Engineer

Data mesh

att låta domänteam leverera sina analysdata som produkter

Vad det är

Ett data mesh är ett sätt att organisera analysdata i en stor organisation så att teamen som driver varje del av verksamheten också äger, publicerar och ansvarar för de data som den delen producerar. Zhamak Dehghani, då principal technology consultant på Thoughtworks, lade fram idén på Martin Fowlers webbplats i en artikel som publicerades i flera delar i maj 2019. I december 2020 formulerade hon om den som fyra principer: domänorienterat ägarskap av data och arkitekturen kring dem, data som produkt, självbetjänad datainfrastruktur som plattform och federerad styrning som maskinerna upprätthåller, på engelska federated computational governance. Enligt henne är de fyra tillsammans både nödvändiga och tillräckliga, och hon räknar med att metoder och verktyg kommer att förändras medan principerna ligger fast.

Idén gäller analytiska data, inte operativa. Dehghani drar gränsen så här: Operativa data ligger i databaserna bakom de system som håller verksamheten igång, de är transaktionella och visar hur läget är just nu. Analytiska data är verksamhetens historik över tid, aggregerad och modellerad, och det är på dem rapporterna bygger och modellerna tränas. De operativa data låter hon ligga kvar där de är, inne i varje tjänst. Det är den analytiska sidan hon vill åt, för det är där hon ser friktionen när verksamheten växer.

Domänägarskap. Ett data mesh delas upp längs gränser som organisationen redan har. Dehghanis genomgående exempel är ett digitalt medieföretag med domäner som poddar och artister. Teamet som driver poddarna och erbjuder gränssnitten för att publicera dem bör också leverera historiken över publicerade poddar och hur många som har lyssnat på dem över tid. Gränserna är hämtade från domändriven design, där varje domäns bounded context avgränsar det område där en ändring stannar lokalt. Därför är det också ett bra ställe att lägga ägarskapet till data.

Data som produkt. När data sprids över många team blir de svårare att hitta och att lita på. Varje team måste därför behandla de data det delar som en produkt, och dem som använder dem som kunder. Artikeln från 2019 räknar upp vad som krävs: En dataprodukt ska gå att hitta, ha en fast adress, vara pålitlig, beskriva sig själv, kunna samverka med andra enligt gemensamma standarder och skyddas av gemensam åtkomstkontroll. En produktägare i domänen ansvarar för mått som kvalitet, hur länge konsumenterna får vänta på data och hur nöjda de är. Ansvaret för kvaliteten flyttas uppströms, så nära källan som möjligt. Arkitektoniskt är en dataprodukt den minsta enhet som kan driftsättas för sig, och den består av tre delar: koden som transformerar och levererar data och upprätthåller reglerna för dem, själva data tillsammans med deras metadata och infrastrukturen som kör alltihop.

En självbetjäningsplattform. Att bygga, driva och övervaka en dataprodukt kräver specialiserad infrastrukturkompetens, och den går inte att kopiera in i varenda domän. Ett plattformsteam erbjuder därför verktyg på en högre nivå, bland annat för lagring, produktscheman, deklarerade datapipelines och spårning av varifrån data kommer, så att en utvecklare med bred profil kan bygga en dataprodukt utan den specialistkunskapen.

Federerad styrning. Produkterna måste ändå passa ihop, annars kan ingen kombinera dem. En grupp produktägare från domänerna och från plattformen avgör vilka regler som ska gälla överallt och vilka som lämnas åt den enskilda domänen, och plattformen upprätthåller de gemensamma reglerna automatiskt. Dehghanis exempel är lyssningssiffror: Hur domänen för poddar modellerar dem är domänens egen sak, men hur en lyssnare identifieras är en gemensam fråga, eftersom samma person också dyker upp som användare i andra domäner.

Använd begreppet i ett beslut

Använd ”Vad är ett data mesh?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Siffror från försäljning, drift och ekonomi samlade på ett ställe, så att två avdelningar slutar komma med två olika svar till samma möte.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Gjorde pipelinen för geodata 50 gånger snabbare — 50x snabbare datapipelines för geodata

Gjorde pipelinen för geodata 50 gånger snabbare Hör av er →

Varför det spelar roll

Dehghanis argument börjar i det som går fel när ett stort företag driver en central datasjö eller ett centralt analysdatalager. Data från alla domäner dras in i en plattform som äger alltihop, och den drivs av ett team specialister på dataverktyg som sitter långt från både dem som skapar data och dem som använder dem. Hon pekar på tre sätt som det fallerar på. Plattformen är central och monolitisk, så den får det svårt när källorna och konsumenterna blir fler. Den är uppdelad i pipelinesteg som inläsning, rensning, aggregering och leverans, så ett enda nytt mått, till exempel hur ofta poddar spelas upp, kräver en ändring i varje steg och en release som måste samordnas mellan team. Och ägarskapet är uppdelat i silor och högt specialiserat: Ingenjörerna som ska få ut något av data känner inte domänerna de kommer ifrån särskilt väl, medan teamen som producerar data inte har något incitament att göra dem korrekta.

Ett data mesh ändrar vem som städar. I den gamla modellen kommer data in med sina fel, och ett centralt team tvättar dem efter inläsningen. I ett data mesh blir en domäns datamängd en dataprodukt först när domänen själv har kontrollerat den mot de överenskomna kvalitetsmålen och de gemensamma reglerna. De som vet vad ett fält ska betyda är också de som står för det.

Styrningen byter också form. Artikeln från 2020 ställer de två modellerna mot varandra i en tabell. Ett centralt team ansvarar självt för kvalitet, säkerhet och regelefterlevnad. Ett federerat team bestämmer hur kvalitet ska modelleras och vilka säkerhets- och regelkrav plattformen ska bygga in och övervaka. Förr mättes framgång i antalet tabeller under styrning. I ett data mesh mäts den i hur mycket produkterna faktiskt används. Manuella kontroller som försöker förhindra fel ersätts av automatiserade processer som upptäcker fel och rättar till dem i efterhand.

Inget av detta framställs som ett universalsvar. I artikeln från 2019 medger Dehghani att den centrala modellen kan fungera för organisationer med en enklare domän och färre slags konsumenter, och skriver att den fallerar i verksamheter med rika domäner, många källor och konsumenter med mycket olika behov. Om ett data mesh är mödan värt beror på vilken av de två sorterna organisationen tillhör.

Så ser det ut i praktiken

Ett riktigt data mesh sträcker sig över många team och många datalager. Skissen nedan håller sig till en enda PostgreSQL-databas, så att varje SQL-sats kan kontrolleras mot manualen, och den visar de två minsta greppen: en domän som publicerar en produkt och en plattform som upprätthåller en gemensam regel.

Orderteamet äger driftstabellerna i schemat `orders`. Sina analysdata publicerar det i ett eget schema, som en versionerad vy med en beskrivning och en uttrycklig tilldelning av behörigheter:

```
-- orderteamet äger det här schemat och allt som ligger i det
CREATE SCHEMA order_data;

CREATE VIEW order_data.orders_by_day_v1 AS
SELECT order_date, currency, count(*) AS orders, sum(total) AS revenue
FROM orders.orders
WHERE status = 'paid'
GROUP BY order_date, currency;

COMMENT ON VIEW order_data.orders_by_day_v1 IS
  'Betalda order per dag och valuta. Ägare: orderteamet.';

GRANT USAGE ON SCHEMA order_data TO analysts;
GRANT SELECT ON order_data.orders_by_day_v1 TO analysts;
```

PostgreSQL-manualen förklarar varför det räcker på konsumentsidan: Den som använder en vy behöver bara USAGE-privilegiet på schemat där vyn ligger, inte på de scheman som vyns fråga läser från. Analytikerna kan alltså hämta data ur produkten medan driftstabellerna bakom den förblir stängda för dem. Evans räknar i sin tur databasscheman till de fysiska former som gränsen kring en bounded context kan ta.

Plattformens sida är en regel som gäller för alla domäner. Den är skriven som en SQL-fråga och körs vid varje release: Inga produktscheman får innehålla en vy utan beskrivning. `obj_description` returnerar kommentaren som är lagrad på ett objekt, och i `pg_class` betyder en `relkind` med värdet `v` att objektet är en vy:

```
-- gemensam regel: varje publicerad vy har en beskrivning
SELECT n.nspname, c.relname
FROM pg_class AS c
JOIN pg_namespace AS n ON n.oid = c.relnamespace
```

```
WHERE c.relkind = 'v'
  AND n.nspname ~ '_data$'
  AND obj_description(c.oid, 'pg_class') IS NULL;
-- varje rad som returneras stoppar releasen
```

Den uppdelningen är den federerade delen i miniatyr. Vad som räknas som en betald order bestämmer orderteamet. Att varje produkt beskriver sig själv och att namnen på produktscheman slutar på `_data` är gemensamma konventioner som plattformen kontrollerar utan att någon behöver komma ihåg dem.

Tjänstesidan Datastyrning och datakvalitet utlovar att varje siffra ska ha en känd ägare och en känd källa, och det är just vad en produktägare för data måste kunna redogöra för. Bland det levererade arbetet finns ett funktionsbaserat dataskikt i PostgreSQL, byggt så att all läsning och skrivning går genom funktioner som databasen kan ge behörighet till, och aldrig direkt mot tabellerna. Det är samma tankesätt som i vyn ovan: Konsumenterna får ett gränssnitt med en behörighet knuten till sig, inte tabellerna under.

Uppdraget En infrastrukturplan för 14 avdelningar beskrivs inte som ett data mesh, men det möter samma spänning som styrningsgruppen i ett data mesh lever med. Vid ett forskningsinstitut arbetade 14 forskargrupper med var sina datakällor, format, skalor och verktyg, och både den tekniska kompetensen och IT-resurserna varierade kraftigt: Ett fåtal hade byggt egna lösningar, medan många hade det svårt. Planen skulle låta forskarna ägna sig åt forskningen, och den vägde flexibilitet mot standardisering, med modulär arkitektur, integrationsvägar för mycket olika datakällor, användarvänliga gränssnitt för olika kunskapsnivåer, skalbar lagring och bearbetning och strategier för onboarding, support och styrning. Att avgöra vad varje grupp behåller för sig själv och vad alla delar är just den avvägning Dehghani lägger på den federerade styrningen, och att sänka kompetenskravet för en grupp är vad hon kräver av en självbetjäningsplattform.

Var det går fel

Ett centralt team med nytt namn. Det ändrar ingenting att kalla dataplattformsteamet ett data mesh om samma team fortfarande tar emot rådata från alla domäner, tvättar dem och ansvarar för kvaliteten. Dehghanis styrningstabell fungerar som checklista: Vem ansvarar för kvaliteten, vem bestämmer säkerhetsreglerna, och mäts framgång fortfarande i tabeller under styrning? Flytta ägarskapet för varje dataprodukt till domänen som producerar data, och låt det centrala teamet behålla plattformen och reglerna.

Produkter som ingen står för. En domän som exporterar tabeller utan ägare, beskrivning eller löfte om hur färska data är har bara flyttat det gamla problemet till sitt eget hörn. Dehghani vill att varje dataprodukt anger sin kvalitet som servicenivåmål, till exempel aktualitet och felfrekvens, och påpekar att en domän kan erbjuda två produkter med samma händelser: en nästan i realtid som kan tappa eller upprepa händelser, och en som kommer senare och är mer exakt. Utse en ägare för varje produkt, publicera målen och testa kvaliteten där data uppstår.

Inga gemensamma identifierare. Om varje domän identifierar samma saker på sitt eget sätt går inga två produkter att koppla ihop. I exemplet från 2019 ser en artist olika ut i domänen som registrerar uppspelningar och i domänen som betalar artisterna. Kom överens över domängränserna om hur sådana delade entiteter, som Dehghani kallar polysemer, ska identifieras, och lämna resten av modellen åt varje domän. Fältformat, adresskonventioner och gemensamma metadatafält hör hemma på samma gemensamma lista.

Varje domän bygger hela stacken på nytt. Utan en självbetjäningsplattform måste varje domän själv sätta upp lagring, datapipelines och övervakning, och den specialiserade kompetens som var en bristvara i ett enda centralt team måste nu finnas i vartenda team. Bygg plattformen innan domänerna ombeds publicera, och mät den på hur lite specialistkunskap en domän behöver för att leverera en produkt.

En produkt som ändras under konsumenterna. En dataprodukt är ett gränssnitt, och ändrar man den på plats går rapporterna som bygger på den sönder. I PostgreSQL avvisar `CREATE OR REPLACE VIEW` en del av det felet: Den nya frågan måste ge samma kolumner med samma namn, ordning och datatyper och får bara lägga till kolumner i slutet. En ändrad betydelse fångar den inte, för beräkningen bakom en kolumn kan mycket väl vara en helt annan. Lägg varje ändring som byter namn på, tar bort eller omdefinierar en kolumn i en ny vy, `orders_by_day_v2`, bredvid den gamla, och avveckla den gamla när konsumenterna har flyttat över.

Ett data mesh där det inte behövs. Fyra principer, ett plattformsteam, en federerad styrningsgrupp och en produktägare i varje domän är mycket maskineri. Mot slutet av artikeln från 2019 medger Dehghani att ”enterprise scale adoption still has a long way to go”. Räcker en handfull källor och ett team,

så behåll ett välfungerande centralt lager med en namngiven ägare för varje tabell, och låna de delar som lönar sig oavsett storlek: ägare, beskrivna datamängder och regler som maskinen kontrollerar.

Läs mer

- Data Mesh Principles and Logical Architecture — Zhamak Dehghani, 2020: de fyra principerna, delarna i en dataprodukt och tabellen som ställer central och federerad styrning mot varandra.
- How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh — den ursprungliga artikeln från maj 2019, med den centrala dataplattformens tre svagheter och de egenskaper en dataprodukt måste ha.
- Domain-Driven Design Reference — Eric Evans sammanfattning av mönstren, bland dem bounded context, längs vars gränser ett data mesh delar upp ägarskapet.
- PostgreSQL: CREATE VIEW — reglerna för att ersätta en vy och de privilegier som användarna av den behöver, så som exemplet använder dem.
- PostgreSQL: COMMENT — hur en beskrivning lagras på ett databasobjekt så att `obj_description` kan läsa tillbaka den.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med ett representativt dataset, frågorna det ska besvara och dagens flaskhals. Vi kan avtala mått på kvalitet, genomströmning och återställning.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är ett data mesh?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/data-mesh/