عنوان مقاله: Bounded Context در توسعه نرم افزار

نویسنده مقاله: تیم فنی نیک آموز

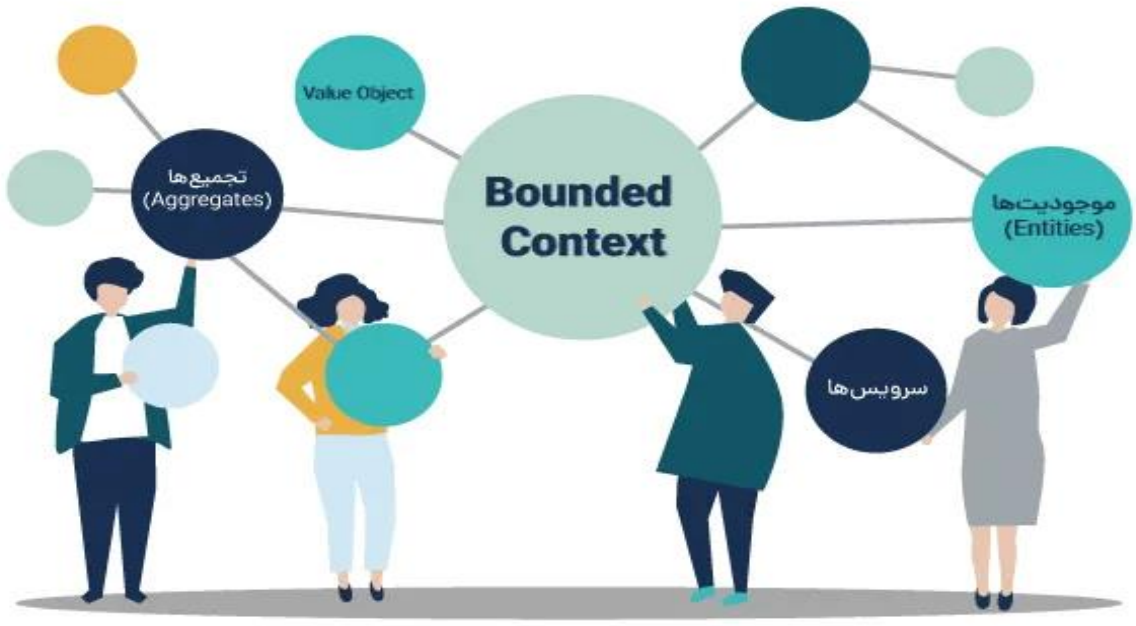
تاریخ انتشار: ۲۸ آذر ۱۴۰۲

منبع: https://nikamooz.com/bounded-context-in-software-development/

مفهوم Bounded Context یک بخش بنیادی از طراحی دامنه محور (Domain Driven Design) محسوب می‌شود و به کمک آن، می‌توان پیچیدگی‌های سیستم‌های نرم افزار گسترده را مدیریت کرد. در این مطلب، ابتدا مفهوم Bounded Context، مزایا، معایب و کاربردهای آن بررسی می‌شوند، سپس نحوه طراحی و تست Bounded Context به‌همراه انواع Relationship مربوط به آن، شرح داده خواهند شد.

Bounded Context چیست ؟

Bounded Context مشابه یک کانتینر مجازی عمل می‌کند و به‌عنوان بخش منحصربه‌فردی از سیستم نرم‌افزاری، قوانین، ترمینولوژی و اهداف مربوط به خود را شامل می‌شود. در حقیقت، مفهوم Bounded Context ، نوعی مرزبندی منطقی و فیزیکی در درون دامنه‌ای (Domain) است که در آن، یک مدل دامنه اعمال شده است. با استفاده از Bounded Context در یک دامنه، مواردی همچون موجودیت‌ها (Entities)، تجمیع‌ها (Aggregates)، سرویس‌ها، Value Object ها و سایر ریپازیتوری‌های آن دامنه خاص کپسوله‌سازی می‌شوند.



مزایای Bounded Context

بارزترین مزیت های Bounded Context به شرح زیر است:

- **ماژولاریتی:** مفهوم Bounded Context با تقسیم‌بندی یک سیستم پیچیده به بخش‌های کوچک و قابل مدیریت، ماژولاریتی (Modularity) را تقویت می‌کند.
- **شفافیت ارتباط:** با استفاده از Bounded Context ، یک زبان مشترک ارائه می‌شود و تعامل میان ذی‌نفعان، توسعه‌دهندگان و متخصصان دامنه (Domain Expert)، بدون ابهام خواهد بود.
- **هم‌ترازی با دامنه:** مفهوم Bounded Context باعث می‌شود تا سیستم نرم‌افزاری، دقیقاً هم‌تراز با مشکلات موجود در دامنه باشد.
- **عملیات مستقل و غیرمتمرکز:** تیم‌های توسعه می‌توانند درون Bounded Context های خود به‌صورت خودمختار، عملیات مختلفی را انجام دهند. این استقلال، پیاده‌سازی و تصمیم‌گیری را سرعت می‌بخشد و به اعضای تیم‌ها این امکان را می‌دهد که برروی وظایف مخصوص به خود، تمرکز کنند.

معایب Bounded Context

هرچند مفهوم Bounded Context در توسعه نرم افزار حائز اهمیت است و مزایای مخصوص به خود را دارا است، اما کاستی‌هایی نیز دارد که در ادامه به آن‌ها می‌پردازیم.

- **افزایش پیچیدگی:** پیاده‌سازی مفهوم Bounded Context به طراحی و برنامه‌ریزی دقیق نیاز دارد و اگر به‌درستی انجام نشود، منجر به ایجاد ناسازگاری در Bounded Context ها خواهد شد.
- **چالش‌های نگاشت Context:** زمانی که چند Bounded Context به تعامل می‌پردازند، Context Mapping یک امر ضروری برای ایجاد ارتباط و سازگاری میان آن‌ها قلمداد می‌شود.
- **احتمال وجود نسخه تکراری:** در برخی شرایط، ریسک ایجاد کاربرد یا داده‌های تکراری (Duplicated Data) در Bounded Context های مختلف وجود دارد. به همین دلیل، لازم است طراحی مرزها و اینترفیس‌های میان Context ها با حوصله و به دقت انجام شوند.

تا این بخش از مطلب، با مفهوم Bounded Context و مزایا و معایب آن آشنا شده‌اید. در بخش بعدی، کاربرد های Bounded Context مورد بررسی قرار می‌گیرند.

کاربرد های Bounded Context

موارد استفاده از Bounded Context به شرح زیر است:

مدل سازی دامنه

یک پلتفرم تجارت الکترونیک (E-Commerce) را به‌عنوان مثال در نظر بگیرید. با استفاده از مفهوم Bounded Context، توسعه‌دهندگان می‌توانند Context های منحصربه‌فردی همچون کاتالوگ فروش، مدیریت سفارش‌ها و ارسال کالا را شناسایی کنند. شایان ذکر است که در هر یک از این Context ها، روی دامنه خاصی از سیستم تجارت الکترونیک تمرکز می‌شود. با تعریف Bounded Context های گوناگون، توسعه‌دهندگان می‌توانند مدل‌هایی را ایجاد کنند که به‌طور دقیق، نمایان‌گر دامنه‌های مربوطه باشند. در چنین شرایطی، مواردی همچون درک دقیق نیازمندی‌های کسب و کار و تسهیل فرآیند پیاده‌سازی، تضمین خواهد شد.

تعامل و مشارکت تیمی

تصور کنید یک تیم توسعه به روش Agile، روی یک پلتفرم شبکه اجتماعی کار می‌کنند. با سازگارسازی مفهوم Bounded Context ، این امکان به وجود می‌آید که سیستم به Context های مختلف ازجمله پروفایل‌های کاربری، مدیریت پست‌ها و سیستم نوتیفیکیشن، تقسیم‌بندی شود. به این ترتیب، هر یک از تیم‌ها می‌توانند به‌صورت مستقل، روی یک Bounded Context مشخص کار کنند و دیگر به هماهنگی گسترده با سایر تیم‌ها نیاز نداشته باشند. این تقسیم‌بندی توسعه جریان کار، به استقلال تیم‌های گوناگون و افزایش سرعت پیاده‌سازی Feature ها منجر می‌شود.

مقیاس پذیری سیستم

یک اپلیکیشن بانکی را در نظر بگیرید که به تراکنش‌های مالی متعددی رسیدگی می‌کند. با به‌کارگیری Bounded Context در این اپلیکیشن، توسعه‌دهندگان می‌توانند کارکردهای بخش‌های مختلف سیستم، شامل مدیریت حساب‌ها، فرآیند تراکنش و مدیریت وام‌ها را به Context های مستقل تقسیم‌بندی کنند. به‌عنوان مثال، اگر Context مربوط به پردازش تراکنش‌ها، متحمل درخواست‌های متعددی از سمت کاربران باشد، می‌توان آن را به‌صورت افقی Scale کرد؛ به طوری که این افزایش مقیاس، روی سایر Context ها تأثیری نگذارد.

هم تراز با قوانین معماری سرویس گرا

یک سیستم رزرو مسافرتی را به‌عنوان نمونه در نظر بگیرید. مفهوم Bounded Context دقیقاً با قوانین موجود در معماری سرویس گرا (Service Oriented Architecture) هم‌جهت است. هر یک از Bounded Context ها این امکان را دارند که درون یک سرویس مانند سرویس رزرو هتل، سرویس خرید بلیط هواپیما و سرویس پرداخت، کپسوله‌سازی شوند. این جداسازی یا اصطلاحاً Decoupling، انعطاف‌پذیری را افزایش داده و قابلیت نگهداری (Maintenance) را تسهیل می‌بخشد. در این شرایط، می‌توان اعمال تغییرات در یک Context، مثل افزودن یک روش پرداخت جدید در سرویس پرداخت را بدون اثر گذاشتن آن روی سایر بخش‌های سیستم انجام داد.

معماری میکروسرویس

معمولاً در معماری میکروسرویس (Microservice Architecture)، هر یک از میکروسرویس‌ها نمایان‌گر Bounded Context های مختلف هستند. در عمل، هرکدام از میکروسرویس‌ها، یک قابلیت یا دامنه خاصی از کسب و کار را کپسوله‌سازی می‌کنند و به واسطه آن، تیم‌های توسعه‌دهنده می‌توانند به‌صورت جداگانه کار کنند. به این ترتیب، با معرفی مفهوم Bounded Context ، خودکفایی و مقیاس‌پذیری در میکروسرویس‌ها ترویج می‌یابد. به منظور آشنایی بیشتر با این معماری مشهور، پیشنهاد می‌شود مقاله مزایای میکروسرویس را نیز مطالعه کنید.

نیازمندی های در حال تغییر کسب و کار

در سناریوهایی که نیازمندی‌های کسب و کار، دستخوش تغییر و تکامل مداوم است، مفهوم Bounded Context نقش کلیدی ایفا می‌کند؛ زیرا به واسطه آن، یک مکانیزم مناسب برای سازگاری با سیستم نرم‌افزاری ارائه می‌شود و تیم‌ها می‌توانند بدون اثرگذاری برروی کل سیستم، Context های خاصی را تغییر یا توسعه دهند. این موضوع، انعطاف‌پذیری و قابلیت پاسخ‌گویی به تغییرات را به‌همراه دارد.

Bounded Context چگونه تعیین می شود؟

تعیین Bounded Context ها می‌تواند با چالش همراه باشد؛ زیرا برای مشخص کردن مفهوم Context Bounding ، لازم است درک عمیق و صحیحی از دامنه و نیازمندی‌های سیستم داشته باشید و در عین حال، بتوانید به‌راحتی با ذی‌نفعان، متخصصان دامنه و توسعه‌دهندگان به تعامل و مشارکت بپردازید. به منظور رسیدن به این فرآیند، می‌توان از روش‌های شهودی و تکنیک‌هایی همچون Context Mapping، رویدادهای دامنه (Domain Events)، قابلیت‌های کسب و کار و User Story ها کمک گرفت. برای آشنایی با مفهوم User Story، می‌توانید مقاله روش های اولویت بندی بک لاگ در اسکرام را مطالعه کنید. در عمل، تمامی این روش‌ها در تشخیص مرزبندی‌ها، تعاملات دامنه‌ها و زیردامنه‌های گوناگون و همچنین انواع و الگوهای یکپارچه‌سازی (Integration) به شما کمک خواهند کرد.

نحوه طراحی Bounded Context

طراحی مفهوم Bounded Context ، یک فرآیند تکرارشونده و تکاملی محسوب می‌شود و نیازمند دریافت بازخورد و بازتعریف مداوم است. بدین طریق، تضمین می‌شود که Bounded context با دامنه و نیازمندی‌های کسب و کار، هم‌تراز و هم‌جهت است. برای این کار، قوانین، الگوها و روش‌های کاربردی گوناگونی، ازجمله الگوهای طراحی دامنه محور (Domain Driven Design)، معماری میکروسرویس و طراحی استراتژیک قابل استفاده هستند.

تست Bounded Context چگونه انجام می شود؟

تست Bounded Context ها، یک بخش ضروری به‌حساب می‌آید که به واسطه آن، خیال شما از بابت کیفیت و قابل اکتفا بودن سیستم راحت خواهد شد. درواقع، فرآیند Test یا همان آزمودن Bounded Context ها، در راستی‌آزمایی کارکرد آن‌ها کاربرد دارد. به این ترتیب، می‌توانید از Unit Testing برای بررسی صحت و سازگاری کدها و از Integration Testing به منظور بررسی تعامل و یکپارچگی کامپوننت‌های درون یک یا چند Bounded Context بهره‌مند شوید. لازم به ذکر است که امکان اجرای تست به‌صورت سرتاسری (End-To-End) نیز وجود دارد و با اعمال آن، می‌توانید کارایی، قابلیت استفاده و قابل اکتفا بودن سیستم خود را بررسی کرده و مشخص کنید آیا این سیستم، نیازمندی‌های کاربر و قابلیت‌های کسب و کار را پاسخگو است.

روابط Bounded Context در طراحی دامنه محور (DDD)

Bounded Context ها مستقل از Context های دیگر کار کنند. با این وجود، گاهی برای تکمیل یک Task خاص یا تبادل اطلاعات، لازم است با یکدیگر تعامل داشته باشند. روابط میان Bounded Context ها حائز اهمیت است؛ زیرا به کمک این روابط، تعاملات میان بخش‌های مختلف سیستم تعریف می‌شوند و مرزبندی‌ها و وظایف بین تیم‌های گوناگون مشخص خواهند شد.

انواع روابط Bounded Context

انواع روابط Bounded Context در DDD به شرح زیر است:

- **رابطه Partnership:** ارتباطی که در آن، دو یا چند Bounded Context مشارکت می‌کنند و اطلاعات را به اشتراک می‌گذارند. می‌توان این ارتباط را ازطریق Integration Event یا استناد به یک سرویس خاص ایجاد کرد.
- **رابطه هسته اشتراکی (Shared Kernel):** در این نوع از رابطه، دو Bounded Context یک مدل، کد یا شمای پایگاه داده (Database Schema) مشترک دارند. این کامپوننت‌های مشترک باید پایدار، پخته و پذیرفته‌شده توسط هردو تیم باشند و باید اعمال تغییرات در این اجزا، به دقت و با هماهنگی انجام شود.
- **رابطه مشتری – تأمین‌کننده (Customer-Supplier):** در این ارتباط، یکی از Bounded Context ها، سرویس‌ها یا دیتایی را به دیگری ارائه می‌دهد. به این ترتیب، Context مربوط به مشتری، برای دریافت برخی قابلیت‌ها یا داده‌ها، به Context تأمین‌کننده اکتفا می‌کند.
- **ارتباط Conformist:** این ارتباط زمانی به وجود می‌آید که یک Bounded Context ، دنبال‌کننده زبان فراگیر (Ubiquitous Languages) و مفاهیم Bounded Context دیگری باشد. به این ترتیب، تضمین می‌شود که ارتباط میان آن‌ها، شفاف و بدون ابهام است.
- **Anticorruption Layer:** در این رابطه، یک Bounded Context برای ترجمه زبان خود و زبان Bounded Context دیگر، از یک لایه (Layer) استفاده می‌کند. در چنین شرایط، دو Context با ادبیات و مدل‌های متفاوت، امکان برقراری ارتباط با یکدیگر را دارند.

- **Open Host Service:** این Relationship زمانی رخ می‌دهد که یک Bounded Context ، یک API عمومی و باز را در دسترس قرار می‌دهد؛ به طوری که سایر Context ها، امکان استفاده از آن را دارند. در چنین شرایطی، سایر Context ها می‌توانند از کارکردهای Host Context به‌صورت استانداردشده بهره‌مند شوند.
- **زبان منتشرشده (Published Language):** در این رابطه، یک Bounded Context ، مدل و ادبیات خود را منتشر می‌کند تا سایر Context ها از آن‌ها استفاده کنند. این رابطه در مواقعی مناسب است که بخواهید چند Bounded Context ، بدون یکپارچه‌سازی مستقیم، با یکدیگر مشارکت داشته باشند.
- **ارتباط Separate Ways:** این ارتباط به شرایطی اشاره دارد که دو یا چند Bounded Context ، دیگر باهم رابطه ندارند و تیم‌های مربوطه می‌توانند به‌صورت مستقل کار کنند.

با توجه به اهمیت طراحی دامنه‌محور در توسعه نرم‌افزار، پیشنهاد می‌شود **مقاله پیاده سازی معماری Domain Driven Design و تست نویسی + راهنمای گام به گام** را مطالعه کنید تا چگونگی کار با DDD را بهتر درک کنید.

سخن پایانی: مفهوم Bounded Context

مفهوم Bounded Context و استفاده از آن در توسعه نرم‌افزار، مزیت‌هایی همچون شفافیت و درک مشترک، تعامل کارآمد اعضا، کارکرد مستقل، مقیاس‌پذیری و سازگاری را به همراه دارد. با کمک Bounded Context ها، یک رویکرد ساختاریافته برای مدیریت پیچیدگی‌ها به وجود می‌آید و فرآیند توسعه و نگهداری سیستم‌های نرم‌افزاری تسهیل می‌یابند. در این مطلب، نکات کلیدی و لازم برای درک مفهوم Bounded Context آموزش داده شدند تا شما از آن، به‌عنوان راهنما بهره‌مند شوید.