مروری بر توسعه نرم افزار مبتنی بر مولفه

مهرناز کلاته [۱] ، علی اکبر صدری [۲*۳]

۱-گروه مهندسی کامپیوتر،دانشکده برق و کامپیوتر،دانشگاه آل طه، تهران، ایران

۲-گروه مهندسی کامپیوتر،واحد علوم و تحقیقات، دانشگاه آزاد اسلامی،تهران،ایران

۳-گروه مهندسی کامپیوتر،دانشکده فنی و مهندسی،دانشگاه امام علی (ع)،تهران،ایران

چکیده

هدف نهایی توسعه نرم افزار ساخت محصولات با کیفیت بالا است. مشتریان صنعت نرم افزار همیشه خواهان محصولات باکیفیت سریع و مقرون به صرفه هستند. توسعه مبتنی بر مؤلفه (CBD)[1] مناسب‌ترین روش برای شرکت‌های نرم‌افزاری برای برآوردن خواسته‌های بازار هدف است. برای انتخاب CBD، تیم‌های توسعه نرم‌افزار باید اجزای عمومی موجود در بازار را سفارشی کنند و انتخاب اجزای مناسب از بین میلیون‌ها مؤلفه شخص ثالث و تجاری برای تیم‌های توسعه بسیار دشوار است. از سوی دیگر، توسعه انبار داخلی خسته کننده و زمان بر است. به دلیل استفاده گسترده از قطعات، CBSE[2] کاملاً با روش سنتی آبشار متفاوت است. CBSE نه تنها مستلزم تمرکز بر مشخصات و توسعه سیستم است، بلکه نیاز به بررسی بیشتر برای زمینه کلی سیستم، خصوصیات اجزای جداگانه و فرآیند اکتساب و یکپارچه سازی اجزا دارد. اصطلاح توسعه نرم افزار مبتنی بر مؤلفه را می‌توان به عنوان فرآیند ساخت یک سیستم با استفاده از مؤلفه‌ها نام برد. چرخه حیات CBD شامل مجموعه‌ای از مراحل است، یعنی شناسایی و انتخاب اجزا بر اساس نیاز، یکپارچه‌سازی و مونتاژ اجزای انتخاب‌شده و به‌روزرسانی سیستم به عنوان اجزای سازنده در طول زمان با نسخه‌های جدیدتر. در این مقاله بررسی ادبیات شاخصی از تکنیک‌های پیشنهادی برای مراحل مختلف چرخه زندگی CBD ارائه می‌کند. هدف کمک به درک بهتر تکنیک‌های مختلف CBD است.

کلمات کلیدی: CBSE، CBD، ارزیابی کارایی، طبقه بندی

[1] Component-Based software development
[2] Component-Based Software Engineering

۱- مقدمه

در دهه گذشته، استفاده گسترده از نرم افزار، تقاضاها و انتظارات جدیدی را در صنعت نرم افزار به ویژه برای بهبود کیفیت و افزایش بهره وری توسعه ایجاد کرده است. برای رویارویی با این چالش ها، توسعه نرم افزار باید بتواند با پیچیدگی ها کنار بیاید و به سرعت با تغییرات در نتیجه افزایش تقاضا برای ادغام مناطق مختلف سازگار شود. مهندسان نرم‌افزار بر این عقیده بوده‌اند که نرم‌افزار نباید همیشه از ابتدا توسعه داده شود و به مونتاژ اجزای قبلی موجود در سیستم‌های نرم‌افزاری بزرگ فکر کرده‌اند. مهندسان نرم افزار از روزهای اولیه محاسبات از انتزاعات و فرآیندها مجددا استفاده کرده اند، اما رویکردهای اولیه برای استفاده مجدد موردی بود. از آنجایی که نیاز روزافزون به تکنیک هایی وجود دارد که بتواند فرآیند توسعه نرم افزار را بهبود بخشد، زمان عرضه به بازار را کاهش دهد و کیفیت محصولات نرم افزاری ارائه شده را بهبود بخشد، رویکرد سازماندهی شده و متمرکزتری برای استفاده مجدد به نام توسعه نرم افزار مبتنی بر مؤلفه وجود دارد[۱].

در زمینه CBD ، یک جزء، بلوک اساسی برای یک سیستم نرم افزاری است. مفهوم جزء نرم افزار به تفصیل مورد بحث قرار گرفته و تعاریف متعددی پدید آمده است. از آنجایی که مؤلفه‌ها در ساختن سیستم‌های نرم‌افزاری مبتنی بر مؤلفه اساسی هستند، توافق بر سر یک تعریف مهم است. تعریف مؤلفه «یک جزء نرم افزاری واحدی از ترکیب با رابط های قراردادی مشخص شده و فقط وابستگی های زمینه صریح است. یک جزء نرم افزاری می تواند به طور مستقل مستقر شود و توسط اشخاص ثالث ترکیب شود.این تعریف این واقعیت را تأیید می‌کند که یک جزء یک بسته نرم افزاری مستقل است که عملکرد را از طریق رابط کاملاً تعریف شده ارائه می‌دهد. تحقیقات به مسائل بسیاری از مفاهیم و تعاریف اساسی گرفته تا تکنیک‌ها و فرآیند توسعه مبتنی بر مؤلفه می پردازد[۲].

هدف هر سازمان نرم افزاری تولید یا دستیابی به نرم افزار کیفی، صرفه جویی در زمان، هزینه و پاسخگویی به تقاضای بازار است. در طول مرحله تجزیه و تحلیل، اجزای مناسب هم از مخزن داخلی و هم از مخزن شخص ثالث جستجو می‌شوند. زمان زیادی صرف شناسایی، انتخاب و تجزیه و تحلیل می‌شود که باعث افزایش هزینه سفارشی‌سازی و یکپارچه‌سازی می‌شود، زیرا ساختار مؤلفه هیچ بینشی برای کمک به تیم با برخی اطلاعات اولیه از جمله اجزای کیفیت یا ویژگی ارائه نمی‌دهد و تیم توسعه هیچ جهتی ندارد. زمان زیادی نیز در طول سفارشی سازی و ادغام اجزای قابل استفاده مجدد صرف می‌شود. این احتمال وجود دارد که اجزای مناسب تری شناسایی نشوند که منجر به اتلاف زمان، هزینه، تلاش و منابع می‌شود. نیاز به ارائه ساختاری برای ارزیابی ویژگی‌های کیفی قطعات تجاری خارج از قفسه (COTS)[3] یا در اجزای خانه در طول مرحله انتخاب وجود دارد. همچنین نیاز به نظارت بر سیستم کنترل نسخه، بازخورد کاربر نهایی، مسائل مربوط به کیفیت خدمات و

[3] commercial off the shelf

بلوغ فروشنده مؤلفه وجود دارد [۳]. در فرآیند مهندسی نرم افزار سنتی، فعالیت‌های معمولی درگیر در ساخت یک سیستم عبارتند از: مشخصات، پیاده سازی، آزمایش و نگهداری می‌باشند. با این حال، به دلیل استفاده گسترده از اجزای موجود، فعالیت های CBD تا حدودی متفاوت است.

CBD نه تنها مستلزم تمرکز بر مشخصات و توسعه سیستم است، بلکه نیاز به بررسی بیشتر برای زمینه کلی سیستم، خصوصیات اجزای جداگانه و فرآیند اکتساب و یکپارچه سازی اجزا دارد. بنابراین، تعدادی از چالش‌ها را برای تکنیک‌هایی که باید از آن پشتیبانی کنند، به همراه آورده است. جامعه تحقیقاتی نرم افزار این چالش‌ها را با پرداختن به مسائلی از تعریف دقیق مفاهیم اصلی، ارزیابی و ادغام اجزا تا توسعه روش‌ها و فرآیندهای توسعه اجزای نرم افزار درک کرده است[۴]. ساختار مقاله به شرح زیر است. در بخش دوم به بررسی کارهای گذشته، در بخش سوم یک نمای کلی از مزایای بالقوه، خطرات و چالش‌های مرتبط با CBD ارائه می‌دهیم. در بخش چهارم بررسی و بحث در مورد شناسایی اجزا و تکنیک‌های انتخاب را ارائه می دهیم. در بخش پنجم به بررسی تکنیک‌های پرداختن به مرحله یکپارچه سازی اجزای CBD را ارائه می‌دهیم. تکنیک‌های استقرار و تکامل CBD به ترتیب در بخش‌های ششم و هفتم مورد بحث قرار گرفته‌اند. و در نهایت، نتیجه گیری در بخش هشتم ارائه شده است.

۲- کارهای مرتبط

ایده مهندسی نرم افزار مبتنی بر مؤلفه در میان جامعه نرم افزاری جدید نیست و شرکت‌های توسعه نرم افزار بیش از دو دهه است که آن را تمرین می‌کنند .ارزیابی کیفیت و معیارها لازم است برای هر جزء جداگانه قبل از استفاده مجدد از آن بدانیم. با این حال، معیارهای جفت مولفه رسمی و غیرمستقیم برای اندازه گیری کیفیت اجزا با توجه به پیچیدگی و عملکرد پیشنهاد شده است.

در مقاله [۵] به بررسی فعالیت های مدیریت ریسک و همبستگی آن‌ها با وقوع خطرات معمولی در توسعه سیستم‌های هزینه می‌پردازد. این امر با کاوش در وقوع خطرات معمولی در سیستم‌های هزینه به دست می‌آید و سپس اثربخشی فعالیت‌های کاهش ریسک مقایسه می‌شود.

در مقاله [۶] هدف تحقیق پیشنهادی بهبود نگرانی‌های امنیتی و کیفیت سیستم‌های نرم‌افزار منبع باز است. که به بررسی وضعیت فعلی ابزارسازی در حوزه مهندسی معکوس می‌پردازد که قصد دارد این عمل را به یک قالب قابل پیش‌بینی بهبود بخشد.

در مقاله [۷] به مطالعه چالش‌های DevOps از جمله عملکرد و کیفیت را بررسی می‌کند. یک مدل تضمین کیفیت برای پیاده سازی در مراحل تجزیه و تحلیل، توسعه، صدور گواهینامه، سفارشی سازی، طراحی، یکپارچه سازی، آزمایش و نگهداری پیشنهاد شده است.

جدول ۱: مطالعه سایر مقالات

عنوان مقاله	محدودیت
مروری بر معیارهای جفت مولفه برای توسعه مبتنی بر مولفه [۸].	این ایده برای اندازه گیری اثربخشی اجزای قابل استفاده مجدد پیشنهاد شده است.
بررسی وضعیت مدیریت ریسک در توسعه با اجزای نرم افزاری [۹].	نتایج نشان می‌دهد که هنوز عوامل مرتبط با ریسک غیرقابل توضیح در فعالیت‌های کاهش ریسک پیشنهادی وجود دارد.
مدلی برای تضمین کیفیت معماری OSS [۱۰]	استفاده از مدل تضمین کیفیت در مهندسی نرم افزار بسیار سخت است.
ساخت ابزارهای مهندسی معکوس با اجزای نرم افزار[۱۱]	مطالعات موردی بیشتری برای نتیجه گیری نتایج مورد نیاز است.
به سمت یک مدل پیش‌بینی اشکال[۱۲].	فقط از مجموعه داده‌های آپاچی برای اعتبارسنجی مدل استفاده می‌شود.
انتخاب مبدا مؤلفه برای سیستم‌های فشرده نرم‌افزار [۱۳].	نیاز به انجام مطالعات تجربی برای تعمیم نتایج وجود دارد

۳- مزایا، خطرات و چالش‌ها

هدف CBD ساختن سیستم‌ها به عنوان مجموعه‌ای از اجزاء است به گونه‌ای که توسعه اجزا به عنوان موجودیت‌های قابل استفاده مجدد، نگهداری سیستم با سفارشی سازی و جایگزینی چنین اجزایی می‌باشد. انگیزه استفاده از CBD کاهش هزینه توسعه، زمان عرضه به بازار و ارائه سیستمی است که در برآوردن تقاضاهای متغیر مشتری کارآمد باشد. برخی از مزایای بالقوه CBD، در جدول ۲ خلاصه شده است. اگرچه CBD بسیار امیدوارکننده است، اما CBD یک رشته جدید است و تعدادی از خطرات و چالش‌های مرتبط با آن وجود دارد. یکی از خطرات بالقوه برای CBD این است که اگر تامین‌کننده

اصلی جزء از کار بیفتد یا پشتیبانی از نسخه فعلی قطعه را متوقف کند، چه اتفاقی می‌افتد. از سوی دیگر، اگر سیستم نیاز به کیفیت بالایی دارد، چگونه می‌توانیم اطمینان حاصل کنیم که قطعه امکان برآورده کردن این نیازها را فراهم می‌کند[۱۴].

جدول ۲: خلاصه مزایای توسعه نرم افزار مبتنی بر مؤلفه

شرح	مزیت
زمان کمتری برای خرید یک جزء نسبت به طراحی، کدگذاری، آزمایش، اشکال زدایی و مستندسازی آن نیاز است.	کاهش زمان توسعه
سیستم‌های مبتنی بر مؤلفه از اجرای مؤلفه‌ها مصون هستند، بنابراین انتخاب بیشتری از مؤلفه‌ها وجود دارد که می‌توان از بین آن‌ها انتخاب کرد تا الزامات را برآورده کنند.	افزایش انعطاف پذیری
اگر یک جزء وجود داشته باشد، در مقایسه با توسعه جدید، عدم اطمینان کمتری در هزینه مربوط به استفاده مجدد از آن وجود دارد.	کاهش ریسک فرآیند
کامپوننت ها در بسیاری از کاربردهای مختلف مورد استفاده مجدد و آزمایش قرار می‌گیرند. ایرادات طراحی و اجرا در استفاده اولیه کشف و برطرف می‌شوند و در نتیجه کیفیت قطعه افزایش می یابد.	کیفیت ارتقا یافته
جایگزینی آسان قطعات منسوخ شده با قطعات جدید پیشرفته	تعمیر و نگهداری کم
برخی از استانداردها را می توان به عنوان مجموعه ای از توسعه اجزای استاندارد پیاده سازی کرد. استفاده از این استانداردها کیفیت کلی قطعات و سیستم ها را بهبود می بخشد.	استاندارد سازی

خطرات بالقوه CBD، در جدول ۳ خلاصه شده است. با این حال، مزایای CBD واقعی است و این نگرانی‌ها نیاز به آماده سازی دقیق و برنامه ریزی برای رسیدگی به این چالش‌ها را با تعریف دستورالعمل‌ها، استانداردها و معماری باز برای CBD افزایش می‌دهد[۱۵]

جدول ۳: خلاصه خطرات توسعه نرم افزار مبتنی بر مؤلفه

شرح	ریسک و چالش ها

انتخاب مولفه یک فرآیند تکراری است و نتیجه به مکانیسم طبقه بندی و بازیابی بستگی دارد. جستجوی مؤلفه در طیف گسترده ای از مخازن مؤلفه انجام می‌شود، حتی زمانی که مؤلفه‌ها یافت می‌شوند، ممکن است عملکرد خاصی را انجام ندهند یا با یکدیگر همکاری نکنند.	ارضای نیاز
مهندسان نرم افزار قبل از اینکه به طور معمول جستجوی کامپوننت را به عنوان بخشی از فرآیند توسعه عادی خود بگنجانند، باید به طور منطقی از یافتن اجزای مناسب از مخازن مطمئن باشند.	یافتن اجزای مناسب
CBD یک چالش قابل توجه برای اطمینان از اینکه خدمات جزء از طریق رابط‌های استاندارد برای اطمینان از قابلیت همکاری ارائه می شود، ایجاد می کند.	قابلیت همکاری
هر جزء باید مورد تایید و تست اعتبار سنجی قرار گیرد. با این حال، بر خلاف برنامه‌های کاربردی سنتی، اجزای جداگانه را می‌توان در مجموعه‌ای از برنامه‌های کاربردی استفاده کرد که فرآیند تست را پیچیده می کند.	تست واحد و ادغام

۱.۳- فرایند توسعه

به دلیل استفاده گسترده از اجزاء، فعالیت‌های درگیر در هر فاز CBD و روابط بین فازهای مختلف از رویکرد سنتی دور است. CBD با تمرکز بر انتخاب و مونتاژ اجزای موجود برای ساختن یک سیستم نرم افزاری، یکپارچه محور است. برای حمایت از هر مرحله از چرخه زندگی CBD را بر اساس دستورالعمل‌های زیر که از [۱۵] اتخاذ شده است، بررسی می کنیم.

- ت

عریف سوال قابل پاسخ:

هدف ما این است که با تجزیه و تحلیل تکنیک‌های پیشنهادی برای هر یک از مراحل آن، درک بهتری از چرخه زندگی CBD به دست آوریم.

- ی

افتن بهترین شواهد

مقالات را از مجلات اصلی مهندسی نرم افزار و کنفرانس های بین المللی بررسی می کنیم.

- ش

واهد را تجزیه و تحلیل کنید.

تحلیلی از تکنیک‌های پیشنهادی ارائه می‌کنیم تا مزایا و محدودیت‌های آن‌ها را برجسته کنیم.

٤- شناسایی و انتخاب جزء

شناسایی و انتخاب اجزا به طور گسترده‌ای به عنوان یک فرآیند مرتبط به هم شناخته شده است که نقش اصلی را در CBD کلی ایفا می‌کند. موفقیت CBD به توانایی شناسایی و انتخاب اجزای مناسب بستگی دارد. انتخاب نامناسب مولفه می‌تواند منجر به اثرات نامطلوب شود، مانند اجزای فهرست کوتاه که به سختی عملکرد مورد نیاز را برآورده می کنند یا هزینه های اضافی را در مراحل یکپارچه سازی و نگهداری ایجاد می‌کنند. در CBD، شناسایی و انتخاب جزء را می‌توان به عنوان فرآیند چهار مرحله‌ای، صفحه جستجو، مراحل ارزیابی و تجزیه و تحلیل طبقه‌بندی کرد. شناسایی مولفه با جستجوی اجزای بالقوه‌ای شروع می‌شود که با نیازهای ذینفعان مطابقت دارند، که ممکن است خود به طور همزمان در حال تکامل باشند. مرحله جستجو معمولاً با پرداختن به اطلاعات اجزای ساختار نیافته که در قالب‌های مختلفی از بروشورهای بازاریابی گرفته تا توضیحات زبان طبیعی موجود است، انجام می‌شود. بنابراین، طبقه‌بندی مؤلفه‌ها یک ویژگی مهم در فرآیند کلی انتخاب مؤلفه ایفا می‌کند. مرحله جستجو منجر به یک لیست کلی از اجزای نامزد می‌شود که پتانسیل لازم برای برآوردن نیازهای ذینفعان را دارند[۱۶].

از آنجایی که نتایج مرحله جستجو برای انتخاب مولفه نهایی بیش از حد عمومی است، از مراحل ارزیابی و تجزیه و تحلیل برای مقایسه و رتبه‌بندی مؤلفه‌های نامزد بر اساس معیارهای ارزیابی سیستماتیک استفاده می‌شود. این بخش یک بررسی شاخص از تکنیک‌هایی ارائه می‌کند که از شناسایی و انتخاب مؤلفه در چرخه حیات CBD پشتیبانی می‌کنند. ابتدا، تکنیک‌های طبقه‌بندی مؤلفه‌ها را مورد بحث قرار می‌دهیم زیرا آنها رابطه مستقیمی با موفقیت جستجوی مؤلفه‌های کاندید دارند. بعداً، مدل‌های شناسایی و انتخاب مؤلفه‌ها را مورد بحث قرار می‌دهیم و آنها را بر اساس مدل‌های زیربنایی آنها دسته‌بندی می‌کنیم.

ابتدا، رویکردهایی را مورد بحث قرار می‌دهیم که مجموعه‌ای از مراحل سلسله مراتبی را برای شناسایی و انتخاب اجزای مناسب پیشنهاد می‌کنند. دسته رویکرد خوشه‌بندی بحثی را در مورد رویکردهایی ارائه می‌کند که از مجموعه‌ای از قوانین از پیش تعریف‌شده و اکتشافی برای شناسایی و انتخاب اجزا استفاده می‌کنند. رویکردهای دسته‌بندی تکراری از فرآیند تکراری مقایسه مؤلفه‌ها با الزامات ذینفعان استفاده می‌کنند و از تکنیک‌های تصمیم‌گیری چند معیاره برای انتخاب مؤلفه‌های مناسب استفاده می‌کنند. در نهایت، رویکردهایی که از مفهوم پایگاه دانش استفاده می‌کنند، مورد بحث قرار می‌گیرند[۱۷].

۱.٤- طبقه بندی

طبقه بندی مولفه‌ها یکی از ویژگی‌های کلیدی است که تأثیر مستقیمی بر اثربخشی و کارایی CBD دارد. هدف طرح‌های طبقه‌بندی، فراهم کردن وسیله‌ای برای مقایسه شباهت‌های اجزا برای انتخاب و بازیابی بهتر اجزا است. رویکردهای طبقه بندی موجود را می توان به چهار نوع جستجوی ساده کلمه کلیدی، طبقه بندی وجهی، تطبیق امضا و تطبیق رفتار دسته بندی کرد. جستجوی ساده کلمه کلیدی از مجموعه ای از کلمات کلیدی برای مقایسه عملکرد مورد نیاز با ویژگی های جزء استفاده می‌کند. رویکردهای طبقه‌بندی روبرو تلاش می‌کند تا اجزا را بر اساس طبقه‌بندی‌های از پیش تعریف‌شده طبقه‌بندی کند. در رویکردهای تطبیق امضا، مؤلفه‌ها با امضای آن ها نشان داده می‌شوند و سلسله مراتبی از معیارهای تطبیق تعریف می شوند در نهایت، رویکردهای تطبیق رفتار از مشخصات رسمی برای توصیف رفتار اجزای نرم‌افزار و تعیین اینکه آیا دو جزء مطابقت دارند استفاده می‌کنند[۱۸].

در مقاله [۱۹]رویکردهای طبقه بندی مؤلفه‌ها را در سه گروه اصلی دسته بندی کرده اند. اول، رویکردهایی که از طبقه بندی و هستی شناسی برای ارائه توضیحات اجزاء استفاده می کنند و از اطلاعات معنایی برای طبقه بندی اجزا استفاده می کنند. دوم، رویکردهایی که از مفهوم مرحله یادگیری در طی شناسایی مؤلفه برای طبقه‌بندی و پشتیبانی از فرآیند انتخاب مؤلفه استفاده می‌کنند و در نهایت، رویکردهایی که مفهومی از اندازه‌گیری فاصله معنایی بین عملکرد مورد نیاز و ارائه شده را پیشنهاد می‌کنند.

در مقاله [۲۰] یک طرح طبقه بندی برای مشخص کردن اجزا بر اساس طرح طبقه بندی و کدگذاری[4]پیشنهاد کرد که یک کد منحصر به فرد به هر جزء اختصاص می‌دهد. طبقه بندی اجزا می تواند بر اساس عواملی مانند گروه فناوری ها، نوع کاربردها و غیره باشد. کد منحصر به فرد اختصاص داده شده بعداً برای انتخاب مؤلفه‌های مناسب از مخزن دانش مبتنی بر مؤلفه استفاده می شود. مکانیسم رتبه مؤلفه بر اساس تجزیه و تحلیل روابط استفاده واقعی مولفه ها و انتشار اهمیت از طریق روابط استفاده است. در این مدل، مجموعه‌ای از اجزای نرم‌افزار به‌عنوان یک گراف جهت‌دار وزنی نشان داده می‌شود که گره‌های آن با اجزا و یال‌ها با رابطه استفاده مطابقت دارند. گره ها در گراف با وزن هایشان که به عنوان عناصر بردار ویژه یک ماتریس مجاور برای گراف جهت دار تعریف می شوند، رتبه بندی می شوند. این تکنیک ها به افزایش انعطاف پذیری توسعه کمک می کند. با این حال، آنها فاقد یک فرآیند مدل‌سازی نیازمندی‌های تعریف‌شده هستند و ارتباط خوبی با انتخاب و بازیابی اجزا ندارند.

در مقاله [۲۱] یک طرح طبقه بندی آگاه از سلسله مراتب را برای کد شی گرا ارائه می‌کند، که در آن اجزای نرم افزار بر اساس ویژگی های رفتاری خود، مانند سرویس ها، الگوریتم‌ها و داده‌ها طبقه بندی می‌شوند. این ویژگی‌ها از توصیف

[4] C&C

کلاس‌های انتزاعی که ساختار چارچوب و هدف را مشخص می‌کنند، ساخته می‌شوند. مجموعه ای از ویژگی‌های مرتبط با یک جزء معین، توصیفگر نرم افزار آن را تشکیل می‌دهد این پایه توصیفگر برای پرس و جو از سیستم، مشخص کردن مجموعه ای از عملکردهای مورد نظر و دریافت یک مجموعه رتبه بندی شده از نامزدهای سازگار استفاده می‌شود.

در مقاله [۲۲] طرح طبقه‌بندی چندتایی بر اساس ارزش ویژگی، طرح‌های طبقه‌بندی وجهی و شمارشی را پیشنهاد می‌کند. مقدار مشخصه در ابتدا در طبقه بندی برای تعیین دامنه، پلتفرم، سیستم عامل و زبان توسعه مربوط به جزء استفاده می‌شود. سپس عملکرد مؤلفه با استفاده از طرح وجهی تنها با استفاده از مجموعه عملکردی طرح وجهی طبقه بندی می‌شود. سپس از طرح شمارش شده برای طبقه بندی و بازیابی ارائه های مؤلفه استفاده می شود.

۲.٤- رویکرد سلسله مراتبی

(OTSO[5]) فرآیندی است که مستقیماً به موضوع انتخاب مؤلفه می‌پردازد. این فرآیند بر اساس ارزیابی سلسله مراتبی است که الزامات را به مجموعه معیارهای سلسله مراتبی تجزیه می‌کند. این فرآیند شناسایی و انتخاب اجزای سیستماتیک، قابل تکرار و نیازمندی محور را تسهیل می‌کند. OTSO چهار فرآیند فرعی معیارهای جستجو، تعریف خط مبنا، تعریف ارزیابی دقیق و وزن دادن به معیارها را شناسایی می‌کند. اصل اصلی روش OTSO یک فرآیند سیستماتیک کاملاً تعریف شده است که کل فرآیند انتخاب مؤلفه را پوشش می‌دهد. معیارهای ارزیابی جزئی اجزا را از اهداف استفاده مجدد، روشی برای تخمین تلاش نسبی یا مزایای هزینه جایگزین‌های مختلف و روشی برای مقایسه جنبه‌های «غیر عملکردی» جایگزین‌ها به دست می‌آورد. فرآیند تعریف معیارهای ارزیابی اساسا الزامات یک جزء را به مجموعه معیارهای سلسله مراتبی تجزیه می‌کند و هر شاخه در این سلسله مراتب به یک ویژگی ارزیابی ختم می شود. با این حال، جنبه‌های کیفی را در فرآیند انتخاب شامل نمی‌شود[۲۳].

در مقاله [۲۴] مدل توسعه سیستم یکپارچه سازی مبتنی بر COTS (CISD) را پیشنهاد می‌کند. مرحله شناسایی آن شامل فرآیند جمع آوری و درک نیازهای کلی سیستم، شناسایی اجزا در مجموعه محصول و اولویت بندی آنها برای ارزیابی بعدی است. عمده فعالیت‌های این مرحله تجزیه و تحلیل نیاز، شناسایی محصول و اولویت بندی محصول می‌باشد. مرحله تجزیه و تحلیل نیازمندی شامل فرآیند درک نیازمندی‌های سیستم و تقسیم این الزامات به حوزه های مختلف برنامه و خدمات است. مرحله شناسایی محصول شامل فرآیند جمع آوری اطلاعات در مورد محصولات جزء کاندید و گروه بندی این محصولات در ترکیبات یا مجموعه های مختلف محصول برای ارزیابی بیشتر است. مرحله اولویت‌بندی محصول شامل بررسی تمام مجموعه‌های محصول نامزد برای ایجاد یک لیست اولویت‌بندی شده برای ارزیابی بیشتر است.

[5] Off the shelf option

۳.٤-رویکرد خوشه بندی

روش‌شناسی توسعه مؤلفه، زبان مدل‌سازی یکپارچه[6] و فرآیند یکپارچه rational را با معناشناسی مرتبط با توسعه مؤلفه گسترش می‌دهد. این مدل از تکنیک‌های خوشه‌بندی برای شناسایی مؤلفه‌ها بر اساس روش استفاده و خوشه‌بندی کلاس استفاده می‌کند. استفاده از روش خوشه‌بندی موردی، موارد استفاده منسجم را با در نظر گرفتن رابطه بین موارد استفاده و خوشه‌بندی موارد استفاده منسجم و کلاس‌ها با اعمال الگوریتم خوشه‌بندی پیشنهادی، خوشه‌بندی می‌کند. یک ماتریس برای استفاده و کلاس‌ها با تخصیص مقادیر بر اساس رابطه بین استفاده و کلاس تعریف می‌شود. COMO با استفاده از UML به عنوان استانداردی برای مشخصات اجزا، فرآیند CBD را بهبود می بخشد. یک محدودیت این است که فقط بر الزامات عملکردی تمرکز می‌کند و ویژگی‌های کیفیت را در مرحله شناسایی جزء در نظر نمی‌گیرد[۲۵].

در مقاله [۲۶] رویکردی را پیشنهاد می‌کند که به شناسایی و انتخاب مؤلفه‌ها از یک مدل شی که یک حوزه تجاری را نشان می‌دهد کمک می‌کند. این رویکرد از یک الگوریتم خوشه بندی بر اساس مجموعه‌ای از قوانین از پیش تعریف شده و اکتشافی استفاده می‌کند. فرآیند شناسایی مؤلفه‌ها با گروه بندی کلاس‌های مرتبط یک مدل دامنه سطح تحلیل آغاز می‌شود. رویکرد شناسایی مؤلفه از مجموعه ای از اکتشافات (خودکار و دستی) برای اصلاح بیشتر راه حل اولیه به دست آمده از الگوریتم خوشه بندی استفاده می‌کند.

٤.٤- رویکرد تکراری

PORE[7] مبتنی بر فرآیند تعاملی جمع‌آوری نیازمندی‌ها و ارزیابی محصول است. از یک رویکرد مبتنی بر الگو برای اصلاح لیست مؤلفه‌ها استفاده می‌کند تا زمانی که مناسب ترین مؤلفه انتخاب شود.

رویکرد PORE دارای سه جزء اصلی زیر است.

- ید

ک مدل فرآیندی که اهداف اساسی را مشخص می‌کند و فرآیندهای عمومی را برای دستیابی به هر یک از این اهداف و همچنین ترتیبی که در آن این اهداف باید به دست آیند را تجویز می‌کند.

- م

جموعه ای از تکنیک ها برای دستیابی به هر یک از این فرآیندها.

[6] UML
[7] Procurement Oriented Requirement Engineering

ک مدل محصول که معنایی و نحوی را برای مدل‌سازی محصولات نرم‌افزاری ارائه می‌کند.

برای هر تکرار، تیم توسعه نرم‌افزار نیازمندی‌هایی را به دست می‌آورد که بین مؤلفه‌ها تمایز قائل می‌شوند و نامزدهایی را که با این الزامات مطابقت ندارند با استفاده از تصمیم‌گیری چند معیاره شناسایی می‌کنند، نامزدهای مؤلفه‌ای را که با الزامات مطابقت ندارند رد می‌کنند و نامزدهای باقی‌مانده را برای کشف موارد جدید بررسی می‌کنند. این رویکرد از فرآیند اکتساب و انتخاب نیازمندی های موازی پشتیبانی می‌کند[۲۷].

در مقاله [۲۸] رویکردی را برای ارزیابی مولفه ها از نظر میزان مطابقت آنها با نیازهای مشتری پیشنهاد می‌کند و چارچوب مدیریت تعارض را برای شناسایی مولفه‌ها بر اساس پیشنهادهای حل و ارزیابی ریسک ارائه می‌دهد. رویکرد مهندسی نیازمندی های مبتنی بر COTS (CRE) اهمیت الزامات غیرعملکردی را به عنوان معیارهای تعیین کننده برای ارزیابی اجزای جایگزین برجسته می‌کند.

CRE از ارزیابی مؤلفه‌های نامزد از طریق تعریف معیارهای سیستماتیک پشتیبانی می‌کند که شامل توصیف واضحی از ویژگی‌های کیفی است که مؤلفه‌های نامزد باید برآورده شوند. با این حال، در مواردی که تعداد زیادی جایگزین COTS وجود دارد، فرآیند تصمیم گیری پیچیده است. یکی دیگر از محدودیت‌های رویکرد این است که برای مواردی که الزامات غیرعملکردی به درستی برآورده نمی‌شوند، پشتیبانی نمی‌شود

۵.٤- رویکرد دانش پایه

CARE[8] برای شناسایی مؤلفه‌ها بر کاربرد پایگاه دانش تمرکز دارد. اهداف و الزامات به عنوان اهداف سازمانی مشخص می شوند که بیشتر به اهداف جزء تخصصی می‌شوند. CARE به اهمیت انعطاف پذیر نگه داشتن نیازمندی‌ها اشاره می‌کند، زیرا آن‌ها باید توسط قابلیت‌های اجزای موجود محدود شوند. در این رویکرد، نیازمندی‌ها به‌عنوان نیازمندی‌های بومی کسب‌شده از مشتریان و نیازمندی‌های خارجی اجزای COTS طبقه‌بندی می‌شوند. این روش بر کاهش شکاف بین الزامات مشتری و مؤلفه با استفاده از پایگاه دانش تأکید دارد. مدل فرآیند فعالیت‌های انجام شده برای تعریف عوامل سیستم، اهداف، نیازمندی‌های سیستم، نیازمندی‌های نرم افزار و معماری را توصیف می‌کند. مدل محصول فرمت محصول ایجاد شده با استفاده از فرآیند را توصیف می‌کند. متا مدل محتوا و ساختار دانش را برای رویکرد CARE توصیف می‌کند. این روش

[8] COTS-aware requirement engineering

اهمیت نقشه‌برداری سیستم مورد نیاز و مشخصات محصول را برجسته می‌کند. با این حال، از عدم تطابق احتمالی بین هر دو مشخصات پشتیبانی نمی‌کند[۲۹].

در مقاله [۳۰] یک چارچوب استفاده مجدد از مؤلفه مشکل گرا را با ترکیب یک مکانیسم حل مسئله با مخزن مؤلفه پیشنهاد می‌کند. سیستم نرم افزار کاربردی به عنوان یک مشکل در نظر گرفته می شود که با همکاری گروهی از عوامل که شرح مشکل، بازیابی مؤلفه، استدلال و بررسی مؤلفه های پویا را انجام می دهند، حل می‌شود. این چارچوب یک رویکرد توصیف مسئله‌محور سه سطحی است که کاربران را قادر می‌سازد تا یک مشکل را به شکل اصلاح از بالا به پایین تعریف کنند.

۵. ادغام اجزا

یکپارچه سازی اجزا را می‌توان به عنوان یک فرآیند مکانیکی اجزا با هم نشان داد. به ندرت اتفاق می‌افتد که دو مؤلفه کاملاً مطابقت داشته باشند، بنابراین فرآیند به طور کلی بیشتر از یافتن دو مؤلفه است که با هم وظایف مورد نظر را انجام می‌دهند، و سپس اتصالAPI . مسئله مهم در هنگام ادغام اجزا، مقابله با ناهماهنگی‌هایی است که ممکن است هنگام کنار هم قرار دادن قطعات ایجاد شده توسط طرف های مختلف، معمولاً از یکدیگر بی خبر، رخ دهد. بنابراین، بسیار مهم است که خدمات جزء از طریق یک رابط استاندارد و منتشر شده برای اطمینان از قابلیت همکاری ارائه شود. فرآیند یکپارچه سازی اجزا شامل سازگاری، اعتبارسنجی و آزمایش اجزای انتخاب شده است. در بخش‌های فرعی زیر، تکنیک‌های پیشنهادی برای ادغام را بررسی می کنیم [۳۱].

۱.۵-سازگاری

هر جزء جداگانه بر اساس نیازها و زمینه خاص خود توسعه می‌یابد. معمولاً لازم است نوعی سازگاری برای کاهش تعارض بین اجزای انتخاب شده ایجاد شود. مکانیسم‌های ادغام اجزای مختلف از دیدگاه یک جزء ارائه شده‌اند. فیلترها شاید قدیمی ترین مکانیسم یکپارچه سازی اجزا باشند. این فقط دسترسی به داده‌های اجزا را بدون در نظر گرفتن عملکرد آن‌ها فراهم می کند. کامل ترین مکانیسم ارائه شده توسط مؤلفه‌های جداگانه یک API است که به یک مؤلفه خارجی امکان دسترسی کامل به همه داده‌ها، توابع و رویدادها را می‌دهد. یک زبان برنامه نویسی داخلی نیز برای یکپارچه سازی اجزا استفاده می شود. این روش مبتنی بر این ایده است که چندین مؤلفه یک مخزن داده مشترک را به اشتراک می‌گذارند و یک شی داده را می‌خوانند و می‌نویسند[۳۲].

۲.۵- تست و اعتبار سنجی

اعتبارسنجی مؤلفه‌های تطبیقی یک وظیفه اصلی برای فرآیند CBD است. تکنیک‌های آزمایش و اعتبارسنجی مؤلفه‌های نرم‌افزار بر رفتار مورد انتظار جزء تمرکز می‌کنند تا از صحت رفتار نشان‌داده‌شده اطمینان حاصل کنند. ساختار داخلی اجزاء معمولاً ناشناخته است، مناسب ترین تکنیک برای آزمایش و اعتبارسنجی مؤلفه، آزمایش جعبه سیاه است. تست جعبه سیاه برای قطعات سفارشی شده برای کشف خطاهای عملکردی و رفتاری قطعات جدید و تغییر یافته در قطعات بر اساس مشخصات داده شده است و استراتژی‌های آن شامل تولید مورد تست توسط کلاس‌های معادل، حدس زدن خطا و تست‌های تصادفی است. این تکنیک‌ها بر برخی تصورات از فضای ورودی و عملکرد مورد انتظار جزء در حال آزمایش تکیه دارند. علاوه بر تکنیک‌های تست و اعتبارسنجی نرم‌افزار سنتی، CBD مجموعه‌ای از چالش‌های جدید را معرفی می‌کند، به عنوان مثال، اجزا باید در محیط جدید قرار بگیرند و اغلب نیاز به تشخیص، تشخیص و مدیریت خطاهای نرم‌افزاری در زمان واقعی دارند. روش‌های تولید مولفه‌های خودآزمودنی که در تکنیک‌های خودآزماییدساخته شده‌اند، یکی از راه‌حل‌های ممکن است که می‌تواند به شناسایی خطاها در طول زمان اجرا کمک کند[۳۳].

٦ . استقرار

اجزای انتخاب شده از طریق زیرساخت‌های تعریف شده یکپارچه می‌شوند و این زیرساخت اتصالی را فراهم می‌کند که یک سیستم را از اجزای متفاوت تشکیل می‌دهد. عملیات زیرساخت از سه سطح اصلی تشکیل شده است، یعنی بالاترین سطح انتزاعی که نحوه تعامل اجزای مختلف برای انجام عملکرد مورد نیاز را مشخص می‌کند، سطح پایین‌تر که توسط اجزا برای تعامل و انجام وظایف مشترک استفاده می‌شود و سطح جزء نرم افزاری که خدمات هماهنگی لازم را پیاده سازی می‌کند. ظهور CBSE منجر به طیف وسیعی از مدل‌های مؤلفه مانند CORBA[9]، Enterprise Java Beans (EJB) و NETشده است که فرم‌ها و رابط‌های استاندارد را تعریف می‌کنند.

در مقاله [۳۴] مشخص می‌کند که مرحله استقرار مؤلفه بر پرداختن به دو وظیفه اصلی تمرکز دارد:

- ق

طعات را بسته بندی کنید تا بتوانند در زمان اجرا وصل شوند

[9] Common Object Request Broker Architecture

تصال، قطع و وصل مجدد در زمان اجرا.

Packager یک مشخصات معماری مبتنی بر مؤلفه نوشته شده در زبان مشخصات معماری مبتنی بر مؤلفه را تجزیه و تحلیل می کند تا تعیین کند که آیا بسته های سازگار امکان پذیر هستند یا خیر، یک طرح تولیدی برای تولید بسته‌ها بر اساس آنچه امکان پذیر است و دستورالعمل‌های خروجی برای اجرای طرح انتخاب می‌کند. به طور مشابه، در مقاله [۳۵] چارچوبی را برای یکپارچه‌سازی مؤلفه‌ها ارائه کرد که دارای بررسی‌های سازگاری ویژگی‌های رابط مؤلفه بود و جفت مؤلفه را در زمان اجرا فعال می‌کرد

در مقاله [۳۶] از مفهوم پورت‌هایی استفاده کرد که یکپارچه‌سازی منعطف و سازگار اجزای نرم‌افزار بر اساس اسناد XML را ممکن می‌سازد. این تکنیک‌ها با در نظر گرفتن مشخصات اجزا و ویژگی‌های رابط آن‌ها به کاهش خطرات فرآیند CBDکمک می‌کند و با ارائه تجزیه و تحلیل کمی اجزای نامزد انعطاف پذیری را افزایش می‌دهد.

در مقاله [۳۷] یک مدل رسمی برای شناسایی شرایطی که در آن استراتژی‌های مختلف استقرار اجزا امن و موفق هستند، پیشنهاد کرد. Framework دو نوع نصب موفق و امن موفقیت آمیز را شناسایی می‌کند. تاسیسات آن‌هایی هستند که در آنها برنامه‌های کاربردی به درستی کار می‌کنند، در حالی که تاسیسات ایمن آنهایی هستند که در آنها هیچ برنامه‌ای موجود در اثر نصب آسیب نمی‌بیند. سپس شرایطی را شناسایی می‌کند که برای تضمین نصب ایمن و موفقیت آمیز کافی هستند. چارچوب در مرحله کنونی عمدتاً تئوری است و باید برای برنامه‌های کاربردی واقعی اعمال شود تا مجموعه‌ای از دستورالعمل‌ها را برای مهندسان نرم‌افزار در هنگام ساخت و اجرای نصب‌ها دنبال کنند.

در مقاله [۳۸] مدل CIMO[10] را پیشنهاد کرد که ادغام اجزا را با ارائه یک لایه مؤلفه و یک لایه سرویس تسهیل می‌کند. لایه کامپوننت شامل اجزای CIMO است که بر اساس اشیاء COM مایکروسافت و لایه سرویس شامل پیکربندی، مدیر، کانتینر و سیستم CIMO است. عمدتاً مسئول پشتیبانی از مدیریت، ارتباطات و پیکربندی اجزاء است. پلتفرم CIMO از راه اندازی و یکپارچه سازی اجزا در برنامه CIMO پشتیبانی می‌کند. پلت فرم CIMO دارای یک سیستم CIMO برای هماهنگی کل سیستم است و دارای تعدادی کانتینر CIMO است که مستقیماً شامل اجزای CIMO است. برای تضمین منحصربه‌فرد بودن مؤلفه‌های داخل برنامه، هر مؤلفه نمونه در یک رجیستری متمرکز به نام مدیر مؤلفه CIMO ثبت می‌شود پیکربندی CIMO اجزای اصلی نرم افزار را در یک برنامه در حال اجرا پیاده سازی می‌کند. برنامه های کاربردی مانند فرآیند، توپولوژی و پیوند بین اجزا را پیکربندی می‌کند، پس از ایجاد یک سیستم برنامه به پایان می‌رسد، پیام‌هایی را به سیستم برنامه در حالت اشکال زدایی می‌فرستد و ساختار سیستم برنامه را در هر گره نشان می دهد.

[10] Component integration model

۷. تکامل جزء

یک سیستم نرم افزاری در طول چرخه عمر خود برای بهبود عملکرد، تصحیح عیوب و مواردی از این دست، تکامل مداوم را طی می کند. یک تکامل موفق مؤلفه به مؤلفه‌هایی نیاز دارد تا با نحوی و معنایی سازگار باشند.

در مقاله [۳۹] یک رویکرد چارچوبی را در سیستم‌های مبتنی بر مؤلفه در حال تکامل بر اساس یکپارچه‌سازی مدل‌سازی محصول و فرآیند ارائه می‌کند. فرآیند در حال تکامل توسط Product Tower نشان داده می شود، سلسله مراتبی از اجزا که نماهایی از محصول را در سطوح مختلف اصلاحات ارائه می‌دهد. برج محصول یک توضیح ساختاری صریح از نرم افزار به عنوان یک چارچوب متحد کننده ارائه می‌دهد. یکی از ویژگی‌های مهم این چارچوب، مقیاس پذیری با استفاده از سلسله مراتب طراحی محصول به عنوان نقطه مرجع برای توسعه یک محصول است. توانایی تخصصی شدن در فرآیند برای هر جزء به اجزای سازنده اجازه می‌دهد تا با استفاده از روش‌های مختلف و در واقع نسخه های مختلف ابزار توسعه توسعه یابند. تکامل یک سیستم از طریق تکامل سلسله مراتب طراحی مدیریت می‌شود.

مفهوم نسخه و پیکربندی نیز در زمینه CBD اعمال می‌شود. یک مدل پیکربندی مبتنی بر کتابخانه‌های مؤلفه ارائه شده است که به پیکربندی و تکامل اجزای سیستم کمک می کند. مدل پیکربندی وابستگی‌های مختلف بین اجزای یک سیستم مبتنی بر مولفه را پیگیری می کند. معیارها همراه با مستندات مؤلفه برای تعیین تأثیر تغییر احتمالی بر سیستم مبتنی بر مؤلفه استفاده می شود. اسناد توسعه نقش کلیدی در مراحل مختلف تکامل نرم افزار دارند. تکامل نرم افزار به این معنی است که توضیحات تغییر در طول زمان تغییر می کند، یعنی هر مرحله تکامل مستلزم تغییرات مجموعه مناسبی از اسناد توسعه است.

در مقاله [۴۰] یک مدل سیستم مشترک با پایه و اساس برای اجزایی با قابلیت تکامل به شیوه ای کنترل شده پیشنهاد می کند. در طول توسعه، هر مرحله از فرآیند تکامل مستلزم تغییراتی در مجموعه مناسبی از اسناد توسعه است. به منظور مدل سازی وابستگی بین این اسناد، مفهوم قراردادهای الزامی/تضمین معرفی شده است. این مدل شامل مفاهیم یک نوع و انتزاعی و همچنین توضیحات عینی برای انواع است. در طول توسعه سیستم، مجموعه‌ای از این توضیحات ایجاد می‌شود. تکامل نرم افزار به این معنی است که این توصیفات در طول زمان تغییر می کنند. این قراردادها عواقب تکامل یک جزء یا مجموعه ای از اجزا را در کل سیستم نشان می‌دهد. این قراردادها را می توان هر زمان که مشخصات یک جزء تغییر کرد، مجدداً بررسی کرد و امکان تعیین تأثیر بر مرحله تکامل مربوطه را فراهم کرد.

۸. نتیجه گیری

در این کار، ما یک بررسی ادبی از تحقیقات پیشرفته در زمینه CBD ارائه کرده‌ایم. مناطق مورد بررسی تکنیک‌هایی برای شناسایی و انتخاب جزء، یکپارچه سازی، استقرار و تکامل بودند. در حوزه شناسایی و انتخاب مؤلفه‌ها، محققان مجموعه‌ای از رویکردهای سیستماتیک را برای شناسایی و انتخاب مؤلفه‌ها با استفاده از تکنیک‌های تحلیل سلسله مراتبی، تکراری و مبتنی بر دانش پیشنهاد کرده‌اند. هدف از این تکنیک‌ها شناسایی و رتبه بندی اجزای نامزد است. و در نهایت اجزایی را انتخاب کنید که به بهترین وجه نیازهای ذینفعان را برآورده می‌کنند. با این حال، اکثر تکنیک‌های پیشنهادی یک فرآیند انتخاب بر اساس یک تعریف الزامی دقیق دارند. این نشان می‌دهد که هر یک از اجزاء باید حذف شوند زیرا الزامات ذینفعان را برآورده نمی‌کنند یا برای برآوردن چنین الزامات محدودکننده‌ای نیاز به تغییر بزرگی دارند. برای کارهای آتی، نیاز به یک فرآیند مشارکتی وجود دارد که در آن هم الزامات ذینفعان و هم اجزای نامزد بتوانند بین الزامات ذینفعان و محدودیت‌های مؤلفه‌های موجود مبادله شوند. به طور مشابه، یکپارچه سازی مؤلفه باید به مسائل مربوط به تأیید سازگاری مؤلفه و بررسی سازگاری با استانداردهای باز بپردازد. توانایی یک سیستم برای اینکه بتواند به شیوه ای کنترل شده تکامل یابد نیز مهم است و ارتباط آن با بقیه چرخه حیات CBD باید به دقت مدیریت شود. تحقیقات آینده نیاز به تجزیه و تحلیل عوامل خطر و پیچیدگی مرتبط با یکپارچه سازی، قابلیت همکاری و استقرار CBD دارد. کار تحقیقاتی بیشتری باید بر روی هزینه‌های مربوط به اکتساب و یکپارچه‌سازی اجزا، آزمایش و قابلیت نگهداری متمرکز شود. مسائل پیچیدگی مربوط به اندازه، رابط، معناشناسی و جفت شدن، و کیفیت کلی سیستم نیز برای محققین مورد توجه است. نیاز به پشتیبانی خودکار برای CBD برای تحقق پتانسیل های کامل آن وجود دارد. همانطور که، هدف CBD ساختن یک سیستم به روش مقرون به صرفه است. بنابراین، مجموعه ای از ابزارها برای انتخاب جزء و یکپارچه سازی ضروری است. ابزارهای دیگر برای تست مؤلفه و پیکربندی مؤلفه مفید خواهند بود. برای نتیجه گیری، تجربیات به دست آمده در CBD ارزشمند هستند و نیاز به گزارش و مستندسازی بیشتر دارند. تلاش تحقیقاتی بیشتری برای رسیدگی به خطرات CBD ضروری است. و تحقیقات تجربی بیشتر منجر به استخراج مدل‌های دقیق‌تر برای سیستم‌های مبتنی بر مولفه خواهد شد.

منابع

1. Khan SU, Khan AW, Khan F, Khan MA, Whangbo TK. Critical success factors of component-based software outsourcing development from vendors' perspective: A systematic literature review. IEEE Access. 2021 Dec 27;10:1650-8.
2. Diwaker C, Tomar P, Solanki A, Nayyar A, Jhanjhi NZ, Abdullah A, Supramaniam M. A new model for predicting component-based software reliability using soft computing. IEEE Access. 2019 Oct 11;7:147191-203.
3. Jha SK, Mishra RK. A review on reusability of component based software development. Reliability: Theory & Applications. 2019;14(4):32-6.
4. Lu T, Liu C, Duan H, Zeng Q. Mining component-based software behavioral models using dynamic analysis. IEEE Access. 2020 Apr 13;8:68883-94.
5. Saari M, Nurminen M, Rantanen P. Survey of Component-Based Software Engineering within IoT Development. In2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO) 2022 May 23 (pp. 824-828). IEEE.
6. Khan AW, Khan SU, Alwageed HS, Khan F, Khan J, Lee Y. AHP-Based Systematic Approach to Analyzing and Evaluating Critical Success Factors and Practices for Component-Based Outsourcing Software Development. Mathematics. 2022 Oct 26;10(21):3982.
7. Gottschalk S, Yigitbas E, Engels G. Model-driven Continuous Experimentation on Component-based Software Architectures. In2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C) 2022 Mar 12 (pp. 20-24). IEEE.
8. Padhy N, Panigrahi R, Satapathy SC. Identifying the reusable components from component-based system: proposed metrics and model. InInformation Systems Design and Intelligent Applications: Proceedings of Fifth International Conference INDIA 2018 Volume 2 2019 (pp. 89-99). Springer Singapore.
9. Tiwari UK, Kumar S. Component-based software engineering: Methods and metrics. CRC Press; 2020 Nov 18.
10. Negi P, Tiwari UK. Machine learning algorithm for assessing reusability in component based software development. EasyChair Preprint. 2020 Sep 7;4142:1-8.
11. Tomar D, Tomar P. New component-based reliability model to predict the reliability of component-based software. International Journal of Reliability and Safety. 2019;13(1-2):83-95.
12. Irina-Miruna RA. Advantages and challenges of using component–based software development in the vision of building a modern educational system. InIE 2019 International Conference [En línea]. Disponible en: http://www. conferenceie. ase. ro/wp-content/uploads/2019/06/ProceedingsIE2019/advantages_and_challenges_of_using_component_ based_software_development_in_the_vision_of_building_a_. pdf 2019.
13. Kahtan H, Abdulhak M, Al-Ahmad AS, Alzoubi YI. A model for developing dependable systems using a component-based software development approach (MDDS-CBSD). IET Software. 2022.
14. Rafi S, Yu W, Akbar MA, Alsanad A, Gumaei A. Prioritization based taxonomy of DevOps security challenges using PROMETHEE. IEEE Access. 2020 Jun 1;8:105426-46.
15. Duc AN, Jabangwe R, Paul P, Abrahamsson P. Security challenges in IoT development: a software engineering perspective. InProceedings of the XP2017 scientific workshops 2017 May 22 (pp. 1-5).
16. Pérez Muñoz ÁG. Análisis y Adaptación de las Herramientas TASTE para el Desarrollo Basado en Componentes.

17. Liu L, Yao Y, Li J. Development of a novel component-based open CNC software system. The International Journal of Advanced Manufacturing Technology. 2020 Jun;108:3547-62.
18. Gehlot S, PoojaRana RS, Singh R. Complexity Metrics for Component Based Software-A Comparative Study. J. Comput.. 2019;14(6):389-96.
19. Heidmann EF, von Kurnatowski L, Meinecke A, Schreiber A. Visualization of Evolution of Component-Based Software Architectures in Virtual Reality. In2020 Working Conference on Software Visualization (VISSOFT) 2020 Sep 28 (pp. 12-21). IEEE.
20. Banerjee P, Sarkar A. Quality evaluation of component-based software: An empirical approach. International Journal of Intelligent Systems and Applications. 2018 Dec 1;11(12):80.
21. Dimri SC, Tiwari UK, Ram M. Reliability estimation of component based software system with path based model. Nonlinear Studies. 2019 Apr 1;26(2).
22. Iliev O, Yoshinov R. Component-based Software Architecture Applied for Design of Heritage Content. Digital Presentation and Preservation of Cultural and Scientific Heritage. 2021 Sep 10;11:99-110.
23. Aggarwal J, Kumar M. Software metrics for reusability of component based software system: a review. Int. Arab J. Inf. Technol.. 2021 May 1;18(3):319-25.
24. Jha SK, Mishra RK. Predicting and accessing security features into component-based software development: a critical survey. InSoftware Engineering: Proceedings of CSI 2015 2019 (pp. 287-294). Springer Singapore.
25. Sastypratiwi H, Yulianti Y. Web Application Development using MVC-component-based approach. In2019 International Conference on Data and Software Engineering (ICoDSE) 2019 Nov 13 (pp. 1-5). IEEE.
26. Khan S, Jha SK, Khatri SK. Dependability and Trustworthiness Analysis for Component Based Software Development. Int. J. Rec. Techn. Eng. 2019;8:2277-3878.
27. Derakhshanmanesh M, Ebert J, Grieger M, Engels G. Model-integrating development of software systems: a flexible component-based approach. Software & Systems Modeling. 2019 Aug 1;18:2557-86.
28. Jingga K, Sunindyo WD. Component-based Development Using Moodle As Alternative for E-learning Software Development. In2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE) 2020 Oct 6 (pp. 125-130). IEEE.
29. Lau KK, Cola SD. AN INTRODUCTION TO COMPONENT-BASED SOFTWARE DEVELOPEMENT. 2019
30. Yigitbas E, Josifovska K, Jovanovikj I, Kalinci F, Anjorin A, Engels G. Component-based development of adaptive user interfaces. InProceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems 2019 Jun 18 (pp. 1-7).
31. Latif K, Yusof Y, Kadir AZ. Development of virtual component-based STEP-compliant CNC system. Progress in Additive Manufacturing. 2022 Feb;7(1):77-85.
32. Tula AK, Eden MR, Gani R. Component based development of computer-aided tools for different applications. InComputer Aided Chemical Engineering 2019 Jan 1 (Vol. 46, pp. 91-96). Elsevier.
33. Cho S, Lee JM, Woo JH. Development of production planning system for shipbuilding using component-based development framework. International Journal of Naval Architecture and Ocean Engineering. 2021 Jan 1;13:405-30.
34. Biondi A, Buttazzo G, Bertogna M. A design flow for supporting component-based software development in multiprocessor real-time systems. Real-Time Systems. 2018 Oct;54:800-29.

35. Jaffar RN, Hussain AA, Chiad W. A new model for study of quality attributes to components based development approach. Periodicals of Engineering and Natural Sciences. 2019 Sep 10;7(3):1177-85.
36. Sehgal R, Mehrotra D, Nagpal R. Complexity metrics for component-based software system. InSoft Computing: Theories and Applications: Proceedings of SoCTA 2017 2019 (pp. 13-22). Springer Singapore.
37. Illarramendi M, Etxeberria L, Elkorobarrutia X, Sagardui G. Runtime Contracts Checker: Increasing Robustness of Component-Based Software Systems. InIOP Conference Series: Materials Science and Engineering 2019 Jul 1 (Vol. 575, No. 1, p. 012006). IOP Publishing.
38. Wang Y, Li F, Zhang B, Li X. Development of a component-based interactive visualization system for the analysis of ocean data. Big Earth Data. 2022 Apr 3;6(2):219-35.
39. Seewald A, Schultz UP, Roeder J, Rouxel B, Grelck C. Component-based computation-energy modeling for embedded systems. InProceedings Companion of the 2019 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity 2019 Oct 20 (pp. 5-6).
40. Wang M, Ma Y, Li G, Zhou W, Chen L. Multi-value models for allocation of software component development costs based on trustworthiness. IEEE Access. 2020 Jul 6;8:122673-84.