نسبت به OPM ارائه می‌دهد. EUP نه‌تنها فازهای تعریف، ایجاد و نگهداری را به‌صورت تفصیلی پوشش می‌دهد، بلکه با افزودن فازهای Production و بازنشستگی، کل عمر سیستم از تولد تا خروج از چرخه را در بر می‌گیرد. در مقابل، OPM اگرچه فازهای تعریف، ایجاد و مراقبت را پوشش می‌دهد، اما عمق و تنوع فعالیت‌ها و نظام‌های پشتیبان آن به گستردگی EUP نیست. بنابراین، از منظر پوشش چرخه‌ی عمر، EUP متدولوژی کامل‌تر و ساخت‌یافته‌تری نسبت به OPM محسوب می‌شود.

معیار سوم: پشتیبانی از فعالیت‌های چتری

فعالیت‌های چتری بخش مهمی از فرآیند ایجاد نرم‌افزار را در بر می‌گیرند که بر فعالیت‌هایی نظارت دارند که از تمامی فعالیت‌های چرخه‌ی حیات نرم‌افزار پشتیبانی می‌کنند. این فعالیت‌ها جزو فعالیت‌های ایجاد مستقیم محصول نبوده و عمدتا ماهیت مدیریتی دارند و تمامی فعالیت‌های چرخه‌ی حیات را می‌پوشانند. اساسی‌ترین فعالیت‌های چتری عبارت‌اند از:

1. فعالیت‌های مدیریت پروژه‌ای
2. فعالیت‌های ناظر بر مدیریت ریسک
3. فعالیت‌های ناظر بر تضمین کیفیت

مدیریت ریسک

در EUP، مدیریت ریسک نقش محوری دارد. در فاز آغاز، ریسک‌های مهم شناسایی و بررسی می‌شوند و یک سند ارزیابی ریسک تولید می‌گردد. در این فاز، از تکنیک‌هایی مانند نمونه‌ی اولیه‌ی دورریختنی و نمونه‌ی اولیه‌ی اثبات مفهوم برای امکان‌سنجی فنی و همچنین سنجش درک از نیازمندی‌ها استفاده می‌شود که نقش مهمی در کاهش عدم قطعیت‌ها و ریسک‌های اولیه دارد. در فاز تفصیل، تمرکز بر نیازمندی‌های دارای ریسک است و این نیازمندی‌ها طراحی و پیاده‌سازی می‌شوند تا ریسک‌های اصلی پروژه کاهش یابند. در انتهای این فاز انتظار می‌رود که تمام ریسک‌های اساسی شناسایی و کنترل شده باشند. علاوه بر این، اجرای فرآیند به‌صورت تکراری–افزایشی باعث می‌شود

ریسک‌ها به‌صورت تدریجی شناسایی و مدیریت شوند. همچنین برای مصنوعات کلیدی مانند توجیه اقتصادی و برنامه‌ی پروژه بازخورد دریافت می‌شود و مشارکت فعال کاربر در تصمیم‌گیری‌ها در نظر گرفته شده است که به کاهش ریسک‌های کسب‌وکار کمک می‌کند. در نظام آزمون نیز در هر تکرار، یکپارچه‌سازی انجام می‌شود که به کشف زودهنگام مشکلات و کاهش ریسک‌های فنی کمک می‌نماید.

در متدولوژی OPM، تحلیل مقدماتی و برنامه‌ریزی مبتنی بر ریسک وجود دارد و در فازهای ابتدایی امکان‌پذیری سیستم و ریسک‌ها بررسی می‌شوند. با این حال، این متدولوژی فاقد بسیاری از سازوکارهای عملی برای مدیریت مستمر ریسک است. در OPM پروتوتایپ‌سازی، رویکرد تکراری–افزایشی، یکپارچه‌سازی مستمر و پایش مستمر برنامه‌ها، فرآیندها و مصنوعات پیش‌بینی نشده است. همچنین به صحت‌سنجی و اعتبارسنجی مستمر و تلاش برای ترخیص زودهنگام اشاره‌ای نشده است که این موارد نقش مهمی در کاهش ریسک در طول پروژه دارند.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP از نظر پشتیبانی از مدیریت ریسک به‌مراتب قوی‌تر از OPM است. EUP با شناسایی سیستماتیک ریسک‌ها، استفاده از نمونه‌های اولیه، اجرای تکراری–افزایشی، یکپارچه‌سازی در هر تکرار و دریافت بازخورد مستمر، مدیریت ریسک را در سراسر چرخه‌ی حیات پشتیبانی می‌کند. در مقابل، OPM اگرچه تحلیل مقدماتی ریسک دارد، اما فاقد سازوکارهای عملی و مستمر برای کنترل و کاهش ریسک در طول پروژه است.

مدیریت پروژه

در EUP، فعالیت‌های مدیریتی پروژه در قالب یک نظام مستقل مدیریت پروژه انجام می‌شوند. این متدولوژی از ابزارها و روش‌هایی مانند نمودار گنت و PERT برای برنامه‌ریزی و زمان‌بندی فعالیت‌ها استفاده می‌کند. در انتهای فاز آغاز، برای تکرارهای ابتدایی فاز تفصیل برنامه‌ی دقیق ارائه می‌شود و برای تکرارهای بعدی و فازهای بعدی برنامه‌ریزی تخمینی و کلی انجام می‌گیرد. این برنامه‌ها به‌طور مکرر در طول فازها بازبینی و اصلاح می‌شوند که نشان‌دهنده‌ی پویایی در مدیریت پروژه است. علاوه بر این، در EUP یک نظام جدید مدیریت سازمان نیز تعریف شده است که به فعالیت‌های مدیریتی در سطح سازمان می‌پردازد. ارتباطات داخل تیم‌ها و بین تیم‌ها اهمیت بالایی دارد و از طریق جلسات روزانه، جلسات برنامه‌ریزی و جلسات بازبینی مدیریت می‌شود. همچنین از مستندسازی و مدل‌سازی مفصل برای تسهیل ارتباط و اشتراک‌گذاری دانش بین تیم‌ها استفاده می‌شود که به هماهنگی و کنترل پروژه کمک می‌کند.

در متدولوژی OPM، فعالیت مشخص و مستقلی برای مدیریت پروژه تعریف نشده است. با این حال، در انتهای فاز استقرار به موضوع پایش دائمی اشاره شده است که می‌توان آن را تا حدی ناظر بر

فعالیت‌های مانیتورینگ و کنترل پروژه دانست. اما این اشاره محدود بوده و فاقد چارچوب، ابزارها و رویه‌های مشخص برای برنامه‌ریزی، زمان‌بندی و کنترل سیستماتیک پروژه است.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP از نظر پشتیبانی از مدیریت پروژه بسیار قوی‌تر از OPM است. EUP با داشتن نظام مستقل مدیریت پروژه، ابزارهای برنامه‌ریزی و زمان‌بندی، بازبینی‌های مکرر و سازوکارهای ارتباطی و سازمانی، مدیریت پروژه را به‌صورت جامع پشتیبانی می‌کند؛ در حالی که OPM تنها اشاره‌ای محدود به پایش در انتهای فاز استقرار دارد و فاقد یک چارچوب مدیریتی صریح و ساخت‌یافته است.

تضمین کیفیت

در EUP، یک نظام مستقل آزمون تعریف شده است که مسئول اجرای انواع آزمون‌ها از جمله آزمون‌های واحد، ادغام و سیستم است. این نظام در سراسر فازهای چرخه‌ی حیات گسترش یافته است؛ به‌گونه‌ای که در فاز تفصیل، آزمون برای معماری مبنای قابل اجرا (EAB) و نیازمندی‌های ریسک‌دار انجام می‌شود، در فاز ساخت آزمون‌ها هم‌زمان با پیاده‌سازی اجزای سیستم صورت می‌گیرند و در فاز انتقال پس از استقرار، آزمون‌های اعتبارسنجی و آزمون پذیرش اجرا می‌شوند. از آن‌جا که این متدولوژی مبتنی بر نیازمندی‌ها است، امکان رهگیری آزمون‌ها به نیازمندی‌ها نیز وجود دارد که این موضوع نقش مهمی در تضمین کیفیت محصول نهایی ایفا می‌کند.

در متدولوژی OPM، تاکید اصلی بر مرور و بازبینی در پایان فاز استقرار است و تا زمان مرگ سیستم، بررسی‌هایی از منظر قابلیت نگهداشت و میزان ارضای نیازمندی‌های کاربران انجام می‌شود. با این حال، در این متدولوژی صحت‌سنجی و اعتبارسنجی مکرر در تمام فازها انجام نمی‌شود و فعالیت‌های تضمین کیفیت عمدتا به پایان فاز استقرار محدود شده‌اند.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP از نظر پشتیبانی از تضمین کیفیت بسیار قوی‌تر از OPM است. EUP با داشتن نظام مستقل آزمون، اجرای آزمون‌ها در فازهای مختلف و امکان رهگیری به نیازمندی‌ها، تضمین کیفیت را به‌صورت مستمر در طول چرخه‌ی عمر پوشش می‌دهد، در حالی که OPM تضمین کیفیت را عمدتا به مرحله‌ی پایانی استقرار محدود کرده و فاقد سازوکارهای مستمر برای صحت‌سنجی و اعتبارسنجی در طول فرآیند است.

معیار چهارم: بی‌درزی و همواری انتقال

یکی از اهداف متدولوژی‌ها، بی‌درز و هموار بودن انتقال بین فازها، فعالیت‌ها و مصنوعات است. بی‌درزی انتقال به معنای حفظ پیوستگی مفهومی و اطلاعاتی بین خروجی‌های مراحل مختلف بوده و همواری انتقال نیز به معنای جلوگیری از ایجاد گسست ناگهانی و تولید مصنوعات کاملا جدید و مستقل از مصنوعات قبلی است. وجود عدم بی‌درزی و ناهمواری در انتقال می‌تواند منجر به از بین رفتن یا تضعیف بخشی از اطلاعات و مفاهیم کلیدی در فرآیند ایجاد نرم‌افزار شود.

در EUP، اگرچه از مفاهیم شی‌گرا برای مدل‌سازی و پیاده‌سازی سیستم استفاده می‌شود و از این منظر نوعی تغییر پارادایم مشاهده می‌گردد، اما برخلاف RUP، استفاده از UML به‌صورت اجباری نیست. به‌طور خاص، در مدل‌سازی کسب‌وکار می‌توان از روش‌هایی مانند DFD نیز استفاده کرد که دید شی‌گرا ندارند و این امر می‌تواند به ایجاد درز پارادایمی منجر شود. همچنین نگاه use-case ذاتا شی‌گرا نیست و تبدیل آن به نمودارهایی مانند نمودار توالی یا فعالیت ممکن است نوعی درز ایجاد کند. با این حال، استفاده از روش‌هایی مانند نمودار فعالیت و تکنیک‌هایی نظیر خط شنا می‌تواند تا حدی این درز را پوشش دهد. از سوی دیگر، در EUP فعالیت‌ها ادامه‌ی طبیعی یکدیگر هستند و مدل‌ها به‌صورت تکراری–افزایشی تکامل پیدا می‌کنند و یا بر مبنای مدل‌های دیگر ساخته می‌شوند. این ویژگی باعث می‌شود انتقال بین فازها، فعالیت‌ها و مصنوعات همواره برقرار بوده و پیوستگی کلی فرآیند حفظ شود، هرچند در برخی نقاط، درزهای پارادایمی محدود وجود دارد.

در متدولوژی OPM، از آن‌جا که در کل فرآیند تنها از یک زبان مدل‌سازی یعنی OPD استفاده می‌شود، شکاف پارادایمی در سطح تسک‌ها وجود ندارد و این متدولوژی از این منظر تا حد خوبی بی‌درز محسوب می‌شود. با این حال، در حوزه‌ی طراحی معماری به‌دلیل عدم الزام به استفاده از زبان مشخص، امکان به‌کارگیری زبان‌های دیگر وجود دارد که می‌تواند به ایجاد درز در مدل‌سازی منجر شود. در OPM، مواردی از تولید محصولات کاملا جدید و مستقل از مصنوعات قبلی مشاهده نمی‌شود و جابه‌جایی بین فعالیت‌ها و فازها نیز هموار و تدریجی است. بنابراین، پیوستگی اطلاعاتی و مفهومی در کل فرآیند تا حد مناسبی حفظ می‌شود.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی EUP و OPM از نظر بی‌درزی و همواری انتقال وضعیت قابل قبولی دارند. EUP به دلیل انعطاف در زبان‌های مدل‌سازی و استفاده از use-case ممکن است در برخی نقاط دچار درز پارادایمی شود، اما رویکرد تکراری–افزایشی و تکامل تدریجی مدل‌ها این درزها را تا حدی جبران می‌کند. در مقابل، OPM به دلیل اتکای عمده به زبان واحد OPD، بی‌درزی بیشتری در سطح تسک‌ها دارد و انتقال بین فازها و فعالیت‌ها هموارتر انجام می‌شود، هرچند در طراحی معماری احتمال ایجاد درز وجود دارد.

معیار پنجم: مبتنی بودن بر نیازمندی‌ها

در EUP، فرآیند ایجاد نرم‌افزار به‌صورت مبتنی بر نیازمندی‌ها تعریف شده است و از روش‌هایی مانند use-case برای ثبت و مدیریت نیازمندی‌ها استفاده می‌شود. تمامی فعالیت‌های طراحی و پیاده‌سازی بر اساس این نیازمندی‌ها انجام می‌گیرند و از این طریق رهگیری مستقیم به نیازمندی‌ها در طول چرخه‌ی حیات فراهم می‌شود. در این متدولوژی، حدود ۲۰ درصد نیازمندی‌ها در فاز آغاز، تا ۸۰ درصد در فاز تفصیل و مابقی در فاز ساخت استخراج می‌شوند و سیستم بر مبنای همین نیازمندی‌ها ساخته می‌شود.

در متدولوژی OPM، نیازمندی‌ها در ابتدای فرآیند ایجاد نرم‌افزار گردآوری و ضبط می‌شوند. این نیازمندی‌ها به‌صورت مستقل و با سطحی از فرمالیسم در فاز آغازین مدل‌سازی می‌شوند و به‌عنوان یک مبنای اصلی برای مراحل بعدی ایجاد نرم‌افزار مورد استفاده قرار می‌گیرند. بدین ترتیب، نیازمندی‌ها نقش مرجع اولیه را برای ادامه‌ی فرآیند ایفا می‌کنند.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی EUP و OPM مبتنی بر نیازمندی‌ها هستند. با این حال، EUP به دلیل استفاده از use-case و استخراج تدریجی نیازمندی‌ها در طول فازها، رهگیری مستقیم و فعال‌تری به نیازمندی‌ها فراهم می‌کند، در حالی که در OPM نیازمندی‌ها عمدتا در ابتدای فرآیند استخراج و به‌عنوان مبنای مراحل بعدی استفاده می‌شوند و نقش آن‌ها در طول فازهای بعدی کمتر به‌صورت پویا دنبال می‌شود.

معیار ششم: ارزیابی محصولات متدولوژی

این معیار مشتمل است بر ۳ معیار ریزدانه که همه‌ی آن‌ها ناظر بر محصولات متدولوژی هستند؛ این سه معیار عبارتند از:

4. قابل آزمون بودن محصولات
5. ملموس بدون محصولات
6. قابلیت رهگیری محصولات به نیازمندی‌ها

قابل آزمون بودن محصولات

برای آن‌که محصولات یک متدولوژی قابل آزمون باشند، لازم است دارای ویژگی‌هایی از جمله تعداد قابل کنترل مصنوعات، پیچیدگی قابل مدیریت، وابستگی شفاف بین مصنوعات و مکمل بودن مدل‌ها باشند. افزایش بیش از حد مصنوعات، پیچیدگی زیاد یا وابستگی‌های مبهم می‌تواند فرآیند آزمون را دشوار کرده و قابلیت آزمون‌پذیری محصولات را کاهش دهد.

در EUP، مدل‌سازی به‌صورت بسیار سنگین انجام می‌شود و در طول فازهای مختلف مدل‌ها و مصنوعات متعددی تولید می‌گردند. این تکثر و پیچیدگی مدل‌ها، اگرچه به غنای تحلیل و طراحی کمک می‌کند، اما در عمل می‌تواند باعث کاهش قابلیت آزمون‌پذیری شود. به‌ویژه زمانی که برخی مدل‌ها نقش مکمل در تکمیل سایر مدل‌ها ندارند و بیشتر جنبه‌ی تزئینی پیدا می‌کنند، وابستگی‌ها مبهم شده و بررسی صحت و سازگاری مصنوعات دشوارتر می‌شود.

در متدولوژی OPM، تنها یک نوع مصنوع اصلی یعنی OPD به همراه توصیف OPL مورد استفاده قرار می‌گیرد. این تمرکز بر یک زبان و یک نوع مدل می‌تواند از نظر تعداد مصنوعات، سادگی و شفافیت ایجاد کند. با این حال، OPL زبانی بسیار پیچیده است و در بسیاری موارد خود نیازمند توضیح و مستندسازی اضافی است. این پیچیدگی زبانی می‌تواند فرآیند آزمون و بررسی صحت مدل‌ها را دشوار سازد.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی EUP و OPM از منظر قابل آزمون بودن محصولات با چالش‌هایی مواجه‌اند. در EUP، تعدد و پیچیدگی زیاد مدل‌ها آزمون‌پذیری را کاهش می‌دهد، در حالی که در OPM، هرچند تعداد مصنوعات محدود است، اما پیچیدگی بالای OPL مانع از آزمون‌پذیری آسان می‌شود. بنابراین، هیچ‌یک از دو متدولوژی در این معیار برتری قاطعی نسبت به دیگری ندارد و هر یک از جنبه‌ای با محدودیت مواجه است.

ملموس بودن محصولات

منظور از ملموس بودن محصولات، میزان قابل درک بودن و قابل ارتباط بودن مصنوعات تولیدشده برای ذی‌نفعان مختلف، به‌ویژه کاربران نهایی و مهندسان و ایجادکنندگان نرم‌افزار است. محصول ملموس محصولی است که مخاطب بتواند نقش، کاربرد و جایگاه آن را در فرآیند ایجاد نرم‌افزار به‌خوبی درک کند.

در EUP، طی فعالیت‌های مختلف، مصنوعات متعددی مانند سند چشم‌انداز، سند توجیه اقتصادی و سند ارزیابی ریسک تولید می‌شوند که ساختاری ساده دارند، برای کاربران نهایی قابل درک هستند و معمولا نیازمند تایید آن‌ها نیز می‌باشند. افزون بر این، برخی مدل‌های UML مانند نمودار

کلاس در فاز تحلیل برای کاربران قابل فهم و ملموس هستند، اما مدل‌هایی که به فاز طراحی و جزئیات پیاده‌سازی مربوط می‌شوند، برای کاربران نهایی قابل درک نیستند. از منظر ایجادکنندگان نرم‌افزار، مدل‌ها و مصنوعات EUP در صورتی که مکمل یکدیگر باشند و صرفا جنبه‌ی زینتی نداشته باشند و در راستای انجام فعالیت‌های فرآیند استفاده شوند، ملموس و قابل استفاده خواهند بود. همچنین با توجه به سنگین‌وزن بودن EUP، استفاده از آن در پروژه‌های متناسب باعث می‌شود مصنوعات آن معنادارتر و قابل درک‌تر باشند.

در متدولوژی OPM، محصولات قابل اجرا برای کاربران نهایی در جریان متدولوژی تولید نمی‌شوند. مصنوعات ایجادشده برای کاربران نهایی از نظر نحو (Syntax) و معنا (Semantic) تنها تا حد کمی با دامنه‌ی مسئله تطابق دارند، اما دارای نمای گرافیکی قابل فهم هستند که به درک کلی سیستم کمک می‌کند. از منظر مهندسان و ایجادکنندگان نرم‌افزار، مصنوعات OPM دارای کاربرد مشخص و بدون ابهام هستند و جایگاه هر یک در فرآیند ایجاد نرم‌افزار به‌طور دقیق تعیین شده است. این ویژگی باعث می‌شود مصنوعات برای تیم فنی ملموس و قابل استفاده باشند، حتی اگر برای کاربران نهایی به‌طور کامل قابل درک نباشند.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی EUP و OPM تا حد مناسبی دارای محصولات ملموس هستند، اما برای گروه‌های متفاوتی از ذی‌نفعان. EUP با تولید اسناد و مدل‌های قابل فهم برای کاربران و مصنوعات فنی برای مهندسان، دامنه‌ی وسیع‌تری از مخاطبان را پوشش می‌دهد، در حالی که OPM بیشتر بر ملموس بودن برای مهندسان تمرکز دارد و برای کاربران نهایی، مصنوعات آن محدودتر و کمتر قابل درک هستند.

قابلیت رهگیری به نیازمندی‌ها

قابلیت رهگیری نیازمندی‌ها به این معناست که کلیه‌ی محصولات و مصنوعات تولیدشده در یک متدولوژی باید به‌صورت شفاف به تک‌تک نیازمندی‌ها قابل ربط و رهگیری باشند. این ویژگی امکان بررسی تحقق نیازمندی‌ها، کنترل تغییرات و ارزیابی صحت فرآیند ایجاد نرم‌افزار را فراهم می‌سازد.

در EUP، فرآیند ایجاد نرم‌افزار به‌صورت مبتنی بر نیازمندی‌ها تعریف شده است و تمام فعالیت‌ها و مصنوعات بر اساس نیازمندی‌ها پیش می‌روند. در نتیجه، رهگیری مستقیم به نیازمندی‌ها در سراسر چرخه‌ی حیات وجود دارد و می‌توان به‌صورت شفاف ارتباط هر فعالیت، مدل و محصول را با نیازمندی متناظر آن مشخص کرد.

در متدولوژی OPM، نیازمندی‌ها مبنای هر یک از گام‌های فرآیند محسوب نمی‌شوند و این متدولوژی در بستری تکرارشونده–افزایشی و مبتنی بر نیازمندی‌ها عمل نمی‌کند. هرچند تا حدی

رویکردی برای نمایش سناریوها وجود دارد و تا حدودی می‌توان تحقق نیازمندی‌ها را در قالب زبان مدل‌سازی متدولوژی مشاهده کرد، اما رهگیری صریح و مستقیم نیازمندی‌ها در محصولات و فعالیت‌ها به‌طور کامل برقرار نیست.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP از نظر قابلیت رهگیری نیازمندی‌ها به‌مراتب قوی‌تر از OPM است. در حالی که EUP تمام فعالیت‌ها و مصنوعات را مستقیما به نیازمندی‌ها مرتبط می‌کند، OPM تنها به‌صورت محدود و غیرصریح امکان ارتباط نیازمندی‌ها با مدل‌ها و سناریوها را فراهم می‌سازد.

معیار هفتم: قابلیت جذب مشارکت فعالانه‌ی کاربر

جذب مشارکت فعالانه‌ی کاربر ناظر بر میزان حضور و نقش‌آفرینی کاربر در فرآیند برنامه‌ریزی، بازنگری و ارزیابی ایجاد نرم‌افزار است و یکی از اصول اساسی رویکردهای چابک محسوب می‌شود. این مشارکت معمولا از دو روش اصلی محقق می‌گردد:

1. حضور نماینده‌ای از مشتری در تیم ایجاد نرم‌افزار
2. برگزاری منظم جلسات برنامه‌ریزی و بازنگری با مشارکت کاربر

در EUP، مشارکت فعال کاربر تشویق شده و در ساختار متدولوژی تعبیه شده است، اما این مشارکت به‌عنوان یک اصل اساسی و دائمی در نظر گرفته نشده است. فرکانس تعامل با کاربر نسبتا کم بوده و مشارکت کاربر بیشتر به‌صورت ساختاریافته و محدود به برخی فعالیت‌ها مانند تایید برخی مصنوعات انجام می‌شود. بنابراین، اگرچه EUP امکان مشارکت کاربر را فراهم می‌کند، اما این مشارکت مستمر و پررنگ نیست.

در متدولوژی OPM، نماینده‌ای از مشتری در تیم ایجاد نرم‌افزار حضور ندارد و همچنین جلسات منظم برنامه‌ریزی و بازنگری با مشارکت کاربر در جریان فرآیند متدولوژی تعریف نشده است. در نتیجه، نقش کاربر در فرآیند ایجاد نرم‌افزار بسیار محدود بوده و مشارکت فعالانه‌ی او به‌صورت ساخت‌یافته پشتیبانی نمی‌شود.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP از نظر قابلیت جذب مشارکت فعالانه‌ی کاربر وضعیت بهتری نسبت به OPM دارد، زیرا سازوکارهایی برای دریافت تایید و بازخورد کاربر در آن پیش‌بینی شده است. با این حال، در مقایسه با رویکردهای چابک، میزان مشارکت کاربر در EUP نیز محدود و کم‌فرکانس است، در حالی که OPM تقریبا فاقد مکانیزم‌های مشخص برای مشارکت فعالانه‌ی کاربر است.

معیار هشتم: قابلیت اجرا و قابلیت اجرا به‌صورت کارا

قابلیت اجرا

قابلیت اجرا ناظر بر میزان امکان‌پذیری به‌کارگیری یک متدولوژی در پروژه‌های واقعی و عملی است و عواملی مانند پیچیدگی فرآیند، تعداد مصنوعات و میزان انطباق‌پذیری با پروژه‌های مختلف را در بر می‌گیرد.

متدولوژی EUP یک متدولوژی سنگین‌وزن است و دارای فرآیندی پیچیده می‌باشد که در طی آن مدل‌ها و مستندات مفصل و متعددی تولید می‌شوند. این ویژگی باعث می‌شود اجرای EUP به همان شکلی که تعریف شده است، در عمل دشوار باشد و از این منظر عملکرد آن در معیار قابلیت اجرا کاهش یابد. به همین دلیل، برای استفاده‌ی موثر از EUP معمولا نیاز به شخصی‌سازی متناسب با پروژه وجود دارد که خود این فرآیند نیز کار آسانی نیست.

در مقابل، متدولوژی OPM دارای فرآیندی نسبتا ساده‌تر است و از این منظر اجرای آن در پروژه‌های واقعی آسان‌تر می‌باشد. این متدولوژی به دلیل پشتیبانی نسبی از فرمالیسم، توانایی پشتیبانی از پروژه‌های پیچیده را تا حدی دارد. با این حال، به دلیل ماهیت غیرتکراری فرآیند، ریسک اجرای آن در پروژه‌ها بالاتر است.

بر اساس بررسی انجام‌شده، می‌توان گفت که OPM از نظر قابلیت اجرا عملی‌تر از EUP است، زیرا فرآیند ساده‌تری دارد، هرچند ریسک بیشتری را متحمل می‌شود. در مقابل، EUP به دلیل پیچیدگی و سنگین‌وزن بودن، اجرای دشوارتری دارد و تنها در صورت شخصی‌سازی مناسب می‌تواند به‌طور موثر مورد استفاده قرار گیرد.

قابلیت اجرا به‌صورت کارا

قابلیت اجرا به‌صورت کارا به میزان بهره‌وری، تمرکز تیم ایجاد نرم‌افزار و حداقل‌سازی عوامل مزاحم در طول اجرای متدولوژی اشاره دارد.

در متدولوژی EUP، فرآیند بسیار پیچیده و سنگین است و اجرای آن به همان شکلی که تعریف شده است، ناکـارآمد خواهد بود و نیاز به شخصی‌سازی متناسب با پروژه دارد. از آن‌جا که در EUP، مدل‌سازی، مستندسازی و فعالیت‌های مدیریتی از جمله مدیریت سازمانی به‌صورت جدی وجود دارند، این متدولوژی بیشتر برای پروژه‌های بزرگ، پیچیده و دارای دغدغه‌های سازمانی مناسب است. در عین

حال، EUP از تکنیک‌هایی مانند معماری سیستم و مبتنی بودن بر نیازمندی‌ها برای حفظ تمرکز تیم توسعه استفاده می‌کند و همچنین وابستگی بیش از حدی به روش‌هایی مانند مکالمه‌ی رو در رو ندارد. با این حال، پیکربندی و شخصی‌سازی این متدولوژی برای پروژه‌ی هدف نیازمند ابزارهای خاص بوده و خود این فرآیند می‌تواند بر کارایی اجرای آن تاثیر بگذارد.

در متدولوژی OPM، واحدهای کاری پیچیدگی بالایی ندارند که این موضوع به اجرای روان‌تر فرآیند کمک می‌کند. با این حال، به دلیل چندمنظوره بودن زبان مدل‌سازی، عواملی برای پرت شدن تمرکز ایجادکنندگان وجود دارد. در این متدولوژی، اعمال تغییرات خطاخیز در جریان ایجاد انجام نمی‌شود و فرآیند نسبتا پایدار باقی می‌ماند. OPM به فناوری یا ابزار خاصی وابسته نیست که این امر به انعطاف‌پذیری اجرای آن کمک می‌کند، اما در عین حال این متدولوژی تا حدی از نبود یک استراتژی مناسب مدیریت پروژه رنج می‌برد که می‌تواند بر کارایی کلی اجرای آن تاثیر منفی بگذارد.

بر اساس بررسی انجام‌شده، می‌توان گفت که OPM از نظر سادگی واحدهای کاری و عدم وابستگی به فناوری، اجرای کاراتری نسبت به EUP دارد، هرچند ضعف در مدیریت پروژه و احتمال پرت شدن تمرکز وجود دارد. در مقابل، EUP با وجود استفاده از سازوکارهای حفظ تمرکز، به دلیل پیچیدگی و نیاز به شخصی‌سازی، در عمل اجرای کارای دشوارتری دارد و بیشتر برای پروژه‌های بزرگ و سازمان‌محور مناسب است.

معیار نهم: قابلیت مدیریت پیچیدگی

قابلیت مدیریت پیچیدگی ناظر بر آن است که واحدهای کاری یک متدولوژی تا چه حد امکان بخش‌بندی و لایه‌بندی دارند. بخش‌بندی به معنای امکان استفاده‌ی مستقل از اجزای مختلف فرآیند بوده و لایه‌بندی نیز به معنای سازمان‌دهی فعالیت‌ها و مصنوعات در سطوح مختلف انتزاع است؛ به‌گونه‌ای که بتوان ساختارهای بزرگ را به اجزای کوچک‌تر و قابل مدیریت‌تر تقسیم نمود.

در متدولوژی EUP، استفاده از هر دو تکنیک بخش‌بندی و لایه‌بندی به‌صورت گسترده دیده می‌شود. این متدولوژی چرخه‌حیات کلی را به ۶ فاز شامل آغاز، تفصیل، ساخت، انتقال، اجرا و بازنشستگی تقسیم کرده است که خود بیانگر یک لایه‌بندی سطح بالا در فرآیند است. علاوه بر این، در ذیل هر فاز تعداد تکرار وجود دارد و هر تکرار نیز به نظام‌های مختلف مانند مدل‌سازی کسب‌وکار، نیازمندی‌ها، تحلیل و طراحی، پیاده‌سازی، آزمون، استقرار، مدیریت تغییر و پیکربندی، مدیریت پروژه، محیط، عملیات و پشتیبانی و مدیریت سازمان تقسیم می‌شود. این ساختار چندسطحی باعث می‌شود فعالیت‌ها و مصنوعات به‌صورت جزءبندی‌شده و لایه‌بندی‌شده سازمان‌دهی شوند و پیچیدگی سیستم‌های بزرگ قابل مدیریت گردد.

در متدولوژی OPM نیز مکانیزم‌هایی برای بخش‌بندی و لایه‌بندی وجود دارد. مدل‌های OPD می‌توانند به‌صورت سلسله‌مراتبی تعریف شوند، به‌طوری که یک OPD سطح بالا بتواند به OPDهای جزئی‌تر شکسته شود. این ویژگی هم نقش بخش‌بندی را ایفا می‌کند و هم امکان لایه‌بندی در سطوح مختلف انتزاع را فراهم می‌سازد. بنابراین، ساختارهای بزرگ سیستم می‌توانند به اجزای کوچک‌تر و قابل مدیریت‌تر تقسیم شوند.

بر اساس بررسی انجام‌شده، می‌توان گفت که هر دو متدولوژی EUP و OPM از نظر قابلیت مدیریت پیچیدگی وضعیت مناسبی دارند. EUP با ساختار چندسطحی شامل فازها، تکرارها و نظام‌ها، مدیریت پیچیدگی را به‌صورت فرآیندی و سازمان‌یافته پشتیبانی می‌کند، در حالی که OPM با استفاده از OPDهای سلسله‌مراتبی امکان بخش‌بندی و لایه‌بندی مدل‌ها را فراهم می‌سازد. با این حال، EUP به دلیل ساختار غنی‌تر و چندلایه‌تر، توان بالاتری در مدیریت پیچیدگی پروژه‌های بزرگ دارد.

معیار دهم: قابلیت‌های گسترش، مقیاس پذیری، پیکربندی و انعطاف

قابلیت گسترش

گسترش‌پذیری ناظر بر آن است که آیا می‌توان به فرآیند یک متدولوژی، قابلیت‌ها یا فعالیت‌های جدیدی اضافه کرد تا متدولوژی برای پروژه‌های خاص یا خاص‌منظوره متناسب‌سازی شود. وجود یک هسته‌ی قابل توسعه و نقاط مشخص برای افزودن قابلیت‌ها، از شاخص‌های مهم گسترش‌پذیری محسوب می‌شود.

متدولوژی EUP دارای فرآیندی بسیار بزرگ و پیچیده است و برای استفاده‌ی کارا از آن، معمولا لازم است به روش‌هایی مانند هرس یا تجمیع قطعات فرآیند آن را متناسب با پروژه‌ی هدف ساخت. با این حال، این متدولوژی به‌صورت یک هسته‌ی قابل گسترش تعریف نشده است و نقاط گسترش و مکانیزم‌های صریحی برای افزودن قابلیت‌ها در آن پیش‌بینی نشده‌اند. بنابراین، اگرچه در عمل می‌توان با حذف یا ترکیب بخش‌هایی از فرآیند آن را تطبیق داد، اما این کار بر پایه‌ی یک چارچوب رسمی گسترش‌پذیر انجام نمی‌شود.

در متدولوژی OPM نیز فرآیند به‌صورت یک هسته‌ی قابل توسعه تعریف نشده است و مکانیزم مشخصی برای افزودن قابلیت‌ها یا فعالیت‌های جدید به فرآیند وجود ندارد. در نتیجه، امکان گسترش رسمی و ساخت‌یافته‌ی این متدولوژی برای پروژه‌های خاص محدود است.

بر اساس بررسی انجام‌شده، می‌توان گفت که نه EUP و نه OPM از نظر قابلیت گسترش وضعیت مناسبی ندارند. هر دو متدولوژی فاقد هسته‌ی قابل توسعه و نقاط گسترش تعریف‌شده

هستند و در نتیجه، تطبیق آن‌ها برای پروژه‌های خاص بیشتر از طریق تغییرات غیررسمی و مهندسی مجدد فرآیند انجام می‌شود تا از طریق یک مکانیزم گسترش‌پذیر استاندارد.

مقیاس‌پذیری

یک متدولوژی را مقیاس‌پذیر می‌دانیم اگر بتواند در پروژه‌هایی با اندازه‌های مختلف و همچنین در پروژه‌هایی با سطوح بحرانیت متفاوت به‌کار گرفته شود. یکی دیگر از عوامل موثر بر مقیاس‌پذیری، میزان و کیفیت مدل‌سازی در متدولوژی است؛ به‌گونه‌ای که هرچه مدل‌های تولیدشده بیشتر و غنی‌تر باشند، امکان به‌کارگیری متدولوژی در پروژه‌های بزرگ‌تر و پیچیده‌تر افزایش می‌یابد.

در EUP، به دلیل وجود مدل‌سازی، مستندسازی و فعالیت‌های مدیریتی سنگین، امکان استفاده از آن در پروژه‌های بزرگ و با سطوح بحرانی بالا وجود دارد. با این حال، در پروژه‌هایی با بالاترین سطح بحرانیت، نیاز به روش‌های کاملا فرمال وجود دارد و پشتیبانی محدود UML از فرمالیسم از طریق OCL برای این سطح از بحرانیت کافی نیست و بهتر است از روش‌های فرمال تخصصی استفاده شود. از سوی دیگر، در پروژه‌های کوچک و کم‌پیچیدگی، استفاده از EUP به‌صورت کامل مقرون‌به‌صرفه نیست و نیاز به پیکربندی و شخصی‌سازی از طریق هرس یا تجمیع قطعات فرآیند دارد تا تنها بخش‌های مورد نیاز استفاده شوند؛ فرایندی که خود بسیار دشوار است.

در متدولوژی OPM، مکانیزم‌هایی برای تعامل بین افراد در پروژه‌های نرم‌افزاری در فرآیند متدولوژی وجود ندارد که این موضوع استفاده از آن را در پروژه‌های بزرگ محدود می‌کند. با این حال، این متدولوژی مکانیزم‌های خاصی برای پروژه‌های بحرانی در نظر گرفته است و به‌گونه‌ای طراحی شده که به‌کارگیری آن برای سیستم‌های با بحرانیت کم نیز به‌صرفه است. بنابراین، OPM از نظر سطوح بحرانیت، در دامنه‌ی وسیعی از پروژه‌ها قابل استفاده است.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP از نظر اندازه‌ی پروژه و پیچیدگی، مقیاس‌پذیری بالایی دارد و برای پروژه‌های بزرگ مناسب است، اما برای پروژه‌های کوچک نیازمند پیکربندی دشوار است. در مقابل، OPM از نظر سطوح بحرانیت انعطاف بیشتری دارد، اما به دلیل نبود سازوکارهای تعامل و مدیریت تیمی، در پروژه‌های بزرگ و تیم‌محور مقیاس‌پذیری کمتری نسبت به EUP دارد.

قابلیت پیکربندی

قابلیت پیکربندی ناظر بر آن است که آیا می‌توان یک متدولوژی را بر اساس شرایط، اندازه و موقعیت پروژه‌ای پیش از شروع یا در طول اجرای پروژه شخصی‌سازی و تنظیم کرد. وجود چارچوب‌ها، ابزارها یا مکانیزم‌های مشخص برای اعمال این تغییرات، نقش مهمی در افزایش انعطاف‌پذیری متدولوژی دارد.

متدولوژی EUP دارای فرآیندی پیچیده و سنگین است و برای موفقیت پروژه لازم است در ابتدای پروژه پیکربندی شود. این پیکربندی معمولا با استفاده از ابزارهای مناسب و به روش‌هایی مانند هرس یا تجمیع قطعات فرآیند انجام می‌شود تا متدولوژی متناسب با پروژه‌ی هدف تنظیم گردد. با این حال، انجام این کار ساده نیست و خود نیازمند دانش و تجربه‌ی بالای تیم است.

در متدولوژی OPM، فرآیند به‌صورت ثابت و غیرقابل تغییر تعریف شده است و چارچوب یا ابزار مشخصی برای پیکربندی متدولوژی ارائه نشده است. در نتیجه، امکان شخصی‌سازی فرآیند بر اساس ویژگی‌های پروژه به‌صورت رسمی و ساخت‌یافته وجود ندارد.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP اگرچه پیکربندی‌پذیر است، اما این پیکربندی دشوار و پرهزینه است، در حالی که OPM عملا فاقد قابلیت پیکربندی است. بنابراین، از منظر این معیار، EUP نسبت به OPM انعطاف بیشتری دارد، هرچند این انعطاف با پیچیدگی بالایی همراه است.

قابلیت انعطاف‌پذیری

قابلیت انعطاف‌پذیری ناظر بر آن است که آیا می‌توان فرآیند یک متدولوژی را در جریان اجرای پروژه مورد بازنگری و اصلاح قرار داد یا خیر. این معیار میزان امکان تطبیق فرآیند با شرایط جدید، بازخوردها و تغییرات پروژه‌ای را در طول چرخه‌ی ایجاد نرم‌افزار بررسی می‌کند.

در متدولوژی EUP، اگرچه میزان انعطاف‌پذیری به اندازه‌ی متدولوژی‌های چابک نیست، اما از طریق نظام محیط و سازوکارهای موجود در فازها، امکان تطبیق فرآیند با نیازهای پروژه یا سازمان تا حدی وجود دارد. این ویژگی اجازه می‌دهد در صورت تغییر شرایط، فرآیند به‌صورت محدود مورد بازنگری و تنظیم قرار گیرد.

در متدولوژی OPM، هیچ مکانیزم یا سناریوی مشخصی برای بازنگری فرآیند متدولوژی در حین اجرای پروژه تعریف نشده است. بنابراین، فرآیند به‌صورت ایستا و غیرقابل تغییر اجرا می‌شود و امکان تطبیق آن با شرایط جدید یا بازخوردهای پروژه‌ای وجود ندارد.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP از نظر انعطاف‌پذیری وضعیت بهتری نسبت به OPM دارد، زیرا امکان تطبیق محدود فرآیند در حین اجرا را فراهم می‌کند، در حالی که OPM فاقد هرگونه سازوکار رسمی برای بازنگری و اصلاح فرآیند در طول پروژه است.

معیار یازدهم: حوزه کاربرد

معیار حوزه‌ی کاربرد ناظر بر آن است که متدولوژی‌های مورد بحث در چه نوع پروژه‌ها و در چه حوزه‌هایی قابل استفاده هستند. این معیار با توجه به عواملی مانند سطح بحرانیت سیستم، اندازه و پیچیدگی پروژه و نوع سیستم هدف مورد بررسی قرار می‌گیرد.

متدولوژی EUP به دلیل فرآیند سنگین خود که در آن مدل‌سازی، مستندسازی و فعالیت‌های مدیریتی از جمله مدیریت سازمانی نقش جدی دارند، برای پروژه‌های بزرگ و پیچیده به‌ویژه پروژه‌هایی با دغدغه‌های سازمانی مناسب است و برای چنین پروژه‌هایی که به یک روش ساخت‌یافته نیاز دارند، انتخاب مطلوبی محسوب می‌شود. این متدولوژی برای انواع سیستم‌های اطلاعاتی نیز قابل استفاده است. با این حال، هرچند در UML از طریق OCL تا حدی پشتیبانی از فرمالیسم وجود دارد، این میزان برای پروژه‌هایی با بالاترین سطوح بحرانیت کافی نیست و در چنین مواردی استفاده از روش‌های کاملا فرمال مناسب‌تر خواهد بود.

در متدولوژی OPM، در مرجع ارائه‌شده زمینه‌ی کاربرد خاصی به‌طور صریح مطرح نشده است. با این حال، این متدولوژی برای ایجاد یک سیستم اطلاعاتی کفایت می‌کند و می‌تواند حداقل انتظارات در این نوع پروژه‌ها را برآورده سازد.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP دامنه‌ی کاربرد گسترده‌تر و مناسب‌تری برای پروژه‌های بزرگ، پیچیده و سازمان‌محور دارد، در حالی که OPM بیشتر برای پروژه‌های ساده‌تر و سیستم‌های اطلاعاتی با سطح انتظارات حداقلی مناسب است و حوزه‌ی کاربرد آن محدودتر و کم‌تعریف‌تر است.

ارزیابی متدولوژی‌ها از منظر زبان مدل‌سازی

معیار اول: پشتیبانی از مدل‌سازی شی‌گرای سازگار، صحیح، دقیق و بدون ابهام

مدل‌سازی شی‌گرای سازگار و صحیح یکی از ارکان اساسی در طراحی و پیاده‌سازی سیستم‌های پیچیده است. این مدل‌سازی باید قادر باشد تا جنبه‌های مختلف سیستم را به‌طور صحیح و دقیق و بدون ابهام بیان کند. این جنبه‌ها شامل مدل‌سازی ساختاری، مدل‌سازی وظیفه‌ای و مدل‌سازی رفتاری هستند.

- مدل‌سازی ساختاری: مدل‌سازی اجزای سیستم و نحوه‌ی تعامل آن‌ها با یکدیگر.
- مدل‌سازی وظیفه‌ای: بیان سرویس‌ها و عملیات‌هایی که سیستم بدون در نظر گرفتن تقدم و تاخیر به بیرون از سیستم ارائه می‌دهد.
- مدل‌سازی رفتاری: توصیف اینکه چه کارهایی با چه ترتیبی در سیستم اجرا می‌شود.

علاوه بر این، متدولوژی باید از سطوح مختلف انتزاع پشتیبانی کند و بتواند مدل‌سازی دنیای واقعی را در نظر بگیرد. همچنین باید از روش‌های صوری پشتیبانی کرده و مدل‌سازی در سطوح مختلف ارتباطی، مانند ارتباط برون‌شی و درون‌شی، را پوشش دهد.

در EUP، مدل‌سازی نقش محوری دارد و این متدولوژی ذاتا مدل‌محور و سنگین‌وزن است. برخلاف RUP، استفاده صرف از UML اجباری نیست و برای مدل‌سازی کسب‌وکار می‌توان از روش‌هایی مانند DFD یا BPMN نیز استفاده کرد. در عین حال، UML به‌عنوان یک زبان غنی، از مدل‌سازی ساختاری، رفتاری و وظیفه‌ای از طریق نمودارهایی مانند کلاس، پکیج، فعالیت، توالی و use-case پشتیبانی می‌کند. مدل‌سازی در EUP از قلمرو مسئله و کسب‌وکار آغاز شده و به‌صورت تکراری و تدریجی تا قلمرو پیاده‌سازی ادامه می‌یابد، به‌گونه‌ای که مدل‌ها در سطوح مختلف انتزاع ایجاد و تکامل پیدا می‌کنند. همچنین در EUP به‌ویژه با اضافه شدن لایه‌های سازمانی، مدل‌سازی از سطح سازمان تا سطح شیء پشتیبانی می‌شود. هرچند OCL در UML تا حدی فرمالیزم را فراهم می‌کند، اما برای پروژه‌های با بالاترین سطح بحرانیت کافی نیست.

در OPM، تمام مدل‌ها بر پایه‌ی یک زبان واحد یعنی OPD ساخته می‌شوند؛ از این‌رو مدل‌ها اشتراک و وابستگی بالایی با یکدیگر دارند که به سازگاری مفهومی کمک می‌کند. OPD از مدل‌سازی ساختاری، رفتاری و وظیفه‌ای به‌صورت یکپارچه پشتیبانی می‌کند. مدل‌های مستقل از پیاده‌سازی برای توصیف سیستم وجود دارند و مدل‌های وابسته به پیاده‌سازی بر مبنای آن‌ها و با استفاده از OPL ایجاد می‌شوند که پیاده‌سازی را تسهیل می‌کند. در فاز تحلیل، مدلی از دنیای واقعی تهیه می‌شود و سند

نیازمندی‌ها در ابتدا دریافت می‌گردد و OPD شباهت بالایی به واقعیت دامنه دارد. مدل‌سازی درون‌شی مستقیما در OPD وجود دارد و مدل‌سازی برون‌شی نیز به‌صورت سلسله‌مراتبی پشتیبانی می‌شود.

بر اساس بررسی انجام‌شده، EUP از نظر گستره‌ی مدل‌سازی، سطوح انتزاع و تنوع دیدگاه‌ها بسیار قوی است و برای پروژه‌های بزرگ و سازمان‌محور مناسب‌تر است، هرچند فرمالیزم آن برای بالاترین سطوح بحرانیت محدود است. در مقابل، OPM با اتکا به یک زبان واحد (OPD/OPL) انسجام و سازگاری بالایی در مدل‌سازی ایجاد می‌کند و سه جنبه‌ی ساختاری، رفتاری و وظیفه‌ای را به‌صورت یکپارچه پوشش می‌دهد، هرچند دامنه‌ی ابزارها و زبان‌های مدل‌سازی آن محدودتر از EUP است.

معیار دوم: ارائه راهبردها و تکنیک‌هایی برای رفع ناسازگاری‌ها و مدیریت پیچیدگی

رفع ناسازگاری‌ها و مدیریت پیچیدگی در مدل‌سازی، از ارکان اصلی موفقیت یک متدولوژی است. برای این منظور، متدولوژی باید ابزارها و تکنیک‌هایی ارائه دهد که از بروز ناسازگاری بین مدل‌ها جلوگیری کند و همچنین پیچیدگی‌ها را در سطوح مختلف مدیریت نماید.

در EUP، مدل‌سازی به‌صورت بسیار جدی و گسترده انجام می‌شود و تعداد زیادی مدل در سطوح مختلف انتزاع تولید می‌گردد. علاوه بر UML، از ابزارها و روش‌های دیگری مانند نمودار جریان داده (DFD) نیز استفاده می‌شود. این تنوع مدل‌ها، در عین افزایش قدرت بیان، مسئله‌ی سازگاری بین مدل‌ها را به چالشی مهم تبدیل می‌کند. برای مدیریت این موضوع، استفاده از ابزارهای پشتیبان مدل‌سازی مانند Enterprise Architect پیشنهاد می‌شود که امکان بررسی سازگاری بین مدل‌های مختلف را فراهم می‌آورند؛ برای مثال تطبیق بین نمودارهای کلاس، مولفه و پکیج و همچنین بررسی سازگاری بین مدل‌های رفتاری مانند نمودار توالی با مدل‌های ساختاری. بدون استفاده از چنین ابزارهایی، حفظ سازگاری در این حجم از مدل‌ها بسیار دشوار خواهد بود. از منظر مدیریت پیچیدگی، UML و همچنین DFD از مکانیزم‌هایی مانند بسته‌بندی، جزءبندی و شکستن مدل‌ها به زیرمدل‌ها پشتیبانی می‌کنند که امکان سازمان‌دهی سیستم‌های بزرگ و توسعه‌ی تدریجی آن‌ها را فراهم می‌سازد.

در OPM، به دلیل استفاده از یک زبان یکپارچه (OPD به همراه OPL)، سازوکارهایی برای جلوگیری از بروز ناسازگاری وجود دارد. توصیف‌های متنی OPL به روشن شدن معنای عناصر مدل و رفع ابهام کمک می‌کند و این امر احتمال ناسازگاری بین مدل‌ها را کاهش می‌دهد. از نظر مدیریت پیچیدگی نیز، زبان OPD از سلسله‌مراتب و تجزیه‌ی مدل‌ها به سطوح مختلف پشتیبانی می‌کند که امکان مدیریت ساختارهای بزرگ از طریق لایه‌بندی و جزءبندی را فراهم می‌آورد.

بر اساس بررسی انجام‌شده، می‌توان گفت که EUP با تکیه بر ابزارهای پیشرفته مدل‌سازی و مکانیزم‌های جزءبندی UML و DFD توان بالایی در مدیریت پیچیدگی دارد، اما حفظ سازگاری بین

مدل‌های متعدد آن بدون ابزارهای پشتیبان دشوار است. در مقابل، OPM به دلیل استفاده از یک زبان یکپارچه و وجود OPL برای رفع ابهام، به‌صورت ذاتی سازگاری بهتری بین مدل‌ها ایجاد می‌کند و با سلسله‌مراتبی کردن OPD، مدیریت پیچیدگی را ساده‌تر می‌سازد.