دانشگاه صنعتی شریف

تمرین اول درس متدولوژی‌های ایجاد نرم افزار

استاد:

دکتر رامسین

تهیه کننده:

امید یعقوبی سامانی

۴۰۱۲۰۵۳۴۸

فهرست مطالب

مقایسه متدولوژی های BON و OOSE براساس معیارها:

متدولوژی OOSE یا Object Oriented Software Engineering در سال ۱۹۹۲ توسط جیکابسن معرفی شد که نسخه ساده شده از متدولوژی Objectory جیکابسن بود که پیشتر در سال ۱۹۸۷ معرفی شده بود. چرخه عمر عمومی این متدولوژی شامل سه فاز تحلیل، ساخت و تست می باشد.

۱. ***تحلیل:*** تمرکز این فاز روی فهم چیستی سیستم و ساخت مدل مفهومی آن است. این فاز برای فاز های بعدی مخصوصا با تولید مدل usecase که مدل محوری کل فرآیند است زمینه سازی انجام می دهد. این فاز شامل دو زیر فاز غیر متوالی و تکراری است.

*آ. **تحلیل نیازمندی ها:*** هدف استخراج و مدل سازی نیازمندی های سیستم است و مدل نیازمندی ها نتیجه انجام این فعالیت می باشد. مدل نیازمندی ها خود به سه زیر مدل تقسیم می شود.

- ***مدل usecase:*** چیزی است که سیستم را محدود و نیازمندی های وظیفه ای را از دیدگاه کاربر توصیف می کند. این مدل مشخص کننده وظیفه‌مندی های رفتاری کامل سیستم با تعریف اینکه چه موجودیت هایی از بیرون با سیستم تعامل دارند و راهی که این موجودیت ها از سیستم استفاده می کنند می باشد. مدل usecase علاوه بر به تصویر کشیدن رابطه بین عامل ها و usecase هایی که با آنها در ارتباط هستند می تواند نشان دهنده رابطه بین خود usecase ها هم باشد. usecase می تواند رفتار usecase دیگر را گسترش یا از usecase دیگر برای اجرای وظیفه خود استفاده نماید.
- ***Domain Model Object:*** شامل اشیا سازگار نمایش دهنده موجودیت ها است که از قلمرو مسئله بدست می آیند و در آن روابط ارث بری، تجمیع و اتحاد نمایش داده می شود.
- ***توصیف Interface:*** شامل مشخصات تفصیلی از اینترفیس کاربر و اینترفیس ها با سیستم های دیگر است.

*ب. **تحلیل استحکام:*** هدف مدل سازی ساختار سیستم از نظر اینترفیس ها، داده ها و اشیا است. همچنین زیر سیستم هایی که سیستم اصلی را می سازند مشخص می شوند. در نتیجه انجام این فعالیت مدل تحلیل ساخته می شود. مدل تحلیل نمایش چگونگی وظیفه مندی هر usecase است که توسط همکاری میان اشیا تحقق می یابد. اشیا سه نوع می باشند:

- ***Entity***: نمایش موجودیت ها با حالت مانا هستند که معمولا از Domain Model Object بدست می آیند.
- ***Interface***: نمایش دهنده موجودیت هایی هستند که تراکنش های بین سیستم و عامل های خارج سیستم را مدیریت می کنند.
- ***Control***: نمایش دهنده وظیفه مندی هایی هستند که از طریق ارث بری متعلق به شی دیگری نیستند و معمولا به عنوان کنترل کننده و هماهنگ کننده پردازش ها در یک usecase می‌باشند.

۲. *ساخت:* تمرکز این فاز روی ساخت طرح کلی نرم افزار و تولید کد می باشد. این فاز شامل دو زیر فاز است. در این فاز مدل هایی که تا اینجا ساخته شده اند به یک پیکربندی فیزیکی از سیستم نگاشت می شوند. که در نتیجه سیستم نرم افزاری با تمرکز روی مسائل پیاده سازی، مدل سازی ساختار در حال اجرا و رفتار سیستم ساخته و کد نهایی تولید می شود.

آ. طراحی: هدف مدل سازی ساختار سیستم در زمان اجراست. همچنین رفتارهای درون اشیا و میان اشیا که لازمه تحقق نیازمندی ها هستند مدل می شود. در نتیجه انجام این فعالیت مدل طراحی تولید می شود. با اضافه کردن جزئیات پیاده سازی به مدل تحلیل، مدل پیاده سازی ساخته می شود که شامل توصیف ویژگی های محیط پیاده سازی، جزئیات طراحی کلاس های(بلوک) ضروری برای پیاده سازی و راههای تعامل و رفتار اشیا در زمان اجرا برای تحقق usecase ها است.

*ب. **پیاده سازی:*** هدف ساخت نرم افزار است. در نتیجه انجام این فعالیت مدل پیاده سازی که شامل کد می شود تولید خواهد شد. برای تولید کد از ویژگی های پکیج ها و بلوک های تعریف شده در مدل طراحی استفاده می شود.

۳. *تست:* تمرکز روی اعتبارسنجی و درستی‌سنجی (V&V) است. مدل تست در نتیجه انجام این فعالیت تولید خواهد شد. تست در سه سطح انجام می شود. ابتدا بلوک ها تست می شوند، پس از آن usecase ها و در نهایت کل سیستم تست می شود.

متدولوژی BON یا Better Object Notation در سال ۱۹۹۲ توسط نرسن در مقاله ای معرفی شد. نسخه بازبینی شده و با جزئیات بیشتر در سال ۱۹۹۵ معرفی که این بار B در آن مخفف کلمه Business بود. متدولوژی تلاش می کند مستقل از زبان باشد اما به طور عمیقی تحت تاثیر مکانیزم های Eiffel و رویکرد Design by Contract می باشد. چرخه عمر عمومی این متدولوژی شامل فاز تحلیل و طراحی می باشد که خودشان شامل ۹ تکلیف هستند. ٦ تکلیف مربوط به تحلیل و ۳ تکلیف مربوط به طراحی می باشند.

می توان ترتیب تکالیف را برای رسیدن به اهداف پروژه تغییر داد اما در نهایت باید تمام مدل های مورد نیاز ساخته شوند. هر تکلیف دارای تعدادی منابع ورودی است که توسط معیار پذیرش بازبینی و مجموعه ای از تحویل دادنی ها تولید می شوند. تحویل دادنی ها توصیف ایستا و پویا از سیستم در حال ساخت را نمایش می دهند.مدل ایستا شامل توصیف رسمی کلاس ها، گروه بندی آنها در کلاسترهایی به عنوان مشتری- سرویس دهنده و روابط توارث بین آنهاست که در نهایت نمایش دهنده ساختار سیستم هستند. مدل پویا رویدادهای سیستم را نشان می دهد، مشخص کننده اشیائی است که مسئولیت ساخت اشیا دیگر را دارند و همچنین سناریوهای اجرایی سیستم که بازنمایی انواع مورد استفاده در سیستم، همراه با نمودار های نمایش دهنده تبادل پیام میان اشیا می باشد. ۹ تکلیف متدلوژی در بخش زیر آمده است:

۱. ***تعیین مرزهای سیستم:*** هدف مشخص کردن دید اصلی از دنیایی که باید تحقق یابد و همچنین سیستمی است که باید مدل شود. با انجام این تکلیف System Chart و Scenario Chart تولید می شوند. تاییدیه مشتری و برنامه های کنترل کیفیت معیار پذیرش این تکلیف می باشند.
۲. ***لیست کردن کلاس های کاندید:*** استخراج کلاس های کاندید از قلمرو مسئله انجام می شود. با انجام این تکلیف Cluster Chart ساخته می شود. تاییدیه کاربران نهایی و کارشناسان معیار پذیرش این تکلیف است.
۳. ***انتخاب کلاس ها و گروه بندی آنها در کلاستر ها:*** با کمک لیستی از کلاس های کاندید تولید شده، مفاهیمی را انتخاب و آنها را به عنوان کلاس مدل می کند. با انجام این تکلیف Static Architecture و Class Dictionary تولید می شوند. سیستم مدیریت کلاس و کلاستر که به طور خودکار سازگاری کلاس ها را بررسی می کند معیار پذیرش این تکلیف است.
۴. ***تعریف کلاس ها:*** با داشتن مجموعه اولیه کلاس ها و گروه بندی آنها کلاس ها از نظر حالت(اطلاعاتی که فراهم می کنند)، رفتار(عملیاتی که می توانند انجام دهند) و قواعدی که باید توسط کلاس و سرویس گیرندگانش رعایت شود تعریف می شوند. کلاس ها شامل توابعی که اطلاعات را بدون تغییر حالت بر می گردانند (queries)، دستوراتی که هیچ اطلاعاتی بر نمی گردانند اما ممکن است

حالتی را تغییر دهند (commands) و محدودیت هایی که قواعد عمومی تجاری و شرط سازگاری هستند (contracts) می باشند. با انجام این تکلیف Class Chart تولید می شود. تاییدیه مشتری و کاربر نهایی معیار پذیرش این تکلیف است.

۵. ***ترسیم رفتار سیستم:*** مدل پویا سیستم توصیف می شود. یک نمودار سناریو اولیه، نوع های مهم سیستم را که قبلا ساخته شده اند را جمع می کند که ارزش زیادی برای یافتن کلاس های کاندید اولیه و انتخاب بین دیدهای دیگر قلمرو دارند. با انجام این تکلیف Event Chart، Creation Chart و Object Scenario تولید می شوند. سازگاری با مدل ایستا و تاییدیه کاربران نهایی و مشتریان معیار پذیرش این تکلیف هستند.

۶. ***تعریف ویژگی های عمومی:*** توصیف غیر رسمی کلاس ها در نمودار کلاس، به اینترفیس های رسمی کلاس ها با قراردادهای نرم افزاری ترجمه می شوند. Queries به functions ، commands به procedures و contracts تبدیل به Pre/Post Condition برای توابع و ثوابت روی کل کلاس می شوند. signature هر عملیات عمومی هم مشخص می شود که نوعا می تواند باعث تعریف روابط جدید بین کلاس ها بشود. با انجام این تکلیف Class Interface تولید می شود. سازگاری با static architecture و پوشش رویدادهای خروجی در event chart و سناریوهای انتخاب شده در Scenario chart توسط ویژگی ها معیار پذیرش این تکلیف هستند.

۷. ***پالایش سیستم:*** شروع کننده بخش طراحی فرآیند است و تکرار فعالیت هایی است که قبلا برای تحلیل کلاس ها انجام شده اند و روی طراحی جدید کلاس ها اعمال می شوند. در این فعالیت تنها مصنوعات قبلی پالایش می شوند و جزئیاتشان تکمیل می گردد. سازگاری با Object Creation Chart و Object Scenario معیار پذیرش این تکلیف هستند.

۸. ***عمومی سازی:*** بهبود وراثت سلسله مراتبی کلاسها با فاکتورگیری حالات و رفتارهای مشترک در این تکلیف انجام می شوند. در این فعالیت تنها مصنوعات قبلی پالایش می شوند. سازگاری با Object Scenario معیار پذیرش این تکلیف هستند.

۹. ***تکمیل و بازبینی سیستم:*** مدل ها جلا داده و کامل می شوند و سازگاری کل سیستم بررسی می شود. در این فعالیت تنها مصنوعات قبلی پالایش می شوند. اعتبارسنجی نحوی کلاس ها، سازگاری ارثبری و ثابت های کلاس های ترکیب شده و سازگاری پیش شرط و پس شرط روتین های چند ریختی معیار پذیرش این تکلیف هستند.

در ادامه این دو متدولوژی با کمک معیار های موجود مورد مقایسه قرار می گیرند.

معیارهای فرآیند:

معیار تعریف:

متدولوژی باید به خوبی تعریف شده باشد و تعریف آن جامع، روشن، منطقی، دقیق، با جزئیات، سازگار و باید به خوبی مستند شده باشد.

چرخه عمر و واحدهای کاری:

متدولوژی BON: در BON92 چرخه عمر متدولوژی و واحدهای کاری که باید در طول فرآیند انجام شود به صورت کلی با عنوان تحلیل و طراحی مشخص شده. در BON95 فعالیت های فرآیند در سه گروه با اهداف تقریبا مشابه دسته بندی شده‌اند.

متدولوژی OOSE: در این متدولوژی چرخه عمر متدولوژی و واحدهای کاری که باید در طول فرایند انجام شوند در سه فاز اصلی تحلیل، ساخت و تست تعریف شده اند.

نقش ها و افراد:

متدولوژی BON: در BON92 اشاره ای به این موضوع نشده اما با دقت در BON95 می توان برخی نقش ها شامل Customer، End-User، Domain Experts، Developer، Analyst و Designer را به طور ضمنی مشاهده کرد.

متدولوژی OOSE: در این متدولوژی اشاره ای به نقش ها و افراد درگیر در متدولوژی نشده است.

زبان مدل سازی:

متدولوژی BON: این متدولوژی زبان خاص مدل سازی خود را تعریف کرده است .

متدولوژی OOSE: این متدولوژی دارای زبان مدل سازی خاص خود می باشد.

محصولات:

متدولوژی BON: محصولاتی که در هر تکلیف در این متدولوژی ایجاد می شوند به خوبی مشخص شده است و در BON95 به آنها تحویل‌دادنی گفته می شود که ده مصنوع تولید می شود و محصولات به صورت بصری هم نمایش داده شده اند که به فهم متدولوژی کمک می کند.

متدولوژی OOSE: در این متدولوژی پانزده مصنوع در ۵ گروه مختلف تولید می شوند و محصولات به صورت بصری هم نمایش داده شده اند که به فهم متدولوژی کمک زیادی می کند.

تکنیک ها و قواعد:

متدولوژی BON: در BON92 تکنیک ها و قواعد تا حدی تعریف شده اند و در BON95 جزئیات بیشتری به آن اضافه شده است و چگونگی انجام فعالیت ها مشخص شده اند.

متدولوژی OOSE: در این متدولوژی تکنیک ها و قواعد به صورت شفاف تعریف شده و چگونگی انجام فعالیت ها مشخص شده اند.

فعالیت های چتری:

متدولوژی BON: در BON92 اشاره ای به فعالیت های چتری نشده و در BON95 هم از مصنوعاتی که تولید می شود به صورت ضمنی می توان فهمید تضمین کیفیت وجود دارد که در تکلیف ۱ آن و در بخش معیار های پذیرش با عنوان برنامه ریزی برای تضمین کیفیت آمده است. همچنین اضافه کردن سرآیند و ایندکس به کلاس ها را که حاوی اطلاعاتی در مورد پیاده سازی و رهگیری کلاس ها است را شاید بتوان بخشی از مدیریت پروژه نامید.

متدولوژی OOSE: در این متدولوژی اشاره ای به فعالیت های چتری نشده است.

معیار پوشش چرخه عمر عمومی ایجاد نرم‌افزار:

چرخه عمر عمومی شامل تعریف، ایجاد و نگهداری است که اگر این سه مرحله کامل پوشش داده شوند چرخه عمر کامل است.

تعریف:

کاوش قلمرو مسئله و مدل سازی آن:

متدولوژی BON: در BON92 با یافتن، نامگذاری و خوشه بندی کلاس ها و ساخت مدل Cluster Chart مرزهای سیستم مشخص می شود.Dynamic Model نشان دهنده ویژگی های رفتاری و Static Model نشان دهنده ویژگی های ساختاری سیستم است. در Event Chart با مشخص شدن رویداد های داخلی و خارجی نیز به نوعی مرزهای سیستم مشخص می شوند همچنین رفتارها را هم مشخص می کند. در BON95

مرزهای سیستم با System Chart و Scenario Chart مشخص می شوند. ویژگی های رفتاری سیستم هم با Event Chart، Creation Chart و Object Scenario مشخص می شوند.

متدولوژی OOSE: در این متدولوژی مرزهای سیستم را می توان با Use Case Model و چیستی را سیستم را با کمک Domain Model مشخص نمود. در تحلیل استحکام نیز ساختار سیستم و زیر سیستم ها بدست می آیند.

استخراج نیازمندی ها:

متدولوژی BON: در این متدولوژی صحبتی از استخراج نیازمندی ها وجود دارد و مدل ها بر مبنای نیازمندی‌های غیر رسمی کاربران ساخته می شوند.

متدولوژی OOSE: در این مدل گفته شده است که نیازمندی ها در مرحله تحلیل نیازمندی ها از روی نیازمندی های غیررسمی کاربران استخراج می شوند.

امکان سنجی:

متدولوژی BON: در این متدولوژی صحبتی درباره امکان سنجی نشده است

متدولوژی OOSE: در این متدولوژی صحبتی درباره امکان سنجی نشده است

ایجاد:

طراحی معماری:

متدولوژی BON: معماری در این متدولوژی با کمک خوشه بندی کلاس ها و به نوعی مشخص شدن زیر سیستم و دقیق تر کردن مدل ها در فازهای بعد بدست می آید.

متدولوژی OOSE: در این متدولوژی در فاز تحلیل استحکام با شناسایی زیر سیستم ها و در فاز طراحی با تعیین ویژگی های محیط پیاده سازی، تعریف بلوک ها(کلاس ها) و افزار آنها در پکیج ها تا حد خوبی معماری مشخص می شود.

طراحی تفصیلی:

متدولوژی BON: در BON92 با انجام فعالیت های توصیف، ایندکس کردن و نمونه سازی از کلاس ها که فعالیت های مرتبط با قلمرو راه حل هستند وارد فاز تحلیل تفصیلی می شود و تا انتها ادامه پیدا می کند. در

BON95 با اضافه کردن جزئیات به مدل ها و یافتن کلاس های جدید و ویژگی های جدید، عمومی سازی که به معنی فاکتور گیری از ویژگی های مشترک کلاس‌ها است و کامل کردن و بازبینی سیستم طراحی تفصیلی انجام می پذیرد.

متدولوژی OOSE: در این متدولوژی در زیرفاز طراحی در فاز ساخت با اضافه کردن جزئیات به مدل ها با در نظر گرفتن ویژگی های پیاده سازی مدل های طراحی ساخته می شوند که توصیف کننده ویژگی های محیط پیاده سازی و جزئیات طراحی کلاس ها(بلوک ها) است که برای پیاده سازی سیستم ضروری است.

پیاده سازی:

متدولوژی BON: در این متدولوژی صحبتی درباره پیاده سازی نشده است.

متدولوژی OOSE: در این متدولوژی در فاز پیاده سازی با استفاده از مشخصات پکیج ها و بلوک های تعریف شده در طراحی، کد ایجاد می شود.

تست:

متدولوژی BON: در BON92 صحبتی درباره تست نشده است. در BON95 هم به طور صریح صحبتی در مورد تست نشده است ولی شاید بتوان تاییدیه کاربران و متخصصان را روی تحویل دادنی ها برخی فاز ها نوعی تست قلمداد کرد.

متدولوژی OOSE: در این متدولوژی در فاز تست verificationو validation مدل پیاده سازی شده انجام می شود که Testing Model را تولید می کند که شامل برنامه ریزی تست، مشخصات تست و نتایج تست است. به طور معمول تست در سه سطح انجام می شود از پایین ترین سطح که بلوک ها هستند و بعد از آن Use Case ها و در نهایت کل سیستم تست می شود.

استقرار:

متدولوژی BON: در این متدولوژی صحبتی در مورد استقرار وجود ندارد.

متدولوژی OOSE: در این متدولوژی صحبتی در مورد استقرار وجود ندارد.

نگهداری:

متدولوژی BON: در این متدولوژی صحبتی در مورد نگهداری وجود ندارد.

متدولوژی OOSE: در این متدولوژی صحبتی در مورد نگهداری وجود ندارد.

معیار پشتیبانی از فعالیت های چتری:

متدولوژی باید به فعالیت های چتری توجه داشته باشد. فعالیت های چتری فعالیت هایی هستند که در همه بخش های چرخه فرآیند لحاظ می شوند و شامل مدیریت ریسک، مدیریت پروژه و تضمین کیفیت هستند.

مدیریت ریسک:

متدولوژی BON: در این متدولوژی صحبتی در مورد مدیریت ریسک وجود ندارد.

متدولوژی OOSE: در این متدولوژی صحبتی در مورد مدیریت ریسک وجود ندارد. اما می توان طراحی های واسط کاربر و نمایش به کاربران که در متدولوژی وجود دارد را نوعی از مدیریت ریسک برای کاهش ریسک ها در نظر گرفت.

مدیریت پروژه:

متدولوژی BON: در این متدولوژی صحبتی در مورد مدیریت پروژه به طور خاص وجود ندارد. البته تکالیف دارای ترتیب هستند که می توان آنها را به ترتیب دلخواه هم اجرا کرد اما همه تحویل دادنی ها باید تولید شوند. همچنین می توان مواردی مثل ایندکس کردن را که در متدولوژی وجود دارد به عنوان مدیریت پروژه در نظر گرفت.

متدولوژی OOSE: در این متدولوژی صحبتی در مورد مدیریت پروژه وجود ندارد. اما فازها باید با ترتیب اشاره شده تولید بشوند.

تضمین کیفیت:

متدولوژی BON: در این متدولوژی صحبت صریحی در مورد تضمین کیفیت پروژه وجود ندارد تنها در برخی فاز ها در BON95 در بخش تحویل دادنی ها تضمین کیفیت وجود دارد.

متدولوژی OOSE: در این متدولوژی صحبتی در مورد تضمین کیفیت پروژه وجود ندارد.

معیار بی‌درزی و گذر هموار:

بی درزی به معنی عدم وجود تغییر الگو در زنجیره ی مدل سازی و فعالیت ها است. گذار بین فازها، مراحل و تکالیف باید هموار باشند و باید ادامه طبیعی هم باشند و میان آنها وقفه ایجاد نشود.

بی درزی و گذار هموار:

متدولوژی BON: متدولوژی هدفش ساخت فرآیند بی درز است و با ایجاد مدل های میانی سعی در ایجاد بی‌درزی می کند البته به دلیل اینکه مدل ها از روی نیازمندی ها ساخته می شوند به طور طبیعی دارای درز هستند. تکمیل کردن به مرور مدل ها هم باعث همواری گذر بین مدل ها است.

متدولوژی OOSE: این متدولوژی چون از روی Use Caseها سایر مدل ها را می سازد ذاتاً دارای درز است و سعی می کند با ساختن مدل های میانی این مشکل را رفع کند و تکمیل کردن به مرور مدل ها هم باعث همواری گذر بین مدل ها است.

معیار مبتنی بودن بر نیازمندی ها:

کارها باید مبتنی بر نیازمندی ها باشند. نیازمندی های وظیفه ای و غیر وظیفه در ابتدای شروع متدولوژی باید استخراج شوند، مستقلاً مدل سازی شده و مبنای کارهای طراحی، پیاده سازی و تست قرار گیرند.

متدولوژی BON: در این متدولوژی تنها مدل های اولیه ساخته شده براساس نیازمندی های غیر رسمی کاربران هستند ولی بقیه مستقیما براساس نیازمندی نیستند و تقریبا همه مدل ها بر مبنای اشیا و کلاس ها هستند که از روی نیازمندی ها شناسایی می شوند و این اشیا به مرور تکمیل و تبدیل به کلاس می شوند.

متدولوژی OOSE: در این متدولوژی همه چیز بر اساس Use Case ها است که از روی نیازمندی ها بدست می آیند . متدولوژی کاملا مبتنی بر نیازمندی ها است.

معیار آزمون پذیری، ملموس بودن و قابل رهگیری بودن به نیازمندی ها:

محصولات باید آزمون پذیر، ملموس و قابل فهم و قابل نگاشت به نیازمندی ها باشند.

آزمون پذیری:

متدولوژی BON: در این متدولوژی تعداد مصنوعات تقریبا زیاد است البته ساده هستند و پیچیدگی کمی دارند و برای کاربران غیر فنی هم قابل فهم هستند و وابستگی ها و ارتباطات میان آنها هم به خوبی تعریف شده است و چون دارای ابزار برای مدل سازی است ناسازگاری های آن هم کم می باشد. با کمک ابزار CASE نیز می توان آزمون را انجام داد. همچنین معیارهای پذیرش در BON95 نیز به آزمون پذیر شدن متدولوژی کمک می کند.

متدولوژی OOSE: در این متدولوژی تعداد مصنوعات تقریبا زیاد است البته ساده هستند و پیچیدگی کمی دارند و برای کاربران غیر فنی هم قابل فهم هستند و وابستگی ها و ارتباطات میان آنها هم به خوبی تعریف شده است و مصنوعات همدیگر را تکمیل می کنند.

ملموس بودن:

متدولوژی BON: مدل های ساخته شده در این متدولوژی با توجه به نشان گذاری آنها برای همه قابل فهم می باشند به خصوص در BON95 که در برخی فازها نیاز به تایید کاربر وجود دارد نشان می دهد مدل ها طوری طراحی شده اند که برای کاربران غیر فنی هم قابل فهم باشند همچنین کاربرد آنها هم در متدولوژی مشخص شده است.

متدولوژی OOSE: مدل های ساخته شده در این متدولوژی هم با توجه نشانه گذاری های ساده برای همه قابل فهم می باشند.

قابل رهگیری به نیازمندی ها:

متدولوژی BON: این متدولوژی تا حدی امکان رهگیری نیازمندی ها را دارد. در BON95 مدل Scenario Chartبرای رهگیری نیازمندی ها مفید است.

متدولوژی OOSE: در این متدولوژی چون همه چیز بر پایه Use Case است امکان رهگیری مستقیم نیازمندی ها وجود دارد.

معیار تشویق به مشارکت فعالانه کاربران:

کاربران باید به طور فعالانه در فرآیند ایجاد نرم افزار دخالت داشته باشند که این کار در مدیریت مخاطرات و تضمین کیفیت اهمیت دارد.

متدولوژی BON: در BON92 صحبتی در مورد حضور فعالانه کاربران وجود ندارد. در BON95 در برخی فازها برای تایید فاز، نیاز به تاییدیه کاربران نهایی وجود دارد که می توان آن را نوعی حضور فعالانه کاربران در فرآیند دانست.

متدولوژی OOSE: در این متدولوژی صحبتی در مورد حضور فعالانه کاربران وجود ندارد.

معیار قابل اجرا بودن و کارا بودن:

باید بتوان فرآیند را به راحتی اجرا کرد و در ایجاد نرم افزار به کار گرفت همچنین باید بتوان از فرآیند به طور کارا در حوزه کاربرد استفاده کرد.

قابل اجرا بودن:

متدولوژی ***BON***: این متدولوژی در مجموع قابل اجرا است. مدل های ساخته شده ساده و قابل فهم هستند و چون هدف مشخصی دارند می تواند به راحتی اجرا شوند. البته چون برخی مدل ها کل سیستم را پوشش می دهند در سیستم های بسیار بزرگ ممکن است باعث پیچیده شدن مدل ها و در نتیجه پایین آمدن قابلیت اجرا باشد.

متدولوژی ***OOSE***: این متدولوژی مجموعا از قابلیت اجرای خوبی برخوردار است. مدل هایی که ساخته می شوند قابل فهم هستند البته ممکن است برخی مدل هایی که کل سیستم را پوشش می دهند کمی پیچیده باشند.

کارا بودن:

متدولوژی ***BON***: این متدولوژی در مجموع، قابل اجرا به صورت کارا، در حد خوبی است البته مشکل پیچیده شدن مدل هایی که کل سیستم را نشان می دهند می تواند کارایی اجرا را کاهش دهد. یکی دیگر از موارد کاهش دهنده کارایی اجرا وابسته بودن مدل ها به ابزار خاص CASE در این متدولوژی می باشد که برای مدل سازی حتما باید از ابزار هایی که خود متدولوژی مشخص کرده استفاده شود. البته ابزارهای خاص متدولوژی با قابلیت هایی که دارند امکان مدیریت موثر مدل ها و رفع ناسازگاری را هم می دهند که می تواند باعث افزایش کارایی اجرا گردد. در مورد استراتژی مدیریت فرآیند هم متدولوژی دست را برای ایجاد مدل ها باز گذاشته و ترتیب ندارد البته برای انجام برخی فعالیت ها حتما باید قبل از تمام مدل های مورد نیاز ساخته شده باشند.

متدولوژی ***OOSE***: این متدولوژی را می توان به صورت کارا اجرا نمود. البته مشکل پیچیده شدن مدل هایی که کل سیستم را نشان می دهند همچنین عدم وابستگی با ابزار خاص باعث افزایش کارایی می گردد. فرآیند هم به خوبی مدیریت می شود و مصنوعات خاصی حتما باید در مراحل ساخته بشوند.

معیار قابل مدیریت بودن پیچیدگی:

فرآیند از تعدادی واحد کاری، نقش، محصول و تعدادی معیار سنجش تشکیل می شود. باید خود این مجموعه از نظر پیچیدگی قابل مدیریت باشد. مدیریت معمولا به دو روش بخش بندی و لایه بندی انجام می شود.

متدولوژی BON: در این متدولوژی به دلیل توصیف دقیق فرآیند مشخص است که چه کاری در چه زمانی باید انجام و چه مدلی در کجا باید ساخته شود. کلاس بندی و خوشه بندی روش دیگری است که به مدیریت پیچیدگی ها کمک می کند. همچنین با کمک ابزار های خاص و با استفاده از ایندکس ها می توان کامپوننت‌ها را به طور دقیق مدیریت نمود. لایه های مختلف انتزاع در کلاس ها هم می تواند به مدیریت پیچیدگی کمک کند.

متدولوژی OOSE: در این متدولوژی به دلیل توصیف دقیق فرآیند مشخص است که چه کاری در چه زمانی باید انجام و چه مدلی در کجا باید ساخته شود. افراز بلوک ها به پکیج یک روش کم کردن پیچیدگی می باشد. همچنین قاعده Separation of concern که اعتقاد واضعان متدولوژی بوده می تواند به کم کردن پیچیدگی ها کمک کند.

معیار قابلیت گسترش، مقیاس پذیری، پیکربندی و انعطاف پذیری:

با بتوان فرآیند یک متدولوژی را با کمک نقاط گسترش و مکانیزم های تعریف شده ان گسترش داد، در پروژه های با انداز مختلف اعمال کرد، در ابتدای پروژه و مناسب موقعیت فعلی ان را پیکربندی نمود و بتوان در جریان اجرای فرآیند پیکربندی را تغییر داد و اصلاح کرد.

گسترش پذیری:

متدولوژی BON: قابلیت گسترش برای این متدولوژی وجود ندارد و نقاط گسترش و مکانیزم های گسترش برای ان تعریف نشده اند.

متدولوژی OOSE: قابلیت گسترش برای این متدولوژی وجود ندارد و نقاط گسترش و مکانیزم های گسترش برای ان تعریف نشده اند.

مقیاس پذیری:

متدولوژی BON: فرآیند های انتزاع و تعمیم که در طول فرآیند انجام می شود می تواند باعث افزایش مقیاس پذیری و اجرای متدولوژی روی پروژه با سایز های بزرگ باشد.

متدولوژی OOSE: ذات شی گرایی متدولوژی می تواند باعث شود بتوان آن را در پروژه با اندازه های مختلف اجرا کرد.

پیکربندی:

متدولوژی BON: این متدولوژی قابلیت پیکربندی فرآیند در ابتدای پروژه را ندارد.

متدولوژی OOSE: این متدولوژی قابلیت پیکربندی فرآیند در ابتدای پروژه را ندارد.

انعطاف پذیری:

متدولوژی BON: نمی توان در طول اجرای فرآیند آن را تغییر داد تنها می توان با رعایت برخی محدودیت ها ترتیب ساخت مدل ها را به دلخواه انتخاب نمود.

متدولوژی OOSE: نمی توان در طول اجرای فرآیند تغییری در متدولوژی اعمال نمود.

معیار حوزه کاربرد:

متدولوژی در چه حوزه هایی کاربرد دارد و برای چه حوزه هایی مناسب نیست حداقل باید بتواند حوزه کاربرد سیستم های اطلاعاتی را پوشش دهد.

متدولوژی BON: این متدولوژی به دلیل ذات شی گرا قابلیت اجرا روی انواع مختلف پروژه ها را دارد. در BON92 سیستم اجاره خودرو و در BON95 سیستم مدیریت کنفرانس و سیستم ضبط ویدیو به عنوان نمونه های موردی آن مطرح شده اند.

متدولوژی OOSE: این متدولوژی به دلیل ذات شی گرا قابلیت اجرا روی انواع مختلف پروژه ها را دارد. سیستم مدیریت انبار و سیستم سوئیچینگ مخابراتی به عنوان نمونه موردی استفاده از این متدولوژی مطرح شده اند. البته به دلیل عدم پشتیبانی از مدل سازی صوری که قابلیت اثبات عملکرد را می دهد ممکن است برای برخی کاربردها مناسب نباشد.

معیارهای زبان مدل سازی:

معیار پشتیبانی از مدل سازی شی گرا سازگار، دقیق و بدون ابهام:

مدل ها باید با کیفیت و شامل هر چه در یک مدل شی گرا نیاز است باشد و بتوان به وضوح شی گرایی را در آن دید، مدل ها نباید یکدیگر را نقض کنند، مدل باید صحیح باشند و اشتباه نداشته باشند همچنین باید با جزئیات کامل باشد و زبان مدل سازی اینها را پشتیبانی کند.

دیدگاه های مختلف مدل سازی:

متدولوژی BON: این متدولوژی فقط دیدهای رفتاری و ساختاری مدل سازی را دارد و مدلی برای دید وظیفه ای ندارد.

متدولوژی OOSE: این متدولوژی هر سه دیدهای رفتاری، وظیفه ای و ساختاری را برای مدل سازی دارد

مدل سازی از منطقی به فیزیکی:

متدولوژی BON: این متدولوژی با اضافه کردن تصمیمات مربوط به پیاده سازی به مدل های منطقی می تواند دید فیزیکی قابل پیاده سازی را بسازد. در این متدولوژی فرمالیسم موجود در آن به تحلیل گر کمک می کند تا دید منطقی را به راحتی به دید فیزیکی قابل پیاده سازی تبدیل کند.

متدولوژی OOSE: این متدولوژی مدل سازی را به صورت کامل در قلمرو مسئله و قلمرو راه حل انجام می‌دهد و با اضافه کردن جزئیات پیاده سازی به دید های منطقی دید فیزیکی ساخته می شود.

سطوح مختلف انتزاع و دانه بندی:

متدولوژی BON: این متدولوژی مدل سازی را در سطح سیستم شروع و تا سطح کلاس ها ادامه می دهد.

متدولوژی OOSE: این متدولوژی مدل سازی را در سطح سیستم شروع و تا سطح کلاس ها ادامه می دهد.

مدل سازی صوری و غیر صوری:

متدولوژی BON: این متدولوژی دارای زبان مدل سازی بصری و متنی خاص خود می باشد و ابزاری هم برای مدل سازی در اختیار قرار می دهد. همچنین در مدل سازی متنی گرامر خاص که برای انجام کارهای خودکار مناسب است را هم دارد که در BON95 معرفی شده است.

متدولوژی OOSE: این متدولوژی تنها دارای مدل سازی بصری است و مدل سازی متنی ندارد.

معیار مدیریت تناقض ها و پیچیدگی ها:

پیچیدگی مدل ها باید مدیریت شود و از بوجود آمدن ناسازگاری بین آنها جلوگیری نمود. زبان مدل سازی باید برای تشخیص و رفع ناسازگاری و پیچیدگی مدل ها استراتژی و تکنیک های مناسب داشته باشد.

متدولوژی BON: این متدولوژی ابزارهایی دارد که با کمک ایندکس ها می تواند کامپوننت ها را مدیریت و ناسازگاری های آنها را تشخیص دهد و رفع کند. همچنین روابط دقیق بین مدل ها به گونه ای تعریف شده

که می توان پیچیدگی را مدیریت و ناسازگاری را کاهش داد. روش های خوشه بندی و کلاس بندی هم کمک به مدیریت پیچیدگی ها می نماید.

متدولوژی ***OOSE***: در این متدولوژی روابط دقیق بین مدل ها به طور دقیق تعریف شده که می تواند پیچیدگی را مدیریت و ناسازگاری های احتمالی را کاهش داد. افراز بلوک ها در پکیج ها یکی از روش های مدیریت پیچیدگی در این متدولوژی می باشد.

مقایسه فرآیند عمومی متدولوژی های OPM و BOM :

متدولوژی BON یا Better Object Notation در سال ۱۹۹۲ توسط نرسن در مقاله ای معرفی شد. نسخه بازبینی شده و با جزئیات بیشتر در سال ۱۹۹۵ معرفی که این بار B در آن مخفف کلمه Business بود. متدولوژی تلاش می کند مستقل از زبان باشد اما به طور عمیقی تحت تاثیر مکانیزم های Eiffel و رویکرد Design by Contract می باشد. چرخه عمر عمومی این متدولوژی شامل فاز تحلیل و طراحی می باشد که خودشان شامل ۹ تکلیف هستند. ٦ تکلیف مربوط به تحلیل و ۳ تکلیف مربوط به طراحی می باشند.

می توان ترتیب تکالیف را برای رسیدن به اهداف پروژه تغییر داد اما در نهایت باید تمام مدل های مورد نیاز ساخته شوند. هر تکلیف دارای تعدادی منابع ورودی است که توسط معیار پذیرش بازبینی و مجموعه ای از تحویل دادنی ها تولید می شوند. تحویل دادنی ها توصیف ایستا و پویا از سیستم در حال ساخت را نمایش می دهند.

مدل ایستا شامل توصیف رسمی کلاس ها، گروه بندی آنها در کلاسترهایی به عنوان مشتری–سرویس دهنده و روابط توارث بین آنهاست که در نهایت نمایش دهنده ساختار سیستم هستند. مدل پویا رویدادهای سیستم را نشان می دهد، مشخص کننده اشیائی است که مسئولیت ساخت اشیا دیگر را دارند و همچنین سناریوهای اجرایی سیستم که بازنمایی انواع مورد استفاده در سیستم، همراه با نمودار های نمایش دهنده تبادل پیام میان اشیا می باشد. ۹ تکلیف متدلوژی در بخش زیر آمده است:

۱. ***تعیین مرزهای سیستم:*** هدف مشخص کردن دید اصلی از دنیایی که باید تحقق یابد و همچنین سیستمی است که باید مدل شود. با انجام این تکلیف System Chart و Scenario Chart تولید می شوند. تاییدیه مشتری و برنامه های کنترل کیفیت معیار پذیرش این تکلیف می باشند.
۲. ***لیست کردن کلاس های کاندید:*** استخراج کلاس های کاندید از قلمرو مسئله انجام می شود. با انجام این تکلیف Cluster Chart ساخته می شود. تاییدیه کاربران نهایی و کارشناسان معیار پذیرش این تکلیف است.
۳. ***انتخاب کلاس ها و گروه بندی آنها در کلاستر ها:*** با کمک لیستی از کلاس های کاندید تولید شده، مفاهیمی را انتخاب و آنها را به عنوان کلاس مدل می کند. با انجام این تکلیف Static Architecture و Class Dictionary تولید می شوند. سیستم مدیریت کلاس و کلاستر که به طور خودکار سازگاری کلاس ها را بررسی می کند معیار پذیرش این تکلیف است.

۴. ***تعریف کلاس ها:*** با داشتن مجموعه اولیه کلاس ها و گروه بندی آنها کلاس ها از نظر حالت(اطلاعاتی که فراهم می کنند)، رفتار(عملیاتی که می توانند انجام دهند) و قواعدی که باید توسط کلاس و سرویس گیرندگانش رعایت شود تعریف می شوند. کلاس ها شامل توابعی که اطلاعات را بدون تغییر حالت بر می گردانند (queries)، دستوراتی که هیچ اطلاعاتی بر نمی گردانند اما ممکن است حالتی را تغییر دهند (commands) و محدودیت هایی که قواعد عمومی تجاری و شرط سازگاری هستند (contracts) می باشند. با انجام این تکلیف Class Chart تولید می شود. تاییدیه مشتری و کاربر نهایی معیار پذیرش این تکلیف است.

۵. ***ترسیم رفتار سیستم:*** مدل پویا سیستم توصیف می شود. یک نمودار سناریو اولیه، نوع های مهم سیستم را که قبلا ساخته شده اند را جمع می کند که ارزش زیادی برای یافتن کلاس های کاندید اولیه و انتخاب بین دیدهای دیگر قلمرو دارند. با انجام این تکلیف Event Chart، Creation Chart و Object Scenario تولید می شوند. سازگاری با مدل ایستا و تاییدیه کاربران نهایی و مشتریان معیار پذیرش این تکلیف هستند.

۶. ***تعریف ویژگی های عمومی:*** توصیف غیر رسمی کلاس ها در نمودار کلاس، به اینترفیس های رسمی کلاس ها با قراردادهای نرم افزاری ترجمه می شوند. Queries به functions ، commands به procedures و contracts تبدیل به Pre/Post Condition برای توابع و ثوابت روی کل کلاس می شوند. signature هر عملیات عمومی هم مشخص می شود که نوعا می تواند باعث تعریف روابط جدید بین کلاس ها بشود. با انجام این تکلیف Class Interface تولید می شود. سازگاری با static architecture و پوشش رویدادهای خروجی در event chart و سناریوهای انتخاب شده در Scenario chart توسط ویژگی ها معیار پذیرش این تکلیف هستند.

۷. ***پالا یش سیستم:*** شروع کننده بخش طراحی فرآیند است و تکرار فعالیت هایی است که قبلا برای تحلیل کلاس ها انجام شده اند و روی طراحی جدید کلاس ها اعمال می شوند. در این فعالیت تنها مصنوعات قبلی پالایش می شوند و جزئیاتشان تکمیل می گردد. سازگاری با Object Creation Chart و Object Scenario معیار پذیرش این تکلیف هستند.

۸. ***عمومی سازی:*** بهبود وراثت سلسله مراتبی کلاسها با فاکتورگیری حالات و رفتارهای مشترک در این تکلیف انجام می شوند. در این فعالیت تنها مصنوعات قبلی پالایش می شوند. سازگاری با Object Scenario معیار پذیرش این تکلیف هستند.

۹. ***تکمیل و بازبینی سیستم:*** مدل ها جلا داده و کامل می شوند و سازگاری کل سیستم بررسی می شود. در این فعالیت تنها مصنوعات قبلی پالایش می شوند. اعتبارسنجی نحوی کلاس ها، سازگاری ارث‌بری و ثابت های کلاس های ترکیب شده و سازگاری پیش شرط و پس شرط روتین های چند ریختی معیار پذیرش این تکلیف هستند.

متدولوژی OPM یا Object Oriented Methodology در سال ۱۹۹۵ توسط دوری به عنوان یک رویکرد نو در مدل سازی تحلیلی که طرفدار ترکیب رویکرد مدل سازی برپایه فرآیند کلاسیک و تکنیک مدل سازی شی گرا بود معرفی شد. رویکرد منحصر به فرد متدلوژی هنوز هم موضوعی جذاب برای محققان است.

قدرت مدل سازی OPM از این واقعیت سرچشمه می گیرد که تنها یک مدل برای مدل سازی ساختاری، رفتاری و وظیفه ای آن وجود دارد. این تک مدله بودن ناسازگاری های تجمیع و تبدیل مدل ها را رفع ولی مدل را بسیار پیچیده می کند. مدل متدولوژی OPD نام دارد که از عناصری از نوع object و process برای مدل سازی رفتاری، ساختاری و وظیفه ای جنبه هایی که باید مدل شوند استفاده می کند. OPD دارای عناصری از نوع state هم می باشد که به طور خاص برای مدل سازی سیستم های بلادرنگ مفید هستند. همچنین برای انواع دیگری از سیستم ها مانند وب‌کاربردها، وب سرویس های معنایی و سیستم های چند عامله نشان گذاری های خاص ساخته شده است.

OPD را می توان با کمک OPL که یک زبان طبیعی محدود است به صورت متنی نمایش داد. معادل های OPL از مدل OPD به صورت خودکار ساخته می شوند که ساخت کد از روی آنها ساده تر می باشد و برای کاربران و متخصصان قابل استفاده تر هستند.

چرخه عمر عمومی متدولوژی شامل فرآیند های آغاز، ایجاد و استقرار است.

۱. **آغاز:** تمرکز روی تحلیل مقدماتی سیستم، تعیین قلمرو سیستم، منابع مورد نیاز و نیازمندی های سطح بالا می باشد. این فرآیند شامل سه زیر فرآیند می باشد.

*آ. **شناسایی:*** نیازها یا فرصت هایی که توجیه کننده ایجاد سیستم هستند مشخص می شوند.

*ب. **ادراک:*** سیستم از طریق تعیین قلمرو ادراک می شود و از اینکه منابع مورد نیاز برای ساخت سیستم در دسترس هستند اطمینان حاصل می گردد.

*پ. **پیکربندی:*** نیازمندی های سطح بالای سیستم تعیین می شوند.

۲. **ایجاد**: تمرکز روی تحلیل تفصیلی، طراحی و پیاده سازی است. . این فرآیند شامل سه زیر فرآیند می باشد.

*آ. **تحلیل**:* نیازمندی ها استخراج و قلمرو مسئله و سیستم در OPD و معادل OPL آن مدل سازی می شود. اسکلت معماری سیستم هم انتخاب می شود.

*ب. **طراحی**:* فعالیت هایی مانند اضافه کردن جزئیات زمان پیاده سازی به مدل ها و پالایش معماری با تعیین سخت افزارها، میان افزارها و کامپوننت های نرم افزاری انجام می شود.

*پ. **پیاده سازی**:* ساخت کامپوننت های سیستم و ارتباط دادن آنها به یکدیگر است که شامل کد زنی و تست کامپوننت های نرم افزاری می باشد.

۳. **استقرار**: تمرکز روی معرفی سیستم به محیط کاربر و فعالیت های متعاقب آن که نگهداری سیستم در طول عمر سیستم می باشد است. . این فرآیند شامل چهار زیر فرآیند می باشد.

*آ. **وفق دادن**:* معرفی سیستم به محیط کاربر که عمدتا شامل آموزش، ایجاد مستندات مناسب، تبدیل داده ها و سیستم و تست های پذیرش می باشد.

*ب. **استفاده و نگهداری**:* دوره ای که سیستم در حال استفاده است را پوشش می دهد. فعالیت هایی انجام می شوند که برای بالا نگه داشتن سیستم و ادامه کار آن ضروری هستند.

*پ. **ارزیابی وظیفه مندی**:* بررسی می کند فرآیند های سیستم جاری وظیفه مندی های خود را انجام و نیازمندی ها را محقق می کنند یا نه. معمولا در زمان استفاده و نگهداری برای تعیین اینکه سیستم هنوز نیازمندی ها را تحقق می بخشد یا نه استفاده می شود. که اگر تحقق نبخشد فعالیت بعدی فعال و باید سیستم جدیدی ایجاد شود.

*ت. **پایان**:* مرگ سیستم جاری اعلام و فرآیند های پسامرگی که باعث ایجاد نسل جدید سیستم می شود انجام می پذیرند.

در ادامه این دو متدولوژی با کمک معیار های موجود مورد مقایسه قرار می گیرند.

معیارهای فرآیند:

معیار تعریف:

متدولوژی باید به خوبی تعریف شده باشد و تعریف آن جامع، روشن، منطقی، دقیق، با جزئیات، سازگار و باید به خوبی مستند شده باشد.

چرخه عمر و واحدهای کاری:

متدولوژی BON: در BON92 چرخه عمر متدولوژی و واحدهای کاری که باید در طول فرآیند انجام شود به صورت کلی با عنوان تحلیل و طراحی مشخص شده. در BON95 فعالیت های فرآیند در سه گروه با اهداف تقریبا مشابه دسته بندی شده‌اند.

متدولوژی OPM: چرخه عمر و واحد های کاری که باید در طول فرآیند اجرا شوند به خوبی و با جزئیات کامل بیان شده است. فعالیت های متدلوژی در سه فاز تقسیم بندی شده اند.

مقایسه: تعریف چرخه عمر و واحد های کاری در OPM بسیار دقیق تر و بسیار با جزئیات است و از این لحاظ این متدولوژی برتری دارد.

نقش ها و افراد:

متدولوژی BON: در BON92 اشاره ای به این موضوع نشده اما با دقت در BON95 می توان برخی نقش ها شامل Customer، End-User، Domain Experts، Developer، Analyst و Designer را به طور ضمنی مشاهده کرد.

متدولوژی OPM: در این متدولوژی اشاره خاصی به نقش افراد به طور صریح نشده است ولی به نقش هایی مثل Customer و Architecture Team که شامل System Architect، Integrator، Analyst، Designer و Developer در مستند متدولوژی اشاره شده است.

مقایسه: در هیچ یک از دو متدولوژی به نقش ها اشاره دقیق نشده است و هیچ کدام در این زمینه بر دیگری برتری ندارد.

زبان مدل سازی:

متدولوژی BON: این متدولوژی زبان خاص مدل سازی خود را تعریف کرده است .

متدولوژی OPM: این متدولوژی زبان خاص متنی و بصری خود را دارد.

مقایسه: متدولوژی OPM علاوه بر زبان بصری زبان متنی هم دارد که از این لحاظ بر BON برتری دارد. اما از نظر تعداد مدل ها چون کلا یک مدل دارد در این زمینه با توجه به مدل های متعدد BON قوی تر است. البته OPM دارای ابزاری است که در صورت نیاز می توان برخی از مدل های UML را از مدل موجود آن استخراج کرد که می تواند به بهتر شدن آن کمک نماید.

محصولات:

متدولوژی BON: محصولاتی که در هر تکلیف در این متدولوژی ایجاد می شوند به خوبی مشخص شده است و در BON95 به آنها تحویل‌دادنی گفته می شود که ده مصنوع تولید می شود و محصولات به صورت بصری هم نمایش داده شده اند که به فهم متدولوژی کمک می کند.

متدولوژی OPM: در این متدولوژی تنها یک مدل که OPD نام دارد ساخته می شود و در مراحل مختلف تکمیل تر خواهد شد. البته مستندات متنی مانند مستند فنی، معماری، توجیه اقتصادی هم در متدولوژی ایجاد می شود. Analyze OP Document و Design OP Document از دیگر محصولات تولید شده در متدلوژی است.

مقایسه: متدولوژی BON با توجه به تعدد محصولات تولید شده از این لحاظ قوی تر از OPM است البته OPM دارای ابزاری است که در صورت نیاز می توان برخی از مدل های UML را از مدل موجود آن استخراج کرد و مصنوعات بیشتری را در صورت نیاز در پروژه تولید کرد. از لحاظ مستندات متنی OPM قوی تر عمل می کند.

تکنیک ها و قواعد:

متدولوژی BON: در BON92 تکنیک ها و قواعد تا حدی تعریف شده اند و در BON95 جزئیات بیشتری به آن اضافه شده است و چگونگی انجام فعالیت ها مشخص شده اند.

متدولوژی OPM: قواعد و تکنیک های متدولوژی بسیار دقیق و با جزئیات بسیار خوبی توصیف و حوه اجرای فعالیت های مختلف هم با جزئیات مشخص شده اند.

مقایسه: با مقایسه دو متدولوژی از لحاظ بیان جزئیات OPM بسیار غنی تر و دقیق تر جزئیات هر فاز را توضیح داده است.

فعالیت های چتری:

متدولوژی BON: در BON92 اشاره ای به فعالیت های چتری نشده و در BON95 هم از مصنوعاتی که تولید می شود به صورت ضمنی می توان فهمید تضمین کیفیت وجود دارد که در تکلیف ۱ آن و در بخش معیار های پذیرش با عنوان برنامه ریزی برای تضمین کیفیت آمده است. همچنین اضافه کردن سرآیند و ایندکس به کلاس ها را که حاوی اطلاعاتی در مورد پیاده سازی و رهگیری کلاس ها است را شاید بتوان بخشی از مدیریت پروژه نامید.

متدولوژی OPM: در متدولوژی اشاره دقیقی به فعالیت های چتری نشده است اما از فرآیند های فاز identifying در مرحله initiating می توان فعالیت های مرتبط را که عمدتا سازمانی هستند را استخراج کرد. به طور مثال یافتن توجیه اقتصادی نوعی مدیریت ریسک می باشد. یا برنامه ریزی استراتژی در همین فاز نوعی فعالیت مدیریتی است که البته در همین فاز انجام و کل فرآیند را پوشش نمی دهند. همچنین فرآیند مربوط به فاز نگهداری هم نوعی فعالیت مدیریتی و تضمین کیفیت سیستم است.

مقایسه: در هر دو متدولوژی اشاره صریحی به فعالیت های چتری نشده اما با دقت در فازهای آن ها می توان دید OPM از لحاظ پوشش برخی فعالیت های چتری بهتر از BON می باشد و این مورد به نظر از اساس دغدغه سازندگان BON نبوده است.

معیار پوشش چرخه عمر عمومی ایجاد نرم‌افزار:

چرخه عمر عمومی شامل تعریف، ایجاد و نگهداری است که اگر این سه مرحله کامل پوشش داده شوند چرخه عمر کامل است.

تعریف:

کاوش قلمرو مسئله و مدل سازی آن:

متدولوژی BON: در BON92 با یافتن، نامگذاری و خوشه بندی کلاس ها و ساخت مدل Cluster Chart مرزهای سیستم مشخص می شود.Dynamic Model نشان دهنده ویژگی های رفتاری و Static Model نشان دهنده ویژگی های ساختاری سیستم است. در Event Chart با مشخص شدن رویداد های داخلی و خارجی نیز به نوعی مرزهای سیستم مشخص می شوند همچنین رفتارها را هم مشخص می کند. در BON95 مرزهای سیستم با System Chart و Scenario Chart مشخص می شوند. ویژگی های رفتاری سیستم هم با Event Chart، Creation Chart و Object Scenario مشخص می شوند.

متدولوژی OPM: در این متدولوژی در فاز conceiving قلمرو مسئله مشخص و مرز سیستم تشخیص داده می شود و چیز های که جز قلمرو مسئله هستند و نیستند با یک دید سطح بالا دقیقا مشخص می شود.

مقایسه: متدولوژی OPM یک فاز کامل را به این کار اختصاص داده است و هدفمند است و از این لحاظ بهتر از BON است که در طول فرآیند و با مدل های متعدد این کار را انجام می دهد البته از نظر تعدد مدل ها و نمایش جنبه های مختلف سیستم BON بهتر از OPM است.

استخراج نیازمندی ها:

متدولوژی BON: در این متدولوژی صحبتی از استخراج نیازمندی ها وجود ندارد و مدل ها بر مبنای نیازمندی‌های غیر رسمی کاربران ساخته می شوند.

متدولوژی OPM: در فاز initializing نیازمندی های سطح بالا که در قلمرو مسئله هستند در حد ۱۰ تا ۲۰ درصد استخراج می شود که خیلی جزئیات ندارند و جزئیات کامل آنها و همچنین بقیه نیازمندی های موجود در طول پروژه بدست می آید. همچنین استخراج نیازمندی ها در فاز analysing انجام می شود.

مقایسه: متدولوژی BON اصلا صحبتی از استخراج نیازمندی وجود ندارد و مبنای کار نیازمندی های غیر رسمی کاربران است اما اختصاص یک فاز کامل در OPM به این کار نشان دهنده برتری آن نسبت به BON می باشد.

امکان سنجی:

متدولوژی BON: در این متدولوژی صحبتی درباره امکان سنجی نشده است

متدولوژی OPM: امکان سنجی و تعیین اینکه آیا انجام پروژه شدنی است یا نه در فاز identifying انجام و باید از لحاظ مختلف عملیاتی، مالی ، فنی، زمان بندی، فرهنگی و ... امکان پذیری سیستم سنجیده شود.

مقایسه: متدولوژی BON امکان سنجی ندارد اما OPM با انجام این کار از این نظر برتر از BON می باشد.

ایجاد:

طراحی معماری:

متدولوژی BON: معماری در این متدولوژی با کمک خوشه بندی کلاس ها و به نوعی مشخص شدن زیر سیستم و دقیق تر کردن مدل ها در فازهای بعد بدست می آید.

متدولوژی OPM: در فازهای analyzing و designing این متدولوژی علاوه بر مدل سازی قلمرو مسئله اسکلت معماری سیستم انتخاب و با اضافه کردن جزئیات زمان پیاده سازی به مدل ها با تعیین سخت افزار، میان افزار و کامپوننت های نرم افزاری معماری سیستم تکمیل می شود.

مقایسه: متدولوژی OPM طراحی و تکمیل معماری را به طور دقیق تر و با جزئیات پیاده سازی انجام می دهد. اما BON علاوه بر این که کار را صراحتا انجام نمی دهد عملا به دلیل نداشتن مرحله پیاده سازی عملا وارد فاز جزئیات پیاده سازی معماری نمی شود. در این معیار OPM قوی تر می باشد.

طراحی تفصیلی:

متدولوژی BON: در BON92 با انجام فعالیت های توصیف، ایندکس کردن و نمونه سازی از کلاس ها که فعالیت های مرتبط با قلمرو راه حل هستند وارد فاز تحلیل تفصیلی می شود و تا انتها ادامه پیدا می کند. در BON95 با اضافه کردن جزئیات به مدل ها و یافتن کلاس های جدید و ویژگی های جدید، عمومی سازی که به معنی فاکتور گیری از ویژگی های مشترک کلاس‌ها است و کامل کردن و بازبینی سیستم طراحی تفصیلی انجام می پذیرد.

متدولوژی OPM: در این متدولوژی طراحی تفصیلی با اضافه کردن جزئیات زمان پیاده سازی به مدل های تحلیل در فاز designing ، به طور کامل تا پایین تر سطح انجام می شود.

مقایسه: با مقایسه دو متدولوژی به نظر فهم طراحی تفصیلی در BON به دلیل داشتن مدل های متعدد راحت تر از OPM می باشد. البته OPM مدل های دقیق تری حتی با جزئیات زمان پیاده سازی دارد که از این نظر بهتر از BON است.

پیاده سازی:

متدولوژی BON: در این متدولوژی صحبتی درباره پیاده سازی نشده است.

متدولوژی OPM: ساخت کامپوننت های سیستم و ارتباط آنها به یکدیگر در فاز implementing انجام می شود.

مقایسه: با توجه به اینکه BON این فاز را پوشش نمی دهد OPM در این معیار بهتر از BON است.

تست:

متدولوژی BON: در BON92 صحبتی درباره تست نشده است. در BON95 هم به طور صریح صحبتی در مورد تست نشده است ولی شاید بتوان تاییدیه کاربران و متخصصان را روی تحویل دادنی ها برخی فاز ها نوعی تست قلمداد کرد.

متدولوژی OPM: تست سیستم پیاده سازی شده در این متدولوژی در فاز implementing انجام می شود که شامل تست های ریزدانه می باشد. پس از استقرار هم تست های درشت دانه سیستمی و تست پذیرش کاربران روی سیستم انجام می شود. بررسی انجام نیازمندی های وظیفه ای در فاز evaluation functionality را هم می توان نوعی تست در نظر گرفت.

مقایسه: متدولوژی OPM با پوشش انواع تست ها در نوع های مختلف بسیار قوی تر از BON که پوششی در این زمینه ندارد است.

استقرار:

متدولوژی BON: در این متدولوژی صحبتی در مورد استقرار وجود ندارد.

متدولوژی OPM: مراحل استقرار سیستم در محیط کاربر در فاز assimilating انجام می شود. آموزش ها، ایجاد مستندات، تبدیل داده ها و تست های پذیرش هم جز فرآیند استقرار است.

مقایسه: متدولوژی OPM به دلیل پوشش فرآیند استقرار قوی تر از BON است.

نگهداری:

متدولوژی BON: در این متدولوژی صحبتی در مورد نگهداری وجود ندارد.

متدولوژی OPM: در فاز using and maintaining عملیات نگهداری انجام می شود که شامل بررسی های دوره ای روی سیستم می باشد. همچنین کیفیت کد باید حفظ شود و ایرادات و مشکلات باید رفع بشوند.

مقایسه: متدولوژی OPM با توجه به اینکه فاز نگهداری را دارد بهتر از BON است.

معیار پشتیبانی از فعالیت های چتری:

متدولوژی باید به فعالیت های چتری توجه داشته باشد. فعالیت های چتری فعالیت هایی هستند که در همه بخش های چرخه فرآیند لحاظ می شوند و شامل مدیریت ریسک، مدیریت پروژه و تضمین کیفیت هستند.

مدیریت ریسک:

متدولوژی BON: در این متدولوژی صحبتی در مورد مدیریت ریسک وجود ندارد.

متدولوژی OPM: به طور صریح صحبت از این فعالیت وجود ندارد امام فعالیت های مربوط به یافتن توجیه اقتصادی را می توان نوعی مدیریت ریسک دانست.

مقایسه: متدولوژی BON با نداشتن این فاز هر چند به طور ضمنی ضعیف تر از OPM است.

مدیریت پروژه:

متدولوژی BON: در این متدولوژی صحبتی در مورد مدیریت پروژه به طور خاص وجود ندارد. البته تکالیف دارای ترتیب هستند که می توان آنها را به ترتیب دلخواه هم اجرا کرد اما همه تحویل دادنی ها باید تولید شوند. همچنین می توان مواردی مثل ایندکس کردن را که در متدولوژی وجود دارد به عنوان مدیریت پروژه در نظر گرفت.

متدولوژی OPM: فعالیت های برنامه ریزی استراتژی، امکان سنجی، برنامه ریزی برای فرآیند های نگهداری را می توان مدیریت پروژه در این متدولوژی دانست.

مقایسه: متدولوژی OPM با توجه به فعالیت های دقیق تری که در این زمینه دارد بهتر از BON است.

تضمین کیفیت:

متدولوژی BON: در این متدولوژی صحبت صریحی در مورد تضمین کیفیت پروژه وجود ندارد تنها در برخی فاز ها در BON95 در بخش تحویل دادنی ها تضمین کیفیت وجود دارد.

متدولوژی OPM: به طور صریحی صحبت از تضمین کیفیت در متدولوژی وجود ندارد. اما فعالیت مربوط به حفظ کیفیت کد را می توان تضمین کیفیت دانست.

مقایسه: با توجه به عدم وجود دقیق این فعالیت در هر دو متدولوژی هیچ یک برتری بر دیگری ندارد.

معیار بی‌درزی و گذر هموار:

بی درزی به معنی عدم وجود تغییر الگو در زنجیره ی مدل سازی و فعالیت ها است. گذار بین فازها، مراحل و تکالیف باید هموار باشند و باید ادامه طبیعی هم باشند و میان آنها وقفه ایجاد نشود.

بی درزی و گذار هموار:

متدولوژی BON: متدولوژی هدفش ساخت فرآیند بی درز است و با ایجاد مدل های میانی سعی در ایجاد بی‌درزی می کند البته به دلیل اینکه مدل ها از روی نیازمندی ها ساخته می شوند به طور طبیعی دارای درز هستند. تکمیل کردن به مرور مدل ها هم باعث همواری گذر بین مدل ها است.

متدولوژی OPM: با داشتن تنها یک مدل این متدولوژی قاعدتا بی درز است البته داشتن تنها یک مدل و به دلیل نبود مدل های رفتاری و عدم امکان نمایش اشیا و پیام هایشان بی درزی دچار مشکل خواهد شد و با توجه به داشتن یک مدل گذر آن هم هموار است.

مقایسه: متدولوژی OPM هرچند می تواند دارای درز باشد اما به دلیل داشتن تنها یک مدل از BON بهتر است.

معیار مبتنی بودن بر نیازمندی ها:

کارها باید مبتنی بر نیازمندی ها باشند. نیازمندی های وظیفه ای و غیر وظیفه در ابتدای شروع متدولوژی باید استخراج شوند، مستقلاً مدل سازی شده و مبنای کارهای طراحی، پیاده سازی و تست قرار گیرند.

متدولوژی BON: در این متدولوژی تنها مدل های اولیه ساخته شده براساس نیازمندی های غیر رسمی کاربران هستند ولی بقیه مستقیما براساس نیازمندی نیستند و تقریبا همه مدل ها بر مبنای اشیا و کلاس ها هستند که از روی نیازمندی ها شناسایی می شوند و این اشیا به مرور تکمیل و تبدیل به کلاس می شوند.

متدولوژی OPM: با توجه به عدم مدل سازی نیازمندی ها در این متدولوژی نمی توان این متدولوژی را مستقیما مبتنی بر نیازمندی ها دانست.

مقایسه: با توجه به عدم مبتنی بر نیازمندی ها بودن هر دو متدولوژی هیچ کدام برتری بر دیگری ندارد.

معیار آزمون پذیری، ملموس بودن و قابل رهگیری بودن به نیازمندی ها:

محصولات باید آزمون پذیر، ملموس و قابل فهم و قابل نگاشت به نیازمندی ها باشند.

آزمون پذیری:

متدولوژی BON: در این متدولوژی تعداد مصنوعات تقریبا زیاد است البته ساده هستند و پیچیدگی کمی دارند و برای کاربران غیر فنی هم قابل فهم هستند و وابستگی ها و ارتباطات میان آنها هم به خوبی تعریف شده است و چون دارای ابزار برای مدل سازی است ناسازگاری های آن هم کم می باشد. با کمک ابزار CASE

نیز می توان آزمون را انجام داد. همچنین معیارهای پذیرش در BON95 نیز به آزمون پذیر شدن متدولوژی کمک می کند.

متدولوژی OPM: این متدولوژی علی رغم داشتن یک مدل به دلیل پیچیده بودن زیاد مدل قابلیت آزمون پذیری پایین دارد.

مقایسه: متدولوژی BON به دلیل داشتن مدل های بیشتر و ساده تر بودن مدل ها قابلیت آزمون پذیری بیشتری دارد.

ملموس بودن:

متدولوژی BON: مدل های ساخته شده در این متدولوژی با توجه به نشان گذاری آنها برای همه قابل فهم می باشند به خصوص در BON95 که در برخی فازها نیاز به تایید کاربر وجود دارد نشان می دهد مدل ها طوری طراحی شده اند که برای کاربران غیر فنی هم قابل فهم باشند همچنین کاربرد آنها هم در متدولوژی مشخص شده است.

متدولوژی OPM: این متدولوژی علاوه بر نداشتن جنبه های رفتاری به دلیل پیچیده بودن زیاد برای کاربران کمتر ملموس است.

مقایسه: ملموس بودن BON به دلیل تنوع مدل های آن و پوشش بیشتر برخی جنبه های مدل سازی بیشتر است.

قابل رهگیری به نیازمندی ها:

متدولوژی BON: این متدولوژی تا حدی امکان رهگیری نیازمندی ها را دارد. در BON95 مدل Scenario Chartبرای رهگیری نیازمندی ها مفید است.

متدولوژی OPM: این متدولوژی به دلیل نداشتن جنبه رفتاری و عدم امکان شناسایی روابط بین کلاس ها و پیام هایشان و همچنین به دلیل نداشتن مدل برای نیازمندی ها قابلیت رهگیری این متدولوژی پایین است.

مقایسه: متدولوژی BON با توجه به مدل هایی که دارد قابلیت رهگیری بسیار بیشتری از OPM دارد.

معیار تشویق به مشارکت فعالانه کاربران:

کاربران باید به طور فعالانه در فرآیند ایجاد نرم افزار دخالت داشته باشند که این کار در مدیریت مخاطرات و تضمین کیفیت اهمیت دارد.

متدولوژی BON: در BON92 صحبتی در مورد حضور فعالانه کاربران وجود ندارد. در BON95 در برخی فازها برای تایید فاز، نیاز به تاییدیه کاربران نهایی وجود دارد که می توان آن را نوعی حضور فعالانه کاربران در فرآیند دانست.

متدولوژی OPM: تست های پذیرش سیستم را می توان حضور مشارکت فعالانه کاربران در ایجاد سیستم دانست.

مقایسه: با توجه به اینکه OPM صراحتا این بخش را دارد و BON هم اصلا فاز پیاده سازی ندارد که بتوان به گونه ای کاربران را دخالت داد پس OPM بهتر است.

معیار قابل اجرا بودن و کارا بودن:

باید بتوان فرآیند را به راحتی اجرا کرد و در ایجاد نرم افزار به کار گرفت همچنین باید بتوان از فرآیند به طور کارا در حوزه کاربرد استفاده کرد.

قابل اجرا بودن:

متدولوژی BON: این متدولوژی در مجموع قابل اجرا است. مدل های ساخته شده ساده و قابل فهم هستند و چون هدف مشخصی دارند می تواند به راحتی اجرا شوند. البته چون برخی مدل ها کل سیستم را پوشش می دهند در سیستم های بسیار بزرگ ممکن است باعث پیچیده شدن مدل ها و در نتیجه پایین آمدن قابلیت اجرا باشد.

متدولوژی OPM: این متدولوژی با توجه به اینکه تنها یک مدل دارد و مدل هم بسیار پیچیده است اجرای ان برای پروژه های بزرگ بعضا با مشکل همراه خواهد بود. البته ابزار هایی که متدولوژی برای استخراج مدل های بیشتر UML دارد می تواند به قابلیت اجرای آن کمک نماید.

مقایسه: در مجموع با توجه به ویژگی های هر دو و کمتر پیچیده بودن BON قابلیت اجرای آن بیشتر است.

کارا بودن:

متدولوژی BON: این متدولوژی در مجموع، قابل اجرا به صورت کارا، در حد خوبی است البته مشکل پیچیده شدن مدل هایی که کل سیستم را نشان می دهند می تواند کارایی اجرا را کاهش دهد. یکی دیگر از موارد کاهش دهنده کارایی اجرا وابسته بودن مدل ها به ابزار خاص CASE در این متدولوژی می باشد که برای مدل سازی حتما باید از ابزار هایی که خود متدولوژی مشخص کرده استفاده شود. البته ابزارهای خاص

متدولوژی با قابلیت هایی که دارند امکان مدیریت موثر مدل ها و رفع ناسازگاری را هم می دهند که می تواند باعث افزایش کارایی اجرا گردد. در مورد استراتژی مدیریت فرآیند هم متدولوژی دست را برای ایجاد مدل ها باز گذاشته و ترتیب ندارد البته برای انجام برخی فعالیت ها حتما باید قبل از تمام مدل های مورد نیاز ساخته شده باشند.

متدولوژی OPM: داشتن یک مدل کمک می کند ناسازگاری در اجرا نداشته باشیم اما پیچیدگی مدل ها فهم آنها را دشوار می کند. برای پروژه های بزرگ امکان اجرای کارای متدولوژی خیلی بالا نیست.

مقایسه: ب توجه به مشخصات هر دو متدولوژی BON را به دلیل پیچیدگی های کمتر و داشتن ابزار های کنترل ناسازگاری می توان به صورت کارا تر اجرا نمود.

معیار قابل مدیریت بودن پیچیدگی:

فرآیند از تعدادی واحد کاری، نقش، محصول و تعدادی معیار سنجش تشکیل می شود. باید خود این مجموعه از نظر پیچیدگی قابل مدیریت باشد. مدیریت معمولا به دو روش بخش بندی و لایه بندی انجام می شود.

متدولوژی BON: در این متدولوژی به دلیل توصیف دقیق فرآیند مشخص است که چه کاری در چه زمانی باید انجام و چه مدلی در کجا باید ساخته شود. کلاس بندی و خوشه بندی روش دیگری است که به مدیریت پیچیدگی ها کمک می کند. همچنین با کمک ابزار های خاص و با استفاده از ایندکس ها می توان کامپوننت‌ها را به طور دقیق مدیریت نمود. لایه های مختلف انتزاع در کلاس ها هم می تواند به مدیریت پیچیدگی کمک کند.

متدولوژی OPM: خود متدلوژی فرآیند هایی را برای مدیریت پیچیدگی ها پیشنهاد کرده است. مثلا استفاده از ساختار سلسله مراتبی برای مدل های OPD. یا اینکه مدل ها نباید در بیشتر از یک صفحه جا بشوند، هر مدل باید حداکثر ۲۰ تا ۲۵ موجودیت داشته باشند، چیزها (اشیا یا فرآیند ها) در مدل ها نباید همپوشانی داشته باشند، نباید ارتباطات زیادی در نمودار مدل باشد، ارتباطات باید از فضای های تحت اختیار چیزها عبور کند و ارتباطات متقاطع از روی هم باید کمینه بشوند که این موارد می تواند از پیچیده شدن تا حدودی جلوگیری کند.

مقایسه: با توجه به رویکردی که OPM پیشنهاد می کند به نظر می رسد به طور بهتری می تواند مدیریت پیچیدگی را انجام داد.

معیار قابلیت گسترش، مقیاس پذیری، پیکربندی و انعطاف پذیری:

با بتوان فرآیند یک متدولوژی را با کمک نقاط گسترش و مکانیزم های تعریف شده ان گسترش داد، در پروژه های با انداز مختلف اعمال کرد، در ابتدای پروژه و مناسب موقعیت فعلی ان را پیکربندی نمود و بتوان در جریان اجرای فرآیند پیکربندی را تغییر داد و اصلاح کرد.

گسترش پذیری:

متدولوژی BON: قابلیت گسترش برای این متدولوژی وجود ندارد و نقاط گسترش و مکانیزم های گسترش برای ان تعریف نشده اند.

متدولوژی OPM: قابلیت گسترش برای این متدولوژی وجود ندارد و نقاط گسترش و مکانیزم های گسترش برای ان تعریف نشده اند.

مقایسه: هیچ کدام از دو متدولوژی بر دیگری در این زمینه برتری ندارند.

مقیاس پذیری:

متدولوژی BON: فرآیند های انتزاع و تعمیم که در طول فرآیند انجام می شود می تواند باعث افزایش مقیاس پذیری و اجرای متدولوژی روی پروژه با سایز های بزرگ باشد.

متدولوژی OPM: به دلیل اینکه کم عمق بودن و جزئیات کم و پیچیده بودن نمی توان به خوبی برای پروژه های بزرگ از این متدولوژی استفاده کرد. البته با ابزاری که متدلوژی پیشنهاد می کند برای استخراج نمودار های UML از مدل OPD می توان کمی این موضوع را تعدیل کرد.

مقایسه: در مجموع با مقایسه ویژگی های دو متدلوژی BON قابلیت مقیاس پذیری بیشتری دارد.

پیکربندی:

متدولوژی BON: این متدولوژی قابلیت پیکربندی فرآیند در ابتدای پروژه را ندارد.

متدولوژی OPM: این متدولوژی قابلیت پیکربندی فرآیند در ابتدای پروژه را ندارد.

مقایسه: در این زمینه هیچ کدام از دو متدولوژی بر دیگری برتری ندارد.

انعطاف پذیری:

متدولوژی BON: نمی توان در طول اجرای فرآیند آن را تغییر داد تنها می توان با رعایت برخی محدودیت ها ترتیب ساخت مدل ها را به دلخواه انتخاب نمود.

متدولوژی OPM: امکان تغییر پیکربندی فرآیند در میانه پروژه وجود ندارد.

مقایسه: در این زمینه هیچ کدام از دو متدولوژی بر دیگری برتری ندارد.

معیار حوزه کاربرد:

متدولوژی در چه حوزه هایی کاربرد دارد و برای چه حوزه هایی مناسب نیست حداقل باید بتواند حوزه کاربرد سیستم های اطلاعاتی را پوشش دهد.

متدولوژی BON: این متدولوژی به دلیل ذات شی گرا قابلیت اجرا روی انواع مختلف پروژه ها را دارد. در BON92 سیستم اجاره خودرو و در BON95 سیستم مدیریت کنفرانس و سیستم ضبط ویدیو به عنوان نمونه های موردی آن مطرح شده اند.

متدولوژی OPM: این متدلوژی به دلیل داشتن شی گرایی همراه با خود به صورت بالقوه می تواند روی انواع مختلف پروژه پیاده سازی شود اما به دلیل عمق کم و نداشتن جزئیات کافی در عمل ممکن است روی پروژه های بزرگ جواب ندهد. سیستم ATM به عنوان نمونه موردی در خود متدولوژی مطرح شده است.

مقایسه: با توجه به ویژگی هایی که هر کدام دارند به نظر می رسد BON حوزه کاربرد بیشتری را پوشش دهد البته باید به این نکته توجه نمود که متدولوژی BON تنها فاز های تحلیل و طراحی را پوشش می دهد.

معیارهای زبان مدل سازی:

معیار پشتیبانی از مدل سازی شی گرا سازگار، دقیق و بدون ابهام:

مدل ها باید با کیفیت و شامل هر چه در یک مدل شی گرا نیاز است باشد و بتوان به وضوح شی گرایی را در آن دید، مدل ها نباید یکدیگر را نقض کنند، مدل باید صحیح باشند و اشتباه نداشته باشند همچنین باید با جزئیات کامل باشد و زبان مدل سازی اینها را پشتیبانی کند.

دیدگاه های مختلف مدل سازی:

متدولوژی BON: این متدولوژی فقط دیدهای رفتاری و ساختاری مدل سازی را دارد و مدلی برای دید وظیفه ای ندارد.

متدولوژی OPM: از نظر مدل سازی رفتاری متدولوژی دارای ضعف است.

مقایسه: BON به دلیل پوشش جنبه رفتاری که باعث دقیق تر شدن متدولوژی می شود و می توان ترتیب کارها را هم مشاهده نمود به دلیل کمتر مبهم بودن، نسبت به OPM برتر ی دارد.

مدل سازی از منطقی به فیزیکی:

متدولوژی BON: این متدولوژی با اضافه کردن تصمیمات مربوط به پیاده سازی به مدل های منطقی می تواند دید فیزیکی قابل پیاده سازی را بسازد. در این متدولوژی فرمالیسم موجود در آن به تحلیل گر کمک می کند تا دید منطقی را به راحتی به دید فیزیکی قابل پیاده سازی تبدیل کند.

متدولوژی OPM: این متدولوژی چون یک مدل دارد بعد از پایان تحلیل با اضافه کردن جزئیات پیاده سازی به مدل می تواند به راحتی وارد طراحی شود از دید منطقی به فیزیکی برود.

مقایسه: مدل سازی از منطقی به فیزیکی به دلیل داشتن تنها یک مدل در OPM بهتر و ساده تر از BON است و ناسازگاری و زحمت کمتری برای بروزرسانی دارد.

سطوح مختلف انتزاع و دانه بندی:

متدولوژی BON: این متدولوژی مدل سازی را در سطح سیستم شروع و تا سطح کلاس ها ادامه می دهد.

متدولوژی OPM: در بالاترین سطح تا پایین ترین سطح در سیستم به عنوان کلاس نمایش داده می شود. زیر سیستم ها به عنوان کلاس مدل سازی می شوند و تا پایین ترین سطح آن نیز به صورت کلاس مدل می شوند.

مقایسه: دو متدلوژی در این زمینه برتری نسبت به یکدیگر ندارند.

مدل سازی صوری و غیر صوری:

متدولوژی BON: این متدولوژی دارای زبان مدل سازی بصری و متنی خاص خود می باشد و ابزاری هم برای مدل سازی در اختیار قرار می دهد. همچنین در مدل سازی متنی گرامر خاص که برای انجام کارهای خودکار مناسب است را هم دارد که در BON95 معرفی شده است.

متدولوژی OPM: این متدولوژی دارای زیان مدل سازی بصر و متنی خاص خود است که می توان از زبان مدل سازی متنی برای مبنا قراردادن آن برای تست و ساخت کد استفاده نمود.

مقایسه: هر دو متدولوژی از لحاظ زبان مدل سازی هر دو نوع متنی و بصری را دارند ولی هیچ کدام فرمالیسم ندارند و چیزی که بتوان مدل ها را با زبان منطق ریاضی توصیف کند ندارند و در این معیار تقریبا با هم مشابه هستند و هیچ کدام برتری محسوسی بر دیگری ندارد.

معیار مدیریت تناقض ها و پیچیدگی ها:

پیچیدگی مدل ها باید مدیریت شود و از بوجود آمدن ناسازگاری بین آنها جلوگیری نمود. زبان مدل سازی باید برای تشخیص و رفع ناسازگاری و پیچیدگی مدل ها استراتژی و تکنیک های مناسب داشته باشد.

متدولوژی BON: این متدولوژی ابزارهایی دارد که با کمک ایندکس ها می تواند کامپوننت ها را مدیریت و ناسازگاری های آنها را تشخیص دهد و رفع کند. همچنین روابط دقیق بین مدل ها به گونه ای تعریف شده که می توان پیچیدگی را مدیریت و ناسازگاری را کاهش داد. روش های خوشه بندی و کلاس بندی هم کمک به مدیریت پیچیدگی ها می نماید.

متدولوژی OPM: به دلیل اینکه یک مدل دارد تقریبا در آن ناسازگاری پیش نمی آید اما مدل می تواند بسیار پیچیده شود که متدلوژی روش هایی را برای رفع پیچیدگی ها پیشنهاد داده است. مثلا استفاده از ساختار سلسله مراتبی برای مدل های OPD. یا اینکه مدل ها نباید در بیشتر از یک صفحه جا بشوند، هر مدل باید حداکثر ۲۰ تا ۲۵ موجودیت داشته باشند، چیزها (اشیا یا فرآیند ها) در مدل ها نباید همپوشانی داشته باشند، نباید ارتباطات زیادی در نمودار مدل باشد، ارتباطات باید از فضای های تحت اختیار چیزها عبور کند و ارتباطات متقاطع از روی هم باید کمینه بشوند که این موارد می تواند از پیچیده شدن تا حدودی جلوگیری کند.

مقایسه: مدیریت پیچیدگی ها در BON بهتر به نظر می رسد چون OPM با توجه به ذات آن که یک مدل دارد حتی در صورت استفاده از روش هایی برای مدیریت پیچیدگی مدل کماکان مدل ها پیچیده می مانند و تنها پیچیدگی مدل ها کم می شود.