Engineer

arkitektur

hur ett systems delar passar ihop, och varför

Vad det är

Arkitektur, inom mjukvara ofta kallad mjukvaruarkitektur eller systemarkitektur, är de grundläggande begrepp och egenskaper som ett system har i sin omgivning och som kommer till uttryck i dess element, i relationerna mellan dem och i de principer som styr hur systemet utformas och förändras över tid. Formuleringen följer definitionen i NIST SP 800-160 Volume 1 Revision 1, Engineering Trustworthy Secure Systems, som publicerades i november 2022.

Varje led i definitionen har en uppgift. Elementen är delarna: tjänster, databaser, köer, frontenden och modulerna inuti dem. Relationerna är hur delarna beror av varandra och vad som passerar mellan dem. Principerna är de regler som avgör var nytt arbete hamnar, och det är de som håller element och relationer på plats när systemet växer. ”I sin omgivning” gör användarna, grannsystemen och begränsningarna till en del av arkitekturen. Att definitionen dessutom talar om förändring över tid betyder att en beskrivning som bara stämmer den dag den ritades är ofullständig.

Bara en del av alla designval räknas som arkitektur. Gruppen bakom formatet Architectural Decision Record (ADR) kallar ett arkitekturbeslut ett motiverat designval som hanterar ett funktionellt eller icke-funktionellt krav av arkitektonisk betydelse, och ett krav har den betydelsen när det har en mätbar effekt på systemets arkitektur och kvalitet. Mallen arc42 uttrycker det rakare: dess avsnitt om beslut är till för viktiga, dyra, riskfyllda eller omstridda beslut.

Eftersom en enda ritning aldrig rymmer allt beskrivs arkitektur i vyer. C4-modellen, som Simon Brown skapade, erbjuder diagram som fungerar som kartor du zoomar in och ut i: ett systemkontextdiagram som visar hur systemet står bland sina användare och de system som omger det, ett containerdiagram som visar applikationerna och datalagren inuti, ett komponentdiagram för en

enskild container och, där det tillför något, ett koddiagram. arc42 delar upp den skrivna beskrivningen i tolv avsnitt, från mål och begränsningar till beslut, risker och teknisk skuld.

Använd begreppet i ett beslut

Använd ”Vad är arkitektur?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Systemets form utarbetad på papper först, så att delarna passar ihop och kostnaden är känd innan någon skriver kod.

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. En infrastrukturplan för 14 avdelningar — 14 forskningsavdelningar under en och samma infrastrukturplan

En infrastrukturplan för 14 avdelningar Hör av er →

Varför det spelar roll

Det arkitekturen egentligen kräver av ett team är sällan ett val mellan färdiga strukturer. Frågan är vilka val ni gör nu, medvetet och skriftligt, och vilka ni låter stå öppna.

Avgör du en strukturfråga tidigt betalar du innan du vet särskilt mycket: valet görs med den minsta kunskap projektet någonsin kommer att ha, och de första veckorna går långsammare. Lämnar du den öppen avgörs strukturen ändå, en bekväm ändring i taget, av den som senast rörde koden. Utvägen är att skilja mellan två sorters val. Ett fåtal blir dyra att ändra så fort koden bygger på dem: hur systemet delas upp, vilken del som äger affärsreglerna, hur data nås och vilken ribba varje ändring måste klara. De förtjänar att avgöras tidigt och med avsikt. Resten går för det mesta billigt att ompröva och kan vänta tills koden visar vad den behöver.

Att skriva ned beslutet är dess andra halva, och det är där resonemanget överlever. En ADR fångar ett enda beslut med motivering, avvägningar och konsekvenser, och samlingen av ett projekts ADR:er utgör dess beslutslogg. ADR-gruppens webbplats ger Michael Nygards blogginlägg ”Documenting Architecture Decisions” från 2011 äran för att idén spreds. Priset är en sida text per beslut. I gengäld kan en senare läsare skilja en medveten avvägning från något som bara blev så, och kontrollera om skälet fortfarande håller.

Kvalitetsmålen avgör mellan strukturer som alla skulle fungera. arc42 ber om de tre viktigaste kvalitetsmålen för arkitekturen, högst fem, alltså dem som de viktigaste intressenterna bryr sig mest om, och påpekar att de är något annat än

projektets mål. Två lösningar med exakt samma funktioner kan bete sig helt olika när de går sönder, när lasten växer och när de ska ändras, och det är kvalitetsmålen som säger vilken skillnad som räknas.

Så ser det ut i praktiken

Plattformens repo skapades den 25 november 2025 och var helt tomt, och sidan om dess grundbeslut konstaterar att sådana val är ”billiga att göra dag tre och nästan omöjliga att ändra på vid månad sex”. Två av de tre beslut som fattades då formade strukturen. Stacken delades upp i lager så att varje del har en enda uppgift: en frontend i Next.js, ett Go-API som validerar och skickar vidare och ett funktionslager i PostgreSQL som äger affärsreglerna. Dataåtkomsten lades bakom lagrade funktioner från början, eftersom det i efterhand ”hade inneburit att varje hanterare skrevs om”. Nio månader och flera tusen commits senare har lagren enligt sidan inte flutit ihop, och ingen hanterare pratar direkt med en tabell. Sidans prov på ett grundbeslut är inte om det lät rätt, utan om det överlevde den arbetsmängd som kom efteråt.

Struktur som ingen bestämmer blir bestämd ändå, av en slump, och sidan om GIS-kartan visar hur det går till. Kartan hade vuxit fram i bitar: spatiala data i PostgreSQL, själva ritandet i Mapbox, ReactJS i frontend och NodeJS bakom. Varje del fungerade för sig, men de ”hade bara inte byggts för att passa ihop”. Rollen omfattade fullstackutvecklingen och den tekniska riktningen, och arkitekturen lades fast först. Därefter flyttades ansvar enligt sidan ”gång på gång tillbaka till det lager där det hörde hemma så fort ett lager började läcka in i nästa”, och en stor del av arbetet gick ut på ”att fånga de små inkonsekvenserna innan de stelnade till arkitektur”.

Sidan om containerskiktet behandlar det nedskrivna skälet som det egentliga resultatet. Branschens standardmotor för containrar publicerar containerportar ovanför reglerna i värdens brandvägg, så skiktet blev en containermotor utan daemon som driver enheter genererade av operativsystemets egen tjänstehanterare, helt utan publicerade portar. En avvägning står uttryckligen som avvägning: enheterna körs på systemnivå snarare än rootless, vilket håller automatiseringen enkel till priset av den striktare isolering som rootless skulle ge, och anteckningen säger åt vilket håll man ska gå om containerutbrytning någon gång blir viktigare. Det är precis vad en ADR rymmer, ett beslut med motivering, avvägningar och konsekvenser. Sidan värderar att valet skrevs ned med sitt skäl högre än själva valet, eftersom ”nästa person kommer att få höra att standarden ska användas i varenda artikel som den personen läser”.

Var det går fel

En noggrann nykomling river upp ett beslut som var medvetet. Skälet fanns bara i någons huvud, så valet ser ut som ett misstag och den standard som varje guide rekommenderar ser ut som rättelsen. Skriv en ADR för varje beslut som är dyrt att riva upp: sammanhanget, alternativen som vägdes, det som valdes, konsekvenserna och vad som skulle göra det värt att ompröva. Lägg ADR:erna bredvid koden, så att en ändring i det ena granskas tillsammans med det andra.

Affärsregler dyker upp i tre lager samtidigt. Varje ändring lade sin logik där det råkade passa just den eftermiddagen, tills samma regel kontrollerades i frontenden, i API:et och i databasen, lite olika på varje ställe. Bestäm vad varje lager äger, skriv ned det och flytta tillbaka ett ansvar så fort det läcker, medan den vilsekomna kopian fortfarande är liten. Där plattformen tillåter det kan gränsen dessutom upprätthållas tekniskt, t.ex. genom att API:et får anropa databasfunktioner men saknar rättigheter till tabellerna.

En regel som införs vid månad sex blir ett städprojekt. Beslutet som allt vilar på lämnades öppet, och varje commit sedan dess har byggt på det som råkade gälla. Avgör de få dyra frågorna först och lämna resten öppna med avsikt. Skriv det i beslutsloggen, så att en öppen fråga syns som öppen i stället för att tyst besvaras av nästa ändring.

Arkitekturdiagrammet förklarar ingenting för någon. En enda ritning blandar användare, tjänster, klasser och tabeller, och linjerna mellan rutorna saknar etikett eller bär bara ordet ”uses”. Rita ett diagram per zoomnivå, som C4 gör, och ge varje linje en etikett som anger relationens riktning och syfte; mellan containrar ska tekniken eller protokollet också stå. C4:s råd om notation ber om just det och avråder från etiketter på ett enda ord, som ”Uses”.

Läs mer

- architecture, NIST CSRC Glossary — definitionen som används ovan, bredvid FIPS-definitionen och definitionen av säkerhetsarkitektur, som delar samma uppslag.
- NIST SP 800-160 Vol. 1 Rev. 1 — Engineering Trustworthy Secure Systems, publikationen som definitionen kommer från.
- Architectural Decision Records — vad som gör ett beslut arkitektoniskt, mallar för att skriva ned ett och verktyg som hjälper till.

- The C4 model — fyra nivåer av diagram, från systemkontext ned till kod, med råd om notation och en checklista för granskning.
- arc42 — praktiska tips och exempel för vart och ett av de tolv avsnitten i arc42-mallen.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är arkitektur?” 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/architecture/