دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر - گروه نرم‌افزار

موضوع:

مقایسه‌ی متدولوژی‌های DAD با FDD

گردآورنده:

آرمان مظلوم‌زاده

درس:

متدولوژی‌های ایجاد نرم‌افزار

استاد:

دکتر رامان رامسین

زمستان ۱۴۰۰

فهرست

۱ بررسی فرایندی دو متدولوژی

در این بخش، به‌ازای معیارهای مختلف سنجش فرایند یک متدولوژی، موارد مطرح را در قالب تحلیل‌هایی برای هریک از متدولوژی‌های FDD و DAD ارائه می‌دهیم و سپس به مقایسه آنها با یکدیگر می‌پردازیم.

۱-۱ تعریف

در این معیار، به بررسی مستندات متدولوژی می‌پردازیم و بررسی می‌کنیم که آیا متدولوژی به خوبی در مستندات شرح داده شده است یا خیر. برای این کار، در مستندات به دنبال توضیحاتی واضح و جامع هستیم که با دقت بالا و جزئیات کافی، متدولوژی را تعریف کرده باشد. مواردی که انتظار می‌رود در مستندات پوشش داده شود شامل چرخه‌ی عمر و واحدهای کاری، تولیدکنندگان، زبان مدل‌سازی، محصولات، روش‌ها و قوانین و مسائل مربوط به فعالیت‌های چتری می‌باشد. این موارد می‌توانند به مرکزیت هر یک از سه حرف p مطرح در فرایند؛ یعنی Process-units، People و Products بیان شوند که رایج‌ترین آن به مرکزیت کارهای فرایندی است و عمدتا انتظار می‌رود این نوع توصیف به کار گرفته شده باشد. در این روش، ساختار مستندات تداعی‌کننده‌ی ساختار چرخه عمر فرایند است و در ذیل آن، دو p دیگر یعنی People و Products توضیح داده می‌شوند. علاوه بر آن، توصیف به مرکزیت نقش و محصول هم می‌توانند به عنوان تکمیل‌کننده ارائه شوند.

در ادامه بررسی می‌کنیم که هریک از متدولوژی‌های مورد بررسی، تا چه اندازه موارد مد نظر در معیار تعریف را پوشش داده اند.

چرخه‌ی عمر و واحدهای کاری

- FDD: فرایند اصلی از ۵ زیرفرایند تشکیل می‌شود که هر یک شامل تعدادی گام دقیق برای تحقق زیرفرایند هستند. بنابراین چرخه عمر با جزئیات کافی توضیح داده شده است. بنابراین FDD اینجا قوی عمل کرده است.
- DAD: به طور کلی از ۳ فاز تشکیل شده است که اهداف هر یک به‌خوبی و صراحتا شرح داده شده است. با توجه به اینکه DAD جدید از ۶ چرخه عمر مختلف تشکیل شده است، مشخص شده است که در برخی از چرخه‌ها، بعضی از این ۳ فاز اصلی ممکن است به فعالیت تبدیل شوند. همچنین در Exploratory به عنوان یک چرخه خاص، فاز تعریف نشده است و صرفا مراحل لازم برای انجام کار کاوش به صورت workflow مشخص شده است. بنابراین DAD نیز مانند FDD قوی عمل کرده است.

تولیدکنندگان

- FDD: برای هر فعالیت، صراحتا اشاره شده است که چه نقش‌هایی و در قالب چه تیم‌هایی درگیر می‌شوند. تا جایی که حتی اینکه هر کلاس را چه کسی باید تغییر دهد نیز مشخص است. بنابراین در این زمینه بسیار قوی عمل کرده است.

- DAD: توضیحات مفصلی درمورد نقش‌های اصلی و پشتیبان داده شده است و همچنین ساختار تیم‌ها در چرخه عمرها داده شده است. از جمله این توضیحات حقوق و مسئولیت‌های هر نقش است که به وضوح مشخص شده است و علاوه بر آن، مواردی از جمله اینکه چه کسی هر نقش را بر عهده می‌گیرد و چه سطح مهارتی باید داشته باشد را به طور کامل پوشش داده است. بنابراین DAD در این زمینه فوق‌العاده عمل کرده است و کامل‌تر از FDD ظاهر شده است؛ با اینکه خود FDD نیز خیلی خوب عمل کرده است.

زبان مدل‌سازی

- FDD: برای مدل‌سازی از نمودارهای کلاس و تعامل به صورت جدی و از نمودار فعالیت به صورت اختیاری استفاده می‌شود که جزوی از UML هستند و بنابراین کاملا شناخته شده هستند. همچنین، از Feature برای مدل‌سازی نیازمندی‌ها استفاده می‌شود که به کمک مثال‌ها و بیان ساختار آن، این مدل نیز به‌خوبی شرح داده شده است. بنابراین FDD در توضیح زبان مدل‌سازی خودش خوب عمل کرده است.
- DAD: مدل‌های مشخصی بیان نشده است و در اکثر موارد اختیار به Practitioner داده شده است؛ برای مثال، درمورد اینکه نیازمندی‌ها باید در چرخه‌ی Agile به صورت User Story باشند یا نه اجباری ندارد. با اینکه به نظر می‌رسد شرح فعالیت‌ها را بعضا با فرض استفاده از User Story پیش برده است. بنابراین DAD در این زمینه به نسبت ضعیف عمل کرده و FDD قوی‌تر ظاهر شده است.

محصولات

- FDD: اینکه با توجه به هدف هر فاز، چه محصولی باید تولید شود به خوبی مشخص شده است. با اینکه این مجموعه حداقلی است، ولی ابهامی درمورد محصولاتی که باید تولید شوند وجود ندارد.
- DAD: درمورد محصولاتی که باید در هر گام تولید شود، توضیحات کافی آورده شده است. این شامل مرحله تولید مستنداتی مانند Vision و Business Case در فاز ابتدایی می‌شود و تا تولید کد و مستندات مورد نیاز برای انجام پشتیبانی ادامه می‌یابد. بنابراین DAD نیز مانند FDD در این زمینه قوی ظاهر شده است.

روش‌ها و قوانین

- FDD: روش انجام کار در FDD مرتبا و در مراحل مختلف از مدل‌سازی کلاس‌ها تا پیاده‌سازی و تست ذکر شده است و حتی روش‌هایی برای ثبت علت تصمیمات به کمک Note ها در نظر گرفته شده است که قابل‌توجه است.
- DAD: روش انجام کار در DAD به دقت شرح داده شده است و تمامی ملاحظات مورد نیاز نیز آورده شده است. توصیه‌های لازم برای انتخاب چرخه عمر مورد نیاز هم از ویژگی‌های خاص DAD است که آن را در این زمینه خاص کرده است. بنابراین می‌توان گفت با وجود قوی بودن FDD، DAD در اینجا بهتر ظاهر شده است.

مسائل مربوط به فعالیت‌های چتری

- FDD: در FDD مستقیما درمورد فعالیت‌های چتری صحبتی نشده است. هرچند توضیحاتی که در مواردی مانند تکرارها و مدل‌سازی تکراری، انجام تست مداوم و continues integration آورده شده است را می‌توان به‌نوعی یک توصیف غیرمستقیم از فعالیت‌های چتری منظور کرد. بنابراین FDD در این زمینه به طور کلی عملکرد خوبی ندارد.
- DAD: در طول انجام چرخه عمر، همواره فعالیت‌های چتری از قبیل مدیریت ریسک، مدیریت پروژه و تضمین کیفیت را در نظر داشته است و نقش‌ها و حتی تیم‌هایی تعریف کرده است که در کنار milestone ها، راهنمایی‌ها و نقشه‌راه‌های سازمانی، انجام فعالیت‌های چتری را به شکل مطلوبی محقق کنند. این موارد به صورت مفصل و با جزئیات فوق‌العاده‌ای در این متدلوژی شرح داده شده اند. بنابراین DAD در این زمینه قوی عمل کرده است و بهتر از FDD ظاهر شده است.

۲-۱ پوشش چرخه‌ی ژنریک ایجاد

برای سنجش پوشش فرایند یک متدولوژی، ساختار آن فرایند با فرایند عمومی ایجاد نرم‌افزار که شامل سه بخش تعریف، ایجاد و نگهداری است، مقایسه می‌شود و انتظار می‌رود این سه بخش را به‌خوبی پوشش دهد.

برای اینکه یک متدولوژی بخش تعریف را به‌خوبی پوشش دهد، نیاز است که دامنه‌ی مساله به خوبی کاوش شود و مدل‌سازی دامنه، استخراج نیازمندی‌ها و تحلیل امکان‌سنجی انجام شود که به نوعی تحلیل مقدماتی است و ریسک‌ها را مورد بررسی قرار می‌دهد تا اطمینان لازم برای ادامه راه حاصل شود.

برای بررسی پوشش ایجاد، انتظار می‌رود فرایند شامل توضیحات لازم در خصوص طراحی در سطح معماری، طراحی تفصیلی، برنامه‌نویسی، تست و استقرار باشد.

برای حصول پوشش نگهداری نیز، انتظار می‌رود به مرحله‌ی نگهداری پرداخته شود و برنامه‌ریزی و فعالیت‌های مختص آن در نظر گرفته شود.

در ادامه پوشش چرخه ژنریک را به تفکیک این ۳ مورد، در فرایند متدولوژی‌های مورد بحث مورد ارزیابی قرار می‌دهیم.

تعریف، ایجاد و نگهداری

- FDD: به طور کلی تحلیل مقدماتی و فعالیت‌های پس از پیاده‌سازی در FDD مغفول مانده اند. این فعالیت‌ها شامل انجام امکان‌سنجی، سنجش و مدیریت ریسک، System Verification، System Validation، Deployment و Maintenance می‌شود که در FDD پوشش داده نشده اند. اما در مورد مواردی که پوشش داده است، در بخش تعریف، شناسایی دامنه به کمک Class Diagram انجام می‌شود و سپس به وسیله‌ی Feature، ثبت و در طول زمان refine می‌شود. در بخش ایجاد هم، به معماری اهمیت خاصی می‌دهد و همچنین در طراحی تفصیلی نیز بسیار با جزئیات و کامل عمل می‌کند؛ به‌گونه‌ای فاصله بین پیاده‌سازی و این مرحله بسیار کم است. پیاده‌سازی و تست هم به‌خوبی پوشش داده شده است و بر انجام unit-testing در سطوح

مختلف تاکید شده است. برای بخش نگهداری نیز همانطور که در نواقص اشاره شد، پوششی وجود ندارد. بنابراین با اینکه می‌دانیم FDD با هدف توسعه یک هسته‌ی Feature-Driven به وجود آمده است، اما ذکر این مورد هم لازم است که خود FDD به عنوان یک Core در این زمینه ضعیف عمل کرده است و پوشش آن ناقص بوده است.

- DAD: در چرخه عمر کلی DAD، با شروع از فاز Inception، فعالیت‌های شناسایی دامنه مساله، امکان‌سنجی و استخراج نیازمندی‌ها به طور کامل در این فاز پوشش داده شده اند و تمام جزئیات لازم ذکر شده اند. همچنین در این فاز یک معماری اولیه در می‌آید. در فاز Construction، در تکرارهای اول معماری اولیه بدست آمده از فاز قبلی کامل می‌شود و Proven Architecture محقق می‌شود. همچنین می‌دانیم که مرحله ایجاد در DAD به صورت Design-based است و بنابراین تحلیل تفصیلی نیز پوشش داده شده است. درمورد پیاده‌سازی و تست نیز صحبت شده است و به‌خوبی به دغدغه‌های مطرح در استقرار نیز پرداخته است؛ مانند CI/CD. همچنین در معرفی چهار چرخه عمر اصلی خود، همواره به نگهداری اشاره کرده است و حتی تغییراتی که با درخواست این فاز به نیازمندی‌ها اعمال می‌شوند به صورت جدی مورد توجه قرار گرفته است. بنابراین DAD جدید در این زمینه کامل ظاهر شده است و خیلی بهتر از FDD است.

۳-۱ پشتیبانی از فعالیت‌های چتری

همانطورکه فعالیت‌هایی برای ایجاد تکه‌های محصول نهایی تعریف می‌شوند، فعالیت‌هایی نیز وجود دارند که ناظر به این فعالیت‌ها هستند و آنها را مورد ارزیابی و کنترل قرار می‌دهند.

زمانی می‌گوییم یک متدولوژی مدیریت ریسک را پوشش می‌دهد که ارزیابی و کاهش ریسک در چرخه عمر متدولوژی دخالت داده شود. روش‌هایی که برای این کار ارائه می‌شوند عبارتند از: تحلیل مقدماتی امکان‌سنجی، برنامه‌ریزی بر اساس ریسک، فرایند تکراری-افزایشی، درگیرشدن فعالانه کاربر، انتشارهای زودهنگام، اعتبارسنجی و صحت‌سنجی مداوم، مرورهای تکراری و ادغام‌های مداوم. مادامی که یک متدولوژی از این روش‌ها استفاده کند، مدیریت ریسک را پوشش می‌دهد.

برای اینکه فعالیت‌های مدیریت پروژه در یک متدولوژی دیده شود، باید روش‌های برنامه‌ریزی، زمان‌بندی و کنترل در فرایند حضور داشته باشند. همچنین برنامه‌ها و زمان‌بندی‌ها باید به صورت تکراری مورد بازبینی و تجدیدنظر قرار گیرند. در آخر باید توجه ویژه‌ای به مدیریت تیم شود به‌طوری‌که تعاملات و همکاری‌های درون‌تیمی و بین‌تیمی بهبود داده شود.

حضور روش‌های ارزیابی و بهبود کیفیت شامل مرورهای تکنیکی تکراری، طراحی با قرارداد، اعتبارسنجی و صحت‌سنجی مداوم و روش‌های بهبود ردیابی‌پذیری نیازمندی‌ها می‌تواند به صورت کارا منجر به پوشش تضمین کیفیت در یک متدولوژی شود.

در این بخش، عملکرد هریک از متدولوژی‌های مورد بحث را در درمورد مهمترین فعالیت‌های چتری مورد بررسی قرار می‌دهیم.

مدیریت ریسک

- FDD: فرض شده است که تحلیل مقدماتی انجام گرفته است و لذا با درصد خوبی مطمئن هستیم که کار قابل انجام است. با توجه به اینکه بخش مهمی از بررسی ریسک‌ها در تحلیل مقدماتی است، بنابراین این یک ضعف قابل توجه برای FDD در این زمینه است. اما فرایند iterative-incremental است که خود کمک شایانی به مدیریت ریسک در طول فرایند ایجاد می‌کند. Continuous Integration در آن مطرح شده است و با توجه نسخه سال ۲۰۰۲، به نظر می‌رسد دخالت فعالانه کاربر نیز تلویحاً مد نظر بوده است و در جریان ایجاد، کاربر باید محصول را ببیند و تایید کند؛ پس Continuous Validation هم تلویحاً وجود دارد. درمورد Continuous Verification هم با تاکید بر ایجاد unit-test ها و اجرای مداوم آنها، تا حدی پوشش وجود دارد. سایر روش‌های موثر در مدیریت ریسک مانند تحلیل امکان‌سنجی، ایجاد prototype، برنامه‌ی ریسک محور، مرورهای تکراری و تحویل زودهنگام در FDD دیده نشده است. بنابراین و خصوصا با توجه به عدم انجام امکان‌سنجی اولیه و نبود مرورهای پیوسته، FDD در این مورد چندن قوی ظاهر نشده است.
- DAD: در شکل کلی آن، در فاز Inception یکی از موارد مهم مطرح، انجام امکان‌سنجی است. در مواردی که نیاز باشد، از prototyping می‌توان استفاده کرد؛ تاجایی که مرحله اصلی در یکی از چرخه‌های عمر، prototyping است. برنامه‌ریزی بر اساس ریسک را با اولویت دادن به کارهای پر ریسک‌تر و توجه دائمی به نقشه‌راه سازمان، امکان‌پذیر کرده است. فرایند دو مورد از چرخه‌های عمر اصلی آن به صورت iterative-incremental است که امکان بررسی ریسک در وجوه مختلف را در زمان کم فراهم می‌کند. دخالت فعال کاربر همواره و به صراحت در این متدولوژی مورد تاکید است. تحویل زودهنگام به دلیل سبک‌کردن بخش Inception نسبت به UP امکان پذیر شده است. به کمک دخالت فعال کاربر، Continuous Validation صراحتا و به‌شکل مطلوبی انجام می‌گیرد. در FDD تاکید بر اجرای مداوم تست‌ها به وضوح دیده می‌شود؛ تاجایی که یک نقش کمکی برای آن در نظر گرفته شده است تا به کمک اجرای تست‌های سیستمی، Continuous Verification را محقق کند. مرورهای مداوم نیز همواره دیده شده است و Continuous Integration نیز در چرخه‌های CD به شکل مطلوب و در سایرین به شکل قابل‌قبولی در نظر گرفته شده است.

مدیریت پروژه

- FDD: مدیریت پروژه در FDD ضعیف است. تنها مورد قابل ذکر، زمان‌بندی طول تکرارها است که ۲ هفته‌ای هست و در این مدت برنامه‌ریزی می‌شود که Feature های مشخصی پیاده‌سازی شوند و با توجه به آن، صاحبان کلاس‌ها درگیر می‌شوند. فعالیت‌های دیگر موثر در این زمینه از جمله برنامه‌ریزی و زمان‌بندی برای انجام تحویل‌ها، مرور و بازبینی تکراری برنامه‌ها و مدیریت تیم در تعاملات بین تیمی در FDD دیده نشده است. بنابراین FDD در این زمینه ضعیف عمل کرده است.

- DAD: با توجه به اینکه تمرکز DAD استفاده در سازمان‌ها است، فعالیت‌های متنوعی برای آن در نظر گرفته است. روش‌های کنترلی زیادی از سمت سازمان برای نظارت بر زمان‌بندی و برنامه‌ریزی وجود دارد، برنامه‌ها مرتبا مورد بازبینی قرار می‌گیرند و با تعریف دقیق نقش‌های مورد نیاز برای مدیریت افراد (برای نمونه Team Lead) در تیم‌ها و سطوح مختلف، امکان مدیریت تیم‌ها را نیز به شکل مطلوبی فراهم کرده است. بنابراین DAD در این زمینه بسیار کامل عمل کرده است و بهتر از FDD است.

تضمین کیفیت

- FDD: به دلیل اینکه همه فعالیت‌های مستقیما از روی Feature هایی که نمایانگر نیازمندی‌ها هستند بدست می‌آیند، ردیابی‌پذیری مستقیم نیازمندی‌ها ایجاد شده است. Continuous Validation نیز به صورت تلویحی در نظر گرفته شده است. همچنین، به کمک تست‌های حداقلی در سطوح مختلف، Continuous Verification نیز تا حدی برآورده شده است. اما همچنان جای خالی مرورهای فنی مرتب احساس می‌شود. بنابراین، به طور کلی عملکرد FDD در این زمینه قابل‌قبول است.
- DAD: دارای مرورهای فنی مداوم است و Continuous Validation به همراه Continuous Verification به شدت مورد تاکید اند. با توجه به اینکه نیازمندی‌ها در ابتدا مدل می‌شوند در قالب مواردی مانند User Story و در ادامه برنامه‌ریزی مبتنی بر آنها انجام می‌گیرد و با تجزیه آنها به وظایف، پیاده‌سازی انجام می‌گیرد، ردیابی‌پذیری به نیازمندی‌ها فراهم شده است. بنابراین در DAD بحث تضمین کیفیت به شکلی جدی در نظر گرفته شده است و در این زمینه بهتر از FDD ظاهر شده است.

۴-۱ نداشتن درز و گذار نرم

زمانی که در زنجیره‌ی مدل‌سازی و فعالیت‌های یک متدولوژی، تغییر پارادایم وجود نداشته باشد، می‌گوییم این متدولوژی بی‌درز است. به عبارت دیگر برای اینکه یک متدولوژی بی‌درز باشد، باید برای ایجادکنندگان در طول فرایند ایجاد، تمرکز کافی ایجاد کند تا فرایند ایجاد حول آن انجام گیرد. برای نمونه، اینکه محصولات یک متدولوژی همگی شی‌گرا باشند، یک عامل تجمیع تمرکز برای تیم محسوب می‌شود. به عنوان راه‌حلی برای ایجاد بی‌درزی، بیان همه‌ی محصولات و وظایف روی یک بیان مشترک مطرح می‌شود؛ مثلا اینکه همه محصولات بر اساس نیازمندی یا موردکاربرد بیان شوند. به عنوان راه‌حلی دیگر، اگر در همه‌ی سطوح از روند ایجاد از مدل مشخص استفاده شود و با گذار در فعالیت‌های فرایند، تنها سطح درشت‌دانگی آن مدل تغییر کند، منجر به ایجاد تمرکز می‌شود؛ مشابه چیزی که در متدولوژی کریستال داریم. بنابراین اینکه مجموعه‌ای مشخص از مدل‌ها درحال بهبود باشند و فرایند ایجاد حول آن انجام شود، می‌تواند عاملی مهم در ایجاد تمرکز برای تیم باشد.

چه متدولوژی بی‌درز باشد یا نباشد، معیار دیگری تحت عنوان «گذار نرم» برای متدولوژی مطرح می‌شود. اصلی‌ترین عاملی که می‌تواند گذار نرم را به خطر بیندازد، تولید یک محصول به‌کلی جدید از روی محصولات قبلی در جریان فرایند ایجاد است.

در ادامه به بررسی بی‌درزی و گذار نرم در متدولوژی‌های مورد بررسی می‌پردازیم.

بدون درز

- FDD: با توجه به اینکه همه چیز در FDD شی گرا است و بر مبنای Feature، تغییر پارادایم در آن وجود ندارد و قویا بدون درز است. بنابراین FDD در این زمینه عالی عمل کرده است.
- DAD: به کمک تثبیت معماری در تکرار های اول، تمرکز خیلی خوبی برای تیم ایجاد می‌شود. با این حال، همه محصولات با یک پارادایم بیان نمی‌شوند و کمی درز وجود خواهد داشت. برای مثال، بیان نیازمندی‌ها به صورت User Story و عملکردی است در حالی که در طول ایجاد، محصولات شی‌گرا از روی آن ایجاد می‌شوند. بنابراین در این مورد FDD بهتر عمل کرده است.

گذار نرم

- FDD: علاوه بر اینکه موتور FDD به صورت تکراری-افزایشی است، کارهایی مانند مدل‌سازی نیز با ذات تکراری و با refinement مداوم ساخته می‌شود و بنابراین تولید محصولات به صورت پیوسته و با گذارهای نرم انجام می‌گیرد. همچنین به دلیل اینکه طراحی به میزان بسیار خوبی مورد توجه است و جزئیات پیاده‌سازی تا حد زیادی در آن مشخص می‌شود، پرشی از طراحی به کد نداریم. بنابراین FDD در این زمینه بسیار خوبی عمل کرده است.
- DAD: به دلیل اینکه محصول به‌کلی جدیدی تولید نمی‌شود و محصولات به صورت پیوسته و افزایشی در حال تولید هستند، گذار نرم در مراحل مختلف آن خواهیم داشت. بنابراین به نظر می‌رسد عملکرد DAD در این زمینه مانند FDD خوب است.

۵-۱ نیازمندی‌ها به عنوان مبنا

اینکه در یک متدولوژی، نیازمندی‌ها در ابتدای فرایند جمع‌آوری شوند، با مدل‌هایی مخصوص خودشان مدل شوند و به‌عنوان معیاری برای طراحی و پیاده‌سازی مورد استفاده قرار گیرند، نشان می‌دهد که این متدولوژی مبنایش نیازمندی‌ها بوده است. بنابراین با تغییر احتمالی نیازمندی‌ها در آینده، این تغییر می‌تواند از سطح منطق به سطح فیزیک منتقل شود.

در ادامه بررسی می‌کنیم که متدولوژی‌های مورد بررسی تا چه اندازه بر مبنای نیازمندی‌ها هستند.

- FDD: در ابتدای فرایند با مدل‌سازی دامنه به کمک class diagram ها، مجموعه‌ای بسیار کلی از نیازمندی‌های کارکردی و غیر کارکردی ثبت می‌شوند. سپس بر مبنای این مدل، functional decomposition انجام می‌شود و Feature ها حاصل می‌شوند که بیشتر نمود نیازمندی‌های عملکردی هستند. با توجه به اینکه FDD به معماری کلی بسنده می‌کند، به نظر می‌رسد تضمینی برای در نظر گرفتن نیازمندی های غیرعملکردی در نظر گرفته نشده است ولی عملکردی‌ها به‌خوبی پوشش داده می‌شوند. در

ادامه نیز کلیه فعالیت‌هایی که انجام می‌شوند بر مبنای این Feature ها خواهد بود. بنابراین FDD یک متدولوژی Feature-Driven است و تا حد زیادی بر مبنای نیازمندی‌ها پیش می‌رود.

- DAD: در قالب فاز Inception، نیازمندی‌ها به دقت استخراج می‌شوند و نیازمندی‌های غیروظیفه‌ای به شکل خوبی در معماری بروز خواهند کرد. از طرفی این معماری در تکرارهای ابتدایی فاز بعدی تثبیت خواهد شد و به عنوان معیار قرار خواهد گرفت. همچنین تمام فعالیت‌های آینده مطابق این نیازمندی‌ها و معماری انجام خواهد شد و کنترل شدیدی نیز بدین منظور در نظر گرفته شده است. بنابراین DAD در این زمینه قوی ظاهر شده است و حتی شاید بتوان گفت بهتر از FDD (به دلیل پوشش بهتر نیازمندی‌های غیروظیفه‌ای).

۶-۱ آزمون‌پذیری، ملموس‌بودن و ردیابی‌پذیری نیازمندی‌ها

زمانی که در یک متدولوژی، تعداد مدل‌ها کم باشد و یا مدل‌ها ساده و قابل‌فهم باشند و وابستگی کمی به یکدیگر داشته باشند، می‌گوییم آن متدولوژی قابلیت آزمون خوبی دارد؛ به‌عبارت‌دیگر افزایش ناسازگاری بین مدل‌هاست که آزمون‌پذیری را کمرنگ می‌کند.

برای اینکه خاصیت ملموس‌بودن را در متدولوژی داشته باشیم، باید محصولات تولیدی برای یک از نقش‌ها، برای آن نقش قابل‌فهم باشد و کاربردش مشخص باشد. در غیر این صورت، ممکن است هر یک از نقش‌ها نسبت به متدولوژی و فرایند آن درک درستی پیدا نکنند که باعث می‌شود آن را جدی نگیرند.

در یک متدولوژی، هر محصول باید مستقیم یا غیرمستقیم تحقق یک نیازمندی باشد؛ در غیر این صورت ردیابی‌پذیری در آن ضعیف خواهد بود و حتما نیاز به تست بر اساس سناریوهای سنجش تحقق نیازمندی خواهد بود که کار دشواری نیز هست. حال اگر ردیابی‌پذیری مستقیم باشد، ردیابی‌پذیری در فعالیت‌ها اتفاق افتاده است و نیاز به کار اضافه‌ای نیست، ولی اگر غیر مستقیم باشد، نیاز به استفاده از ماتریس ردیابی‌پذیری وجود خواهد داشت.

در ادامه به ارزیابی متدولوژی‌های مورد بررسی با توجه به این سه معیار می‌پردازیم.

آزمون‌پذیری

- FDD: در FDD تاکید شده است که مدل‌ها باید حداقلی باشند ولی به یکدیگر وابسته هستند؛ چرا که همگی در حال تحقق Feature ها و بر اساس آنها هستند. بنابراین تغییر هر یک منجر به تغییر دیگری می‌شود. اما به طور کلی می‌توان گفت به‌دلیل تعداد مدل‌های اندک، FDD در این زمینه عملکرد خوبی دارد.
- DAD: با اینکه DAD افراد را به مدل‌سازی توصیه می‌کند، ولی در طول فرایند مدل‌های به‌هم‌وابسته چندانی نخواهیم داشت. بنابراین ناسازگاری بین مدل‌ها چندان مورد بحث نیست. پس DAD نیز مانند FDD در این زمینه عملکرد خوبی دارد.

ملموس‌بودن

- FDD: کاربرد هر محصول برای نقشی که تولیدکننده آن است به‌خوبی مشخص شده است و بنابراین تولیدکننده‌ی به خوبی متوجه اهمیت محصولات تولیدی هست. پس عملکرد FDD در این زمینه خوب است.
- DAD: علاوه بر اینکه هدف تولید هر محصول مشخص است، milestone هایی نیز مشخص شده است که در سطح کلان نشانگر اهمیت محصولات تولیدی هستند. بنابراین DAD نیز مانند FDD در این زمینه خوب عمل کرده است.

ردیابی‌پذیری نیازمندی‌ها

- FDD: این متدولوژی Feature-driven است و همه چیز بر مبنای مستقیم Feature است و بنابراین ردیابی‌پذیری نیازمندی‌ها در آن به صورت مستقیم انجام می‌شود. بنابراین در این زمینه عالی عمل کرده است.
- DAD: با اینکه requirement-based است، ولی نیازمندی‌ها به صورت مستقیم قابل ردیابی نیستند و باید از ماتریس ردیابی‌پذیری برای ردیابی آن استفاده کرد. بنابراین FDD در این زمینه بهتر عمل کرده است.

۱-۷ تشویق به درگیرکردن کاربر به صورت فعال

برای مدیریت ریسک و تضمین کیفیت در یک متدولوژی، گرفتن بازخورد مداوم از کاربر ضروری است؛ تاجایی که توصیه می‌شود در جلسات مرور برنامه‌ریزی و مرور محصول کاربر درگیر شود.

در ادامه میزان تشویق به درگیر کردن کاربر را در متدولوژی‌های مورد بررسی مورد ارزیابی قرار می‌دهیم.

- FDD: دخالت فعالانه کاربر مستقیما ذکر نشده است. ولی از آنجایی که در نسخه جدید ۲۰۰۲ آن، اشاره شده است که محصول باید مرتبا به کاربر نشان داده شود و باید تایید کاربر حاصل شود، می‌توان گفت این مورد تلویحاً لحاظ شده است. بنابراین عملکرد FDD در این زمینه قابل‌قبول است.
- DAD: یکی از نقاط قوت DAD، فراهم کردن زمینه‌ی دخالت کاربر در فرایند ایجاد به کمک تعریف Milestone ها و دخالت دادن راهنمایی‌ها و نقشه‌راه‌های سازمان به همراه Vision آن است. علاوه بر همه اینها، همواره در ۴ چرخه‌ی اصلی آن، مبحث Demo و گرفتن بازخورد مورد تاکید شدید است. بنابراین DAD در این زمینه عالی ظاهر شده است و بهتر از FDD عمل کرده است.

۱-۸ انجام‌پذیری و استفاده کارا

گاهی پیچیدگی فرایند متدولوژی اینقدر زیاد است که به‌کلی امکان اجرای آن برای برخی پروژه‌ها میسر نیست و یا اینکه منحصرا برای استفاده‌های خاصی طراحی شده است. در هر دو حالت، قابلیت انجام چنین متدولوژی‌هایی ضعیف است.

زمانی که یک متدولوژی برای یک پروژه انجام‌پذیر باشد، به سراغ بررسی معیار «استفاده کارا» می‌رویم. از جمله مواردی که امکان استفاده‌ی کارا از یک متدولوژی را افزایش می‌دهند عبارتند از: عدم پیچیدگی واحدهای کاری یک فرایند و حذف موارد زاید از آنها، ملموس‌بودن محصولات و فعالیت‌ها، تعریف فعالیت‌هایی مخصوص ایجاد تمرکز (مانند جلسات مدیریت تیم)، ردیابی‌پذیری مستقیم نیازمندی‌ها و تولید مدل‌ها بر اساس معماری سیستم. همچنین پرهیز از برخی موارد مانند

وابستگی به روش‌های خطاخیز (مانند تاکیدی که اجایل ها بر ارتباطات حضوری دارند که در پروژه‌های بزرگ خطاخیز است)، وابستگی به روش‌ها و ابزار خاص (مانند وابستگی به UML به عنوان تنها ابزار مدل‌سازی) و عدم پشتیبانی خوب از فعالیت‌های مربوط به مدیریت پروژه، سبب قوی شدن متدولوژی در این معیار خواهد شد.

در ادامه، متدولوژی‌های مورد بررسی را در انجام‌پذیری و استفاده کارا مورد ارزیابی قرار می‌دهیم.

انجام‌پذیری

- FDD: با توجه به اینکه FDD دارای جزئیات کافی در هر مرحله است و مدل‌سازی محدودی نیز دارد، یک متدولوژی حداقلی است و بنابراین انجام‌پذیری بالایی دارد.
- DAD: چرخه‌های عمر در DAD به خوبی شرح داده شده اند و مجموعه آنها باعث می‌شود طیف وسیعی از پروژه‌ها به کمک آن قابل انجام باشند و با پیچیدگی و زوائد حداقلی به سرانجام برسند. بنابراین DAD در این زمینه بهتر از FDD ظاهر شده است.

استفاده کارا

- FDD: واحدهای کاری ساده و مشخص هستند و هدف از تولید هر محصول و فعالیت مشخص و ملموس است. جلسات مرور و بازبینی و مدیریت تیم در آن تعریف نشده است و فعالیت‌های مدیریت پروژه نیز در آن ضعیف است. ردیابی‌پذیری نیازمندی‌ها به صورت مستقیم خواهد بود؛ چراکه Feature-Driven است و همه چیز مستقیما از روی Feature ساخته می‌شود. به معماری توجه ویژه‌ای می‌شود که هم تمرکز خوبی به تیم می‌دهد و هم تولید مدل‌ها بر اساس معماری را ممکن می‌کند. متکی به روش‌های خطاخیز نیست؛ به عنوان مثال برای ساخت Object Model، این اختیار داده شده است که از مدل دلخواه استفاده شود، با این شرط که شان استفاده از آن ذکر شده باشد. اما ضعف بزرگ آن عدم پشتیبانی از فعالیت‌های مدیریت پروژه است که امکان استفاده کارا از آن را در بسیاری از پروژه‌ها با چالش روبرو خواهد کرد. بنابراین FDD در این زمینه خوب ظاهر شده است ولی عالی نیست.
- DAD: با توجه به اینکه توانایی انتخاب چرخه عمر وجود دارد، احتمال استفاده کارا از این متدولوژی فوق‌العاده افزایش می‌یابد. از نظر معیارها نیز، واحدهای کاری مشخص و شناخته‌شده‌ای دارد که در متدولوژی‌های مشهوری مانند Scrum استفاده شده اند که موارد زائد از آنها حذف شده است و موارد مورد نیاز اضافه شده است تا تمرکزها بهتر ایجاد شوند و محصولات و فعالیت‌ها ملموس‌تر شوند. نیازمندی‌ها مبنای انجام مرحله ایجاد هستند که می‌توانند در قالب User Story ها مدل شوند ولی ردیابی‌پذیری مستقیم وجود ندارد. به دلیل اینکه از معماری به عنوان معیار اصلی تولید نرم‌افزار استفاده می‌شود و در ابتدای Construction نهایی می‌شود، مدل‌ها بر اساس معماری خواهند بود. همچنین دست Practitioner ها را در انتخاب مدل‌ها باز گذاشته است تا وابستگی به روش‌های خطاخیز نداشته باشیم. درمورد نیاز به ابزاری های CI/CD برای اجرای برخی چرخه‌های عمر هم می‌توان گفت به علت اینکه امروزه این ابزار به طور

گسترده در پلتفورم‌های مختلف در دسترس است، خطاخیزی ایجاد نمی‌کند. همچنین از فعالیت‌های مدیریت پروژه با تاکید بسیار زیادی پشتیبانی می‌کند و موارد لازم را برای آن فراهم می‌کند. بنابراین DAD در این زمینه فوق‌العاده ظاهر شده است و بهتر از FDD است.

۹-۱ مدیریت پیچیدگی

برای مدیریت پیچیدگی در متدولوژی‌ها، از دو روش جزءبندی و لایه‌بندی و یا از ترکیبی از این دو استفاده می‌شود. این مدیریت پیچیدگی هم برای محصولات مطرح می‌شود و هم برای فعالیت‌ها و نقش‌ها که به ویژه برای فعالیت‌ها اهمیت فوق‌العاده‌ای دارد. در جزءبندی تقسیم‌بندی در یک سطح مشخص انجام می‌شود؛ مثلا در نمودارهای کلاس بزرگ، برخی بخش‌ها را می‌توان در قالب بسته[1] درآورد و بدین ترتیب پیچیدگی نمودار را کم کرد. حالا اگر همین بسته‌ها را تبدیل به نمودارهای سطح بالاتری بکنیم، کار لایه‌بندی را به صورت پایین به بالا[2] انجام داده‌ایم که باز به مدیریت پیچیدگی کمک خواهد کرد. همانطور که گفته شد، این روش برای فعالیت‌ها نیز می‌توانند استفاده شود؛ مثلا زمانی که در یک متدولوژی، تعدادی فاز تعریف شده است که هر یک دارای تعدادی مرحله است و هر مرحله نیز خود دارای تعدادی وظیفه به عنوان ریزدانه‌ترین واحد کاری است، به‌خوبی از روش لایه‌بندی در مدیریت پیچیدگی فعالیت‌ها استفاده شده است.

در ادامه متدولوژی‌های مورد بررسی در مدیریت پیچیدگی مورد ارزیابی قرار می‌گیرند.

- FDD: مدیریت پیچیدگی نیازمندی ها به شکل مطلوبی و در سه سطح با روش لایه‌بندی انجام می‌گیرد. در مورد سایر محصولات از روش‌های مدیریت پیچیدگی UML می‌توان استفاده کرد که به اندازه کافی خوب هستند. فعالیت‌ها نیز در درشت دانگی های مختلف مطرح شده اند و بنابرین مدیریت پیچیدگی در سطح فعالیت‌ها نیز انجام شده است.. درمورد نقش‌ها نیز سعی شده است با در نظر گرفتن نقش‌های مدیر، پیچیدگی در سطح محدودی مدیریت شود که البته جای خالی نقش‌های مدیریتی سطح بالا احساس می‌شود؛ چراکه در مواردی پیچیدگی‌هایی برای نقش ها ایجاد شده است و ممکن است به صورت همزمان درگیر وظایف زیادی شوند. بنابراین، FDD به طور کلی در تمام جنبه ها مدیریت پیچیدگی را اعمال کرده است و عملکرد آن از این نظر خوب است.
- DAD: مدیریت پیچیدگی نیازمندی‌ها در چرخه‌های Lean و CD-Lean با اولویت‌بندی آنها انجام شده است. درمورد کد نیز با تعریف ساختار های CI/CD و کار با code-base، مدیریت پیچیدگی کدها انجام می‌شوند. از طرفی فعالیت‌ها نیز به‌خوبی در قالب فازها قرار گرفته اند و به کمک تکرارهای داخلی مدیریت پیچیدگی شده اند. همچنین نقش‌ها به شکل جامعی در قالب تیم‌های مختلف و با نقش‌های مدیریتی مورد نیاز شان قرار داده شده اند. بنابراین به طور کلی DAD در این زمینه خیلی خوب ظاهر شده است و اندکی بهتر از FDD عمل کرده است.

[1] Package
[2] Bottom-up

۱-۱۰ گسترش‌پذیری/ مقیاس‌پذیری/ تنظیم‌پذیری/ انعطاف‌پذیری

زمانی که در یک متدولوژی نقاطی وجود داشته باشد که بتوان از آن محل‌ها متدولوژی را گسترش داد، می‌گوییم متدولوژی قابل‌گسترش است؛ به عبارت دیگر متدولوژی باید خودش یک هسته باشد که بتوان برای هر پروژه، افزونه‌هایی را به آن اضافه کرد. نمونه‌هایی از چنین متدولوژی‌های قابل‌گسترش، Scrum و FDD هستند که خودشان قابل‌اجرا نیستند و به صورت یک هسته ارائه‌شده‌اند.

برای اینکه یک متدولوژی مقیاس‌پذیر باشد، باید بتواند پروژه‌های با اندازه‌های مختلف و با سطح حساسیت‌های متفاوت را پوشش دهد. لازم به ذکر است که اندازه‌ی پروژه یعنی حداکثر تعداد افرادی که می‌توانند در پروژه مشارکت داشته باشند.

در برخی متدولوژی‌ها، امکان اینکه در ابتدای پروژه و متناسب با شرایط پروژه بتوان متدولوژی را پیکربندی کرد، تعبیه شده است. برای نمونه، متدولوژی RUP به کمک ابزار RMC، این امکان را فراهم کرده است. همچنین متدولوژی Crystal Clear دارای یک فعالیت است که در آن باید مجموعه فعالیت‌های مورد نیاز در طول پروژه تعریف شود. این متدولوژی‌ها را تنظیم‌پذیر می‌گوییم.

متدولوژی‌ای که امکان تغییر پیکربندی و بازنگری در فرایند خود را در اواسط پروژه در خود تعبیه کرده باشد و به‌نوعی فعالیتی برای با بازنگری در فرایند در خود تعریف کرده باشد (مانند جلسات retrospective در Scrum)، انعطاف‌پذیری بالایی خواهد داشت.

در ادامه، به بررسی این ۴ فاکتور در متدولوژی‌های مورد بررسی می‌پردازیم.

گسترش‌پذیری

- FDD: با توجه به اینکه FDD به صورت یک core و با هدف قابل‌گسترش‌بودن توسعه داده شده است، آماده‌ی گسترش هست و افزونه‌های مختلفی نیز بدین منظور برای آن ارائه شده است که قابل استفاده است. بنابراین FDD در این زمینه فوق‌العاده عمل کرده است.
- DAD: با توجه به اینکه چرخه عمرهای داخلی در DAD عمدتا بر اساس متدولوژی‌های Kanban و Scrum هستند، در حال حاضر افزونه‌هایی برای آنها ارائه شده است. ولی صراحتا نقاط مشخصی برای گسترش فراهم نشده است. بنابراین عملکرد DAD در این زمینه چندان قابل‌قبول نیست و FDD بهتر عمل کرده است.

مقیاس‌پذیری

- FDD: با وجود داشتن ویژگی‌های برجسته‌ای از قبیل مدیریت پیچیدگی در نیازمندی‌ها، Continuous Integration، ردیابی‌پذیری به نیازمندی‌ها، اما به دلیل پشتیبانی ضعیف از فعالیت‌های مدیریت پروژه‌ای، نقص بزرگی برای اجرا در پروژه‌های با تعداد نفرات زیاد یا سطح بحرانیت بالا دارد. بنابراین FDD در این زمینه ضعیف عمل کرده است.

- DAD: پشتیبانی فوق‌العاده از فعالیت‌های مدیریتی از یک طرف و فراهم‌کردن یک متدولوژی مخصوص برای پروژه‌های بزرگ به اسم Program، با وجود اجایل بودن این امکان را فراهم کرده است که از این متدولوژی برای انجام پروژه‌های بزرگ استفاده شود. بنابراین DAD در این زمینه خوب عمل کرده است و عملکرد آن خیلی بهتر از FDD است.

تنظیم‌پذیری

- FDD: این متدولوژی در یکی از گام‌ها باید اجزای خود را تعیین کند و بنابراین این امکان را به Practitioner می‌دهد که آن را تنظیم کند. پس FDD در این زمینه خوب ظاهر شده است.
- DAD: علاوه بر امکان انتخاب بین چرخه عمرهای مختلف، یکی از مهمترین ارکان چرخه عمرهای موجود در این متدولوژی، پرداختن به WoW است که امکان تنظیم برای هر چرخه را نیز فراهم کرده است. بنابراین می‌توان گفت DAD در این زمینه کمی کامل‌تر از FDD عمل کرده است.

انعطاف‌پذیر

- FDD: در طول انجام اشاره‌ای به مرور و بازبینی نشده است. اما به دلیل ذات iterative-incremental، این امکان وجود دارد که بازبینی انجام شود و بازخوردها اعمال شوند. بنابراین،
- DAD: از طریق بهبود مداوم WoW، مرتبا فرایند پایش می‌شود که این کار در نتیجه برگزاری جلسات مرور و بازنگری حاصل می‌شود. بنابراین DAD صراحتا یک فعالیت را برای آن در نظر گرفته است و بهتر از FDD ظاهر شده است.

۱-۱۱ فضای کاربرد

می‌دانیم که هیچ متدولوژی‌ای وجود ندارد که بتواند تمام دامنه‌های موجود برای یک پروژه را پوشش دهد. بنابراین انتظار می‌رود هر متدولوژی فضای کاربرد خود را مشخص کند؛ بدین معنی که مشخص کند برای چه نوع پروژه‌هایی کارا است. ولی در عین حال انتظار می‌رود هر متدولوژی بتواند برای طراحی سیستم‌های اطلاعاتی مناسب باشد. بنابراین می‌توان فضای کاربرد هر متدولوژی را با این معیار حداقلی و با توجه به ادعای متدولوژی، مورد ارزیابی قرار داد.

در ادامه فضای کاربرد متدولوژی‌های مورد بررسی را مورد ارزیابی قرار می‌دهیم.

- FDD: صراحتا قید شده است که برای توسعه سیستم‌های تجاری به کار گرفته می‌شود. هرچند که با توجه به فقدان فعالیت‌های مدیریتی، استفاده از آن نیازمند اضافه‌کردن افزونه‌هایی به آن است. بنابراین عملکرد FDD در این زمینه عادی بوده است و درخشان نیست.
- DAD: چند چرخه عمر مختلف و با کاربردهای مختلف و با تمرکز بر پروژه‌های سازمانی را داراست که طیف وسیعی از پروژه‌های سازمانی را در بر می‌گیرد. ولی با توجه به اینکه در آن از متدولوژی‌های مشهوری مثل Scrum و Kanban نیز بهره گرفته شده است و فعالیت‌های مدیریتی فوق‌العاده‌ای را نیز در نظر گرفته

است، می‌توان گفت به طور کلی برای اغلب پروژه‌های موجود مناسب است؛ مگر پروژه‌های با پیچیدگی فوق‌العاده زیاد که نیازمند مقیاس‌پذیری در سطح بالا هستند. بنابراین DAD در این زمینه بهتر از FDD است و یک متدولوژی با فضای کاربرد گسترده ارائه کرده است.

۲ بررسی زبان مدل‌سازی دو متدولوژی

در این بخش، به‌ازای دو معیار مطرح برای سنجش زبان مدل‌سازی یک متدولوژی، دو متدولوژی FDD و DAD را مورد بررسی قرار می‌دهیم و سپس آنها را مورد مقایسه قرار می‌دهیم.

۲-۱ پشتیبانی از مدل‌سازی شی‌گرا به صورت سازگار، دقیق و بی‌ابهام

انتظار می‌رود در زبان مدل‌سازی یک متدولوژی، دیدهای مختلف ساختاری، وظیفه‌ای و رفتاری تا حدی پوشش داده شود. همچنین مدل‌سازی از سطح منطق و دامنه‌ی کسب‌وکاری، به سمت فیزیک و دامنه‌ی پیاده‌سازی برود. علاوه بر این، بتواند در سطوح مختلف درشت‌دانگی، مدل‌سازی انجام دهد؛ یعنی بتوانند مدل‌سازی کل سازمان، سیستم، زیرسیستم‌ها، ارتباطات بین کلاسی و ارتباطات درون کلاسی را به‌خوبی پوشش دهد. در آخر اگر برای مدل‌سازی از فرمالیزم هم به‌خوبی پشتیبانی کند، می‌توانیم بگوییم این متدولوژی، یک پشتیبانی قوی از مدل‌سازی ارائه می‌دهد.

در ادامه به بررسی این فاکتورها در زبان مدل‌سازی متدولوژی‌های مورد نظر می‌پردازیم.

- FDD: مدل‌سازی در سطوح مختلف درشت‌دانگی مشاهده می‌شود؛ شامل مدل‌سازی دامنه مساله، سیستم، inter-object و intra-object. دید ساختاری به وسیله Object Model و دید رفتاری مشخصا به وسیله sequence diagram بدست می‌آید. Feature ها نیز تا حدی گویای دید عملکردی هستند اما به طور کلی می‌توان گفت به اندازه رفتاری و ساختاری به عملکردی پرداخته نشده است. همچنین به فرمالیزم نیز پرداخته نشده است. بنابراین عملکرد FDD در این زمینه خوب است.
- DAD: مدل‌سازی در DAD مورد تاکید ست و وجوه مختلف را در بر می‌گیرد. همچنین، از سطح کسب‌وکاری در Inception، تا سطح کد ادامه می‌یابد و بنابراین در سطوح مختلف از منطق به سمت فیزیک انجام می‌شود. علاوه بر آن، امکان مدل‌سازی در سطوح مختلف درشت‌دانگی از کل سازمان تا ارتباطات بین کلاسی و درون کلاسی فراهم است. به عنوان یک ضعف مشترک بین اجایل‌ها، DAD از فرمالیزم نیز پشتیبانی نمی‌کند. بنابراین می‌توان گفت DAD نیز مانند FDD در این زمینه خوب عمل کرده است.

۲-۲ ارائه راهکارها و ترفندهایی برای مقابله با ناسازگاری مدل و مدیریت پیچیدگی مدل

همواره در تولید مدل‌ها، بحث ناسازگاری و پیچیدگی مطرح بوده است. بنابراین انتظار می‌رود یک متدولوژی، راهکارهایی برای کنترل سازگاری بین مدل‌ها و کنترل پیچیدگی ارائه دهد. برای اینکه کنترل ناسازگاری در سطح زبان مدل‌سازی پوشش داده شود، نیاز است که محدودیت‌های هر مدل به همراه وابستگی‌های آن به سایر مدل‌ها، در قالب semantic هایی در زبان مدل‌سازی تعریف شده باشد که در غیر این صورت، به‌ناچار بخش فرایندی باید فعالیت‌هایی برای کنترل

ناسازگاری‌های مدل‌ها در نظر بگیرد. همچنین برای پوشش کنترل پیچیدگی مدل‌ها، انتظار می‌رود ابزارهایی مانند بسته در UML تعریف شده باشند که جزءبندی و لایه‌بندی کمک کند و پیچیدگی نموداری مانند نمودار کلاس را کنترل بتواند مدیریت کند.

در ادامه به بررسی این راهکارها در متدولوژی‌های مورد نظر می‌پردازیم.

- FDD: با توجه به اینکه از زبان مدل‌سازی UML در FDD می‌تواند استفاده شود، بنابراین مدیریت پیچیدگی مدل‌ها پوشش داده می‌شود. برای مدیریت مدلی که نگه‌دارنده‌ی نیازمندی‌ها است نیز، مدیریت پیچیدگی در نظر گرفته شده است. ولی چون این مدل‌ها semantic بالایی ندارند، رفع ناسازگاری در آنها ساده نخواهد بود. بنابراین عملکرد FDD در این زمینه عادی بوده است.
- DAD: با توجه به اینکه امکان استفاده از UML فراهم است، روش‌های مناسبی برای مدیریت پیچیدگی مدل‌ها وجود خواهد داشت. اما به دلیل ناقص‌بودن بخش semantic در UML، مانند FDD، ضعف در semantic خواهیم داشت و بنابراین کنترل ناسازگاری ساده نیست. بنابراین عملکرد DAD نیز مانند FDD در این زمینه چندان درخشان نیست.