

Vad betyder egentligen arkitektur?

Sten Sundblad

Det här är en fråga vi ofta stöter på, och det är långt ifrån alltid svaren är som de borde. Många förväxlar arkitektur och teknisk design och talar om en lösnings tekniska utformning som om den vore arkitektur. Man kan tycka att det spelar väl inte så stor roll om man vet vad arkitektur är, bara slutresultatet blir gott, men så enkelt kanske det ändå inte är.

Man talar mer och mer om att IT-lösningar måste vara adaptiva – de måste lätt (hick!) kunna anpassas till skiftande yttre omständigheter, vilket innebär att arkitekturen måste ändras. De måste också kunna effektiviseras utan att användare skall behöva ändra sitt beteende, vilket innebär att arkitekturen förblir oförändrad. För att uppnå det är det viktigt att tydligt hålla isär det som är arkitektur från det som är teknisk design.

Ett armbandsurs arkitektur

När man talar om *klassisk* eller *traditionell* arkitektur är det ofta uppenbart vad som är arkitektur och vad som är teknisk design. Ett föremåls arkitektur är vad som kan upplevas från en position som är extern i förhållande till föremålet. Den avgör hur man *upplever* föremålet, hur man kan *interagera* med det, och hur det kan interagera med andra föremål. Föremålets tekniska design bestämmer hur dess arkitektur skall implementeras. Dess arkitektur utgör begränsningar som den tekniske designern måste hålla sig inom, samt krav som den tekniska utformningen måste uppfylla.

Fredrick P. Brooks Jr är den förste vi har hört – eller egentligen läst – som har talat om en mjukvaras arkitektur. Han ledde i början på 1960-talet det då största projektet för utveckling av mjukvara någonsin, utvecklingen av operativsystemet för IBM's stordator 360. I boken *The Mythical Man-Month*¹ citerar han kollegan Gerrit Blaauw som säger att (i min översättning) "där arkitektur talar om *vad* som händer berättar implementeringen *hur* det fås att hända".

Blaauw tar som exempel ett så enkelt föremål som en klocka och påpekar att arkitekturen för en sådan består av urtavlan, visarna och uppskrivningsknappen medan implementeringen beskriver vad som händer *inuti* boetten.

Jag sitter just nu och tittar på mitt eget, se Figur 1, som jag har lagt på bordet framför mig. Det är försett med ett läderarmband i någon slags

ormskinn. Jag förutsätter att det inte är någon utrotningshotad art som bidragit med råvaran och tycker att det är ganska elegant och bidrar till det sobra intryck som jag tycker att uret ger. Eftersom armbandet bidrar till min upplevelse av att äga och använda uret ingår det i urets arkitektur.



Figur 1 – Stens armbandsur

inte så mycket till adaptivitet, men alltid något.

När armbandet blir slitet, eller om jag helt enkelt tröttnar på det, kan jag byta ut det. Om jag väljer ett identiskt armband förändrar jag därmed inte urets arkitektur, men om jag byter till ett helt annat slag av armband gör jag faktiskt det. Detta ger urets arkitektur ett drag av adaptivitet – jag kan förändra den utan att påverka den tekniska utformningen. Kanske

Uret är, som de flesta idag, batteridrivet. Det innebär att jag aldrig behöver dra upp klockan, men i stället att jag ibland måste be urmakaren byta batteri åt mig. Den kräver alltså ett annorlunda beteende av mig som användare än en uppdragbar variant skulle göra. Därför ingår även denna egenskap till urets arkitektur trots att jag inte kan urskilja det med mina ögon. Men jag kan uppleva det, dels genom att jag inte behöver dra upp det och dels genom att jag ibland måste byta batteri.

Jag tycker själv att urtavlans design är tilltalande, så jag trivs med den. Hade den varit större och

klumpigare hade jag säkert vantrivts med den, vilket kanske hade påverkat mitt humör negativt. Den kräver inte heller särskilt mycket putsande, så materialet verkar vara väl valt. Jag misstänker att den inte är vattentät, så jag tar av mig den innan jag lägger mig i badkaret. Alla dessa egenskaper ingår i urets arkitektur, eftersom de antingen påverkar min upplevelse av att äga och använda uret eller påverkar mitt *sätt* att använda det.

Urverket ingår däremot *inte* i klockans arkitektur – det är tillverkarens eller urmakarens *implementering* av urets arkitektur.

Krav på tillförlitlighet, till exempel vad avser urets förmåga att visa korrekt tid, ingår i ett föremåls arkitektur. Uppfyllandet av sådana krav upplevs via urtavlan, som är ett av urets gränssnitt.

Under alla de år jag ägt detta ur har det tillförlitligt visat tiden, så det valda urverket har visat sig vara en god implementering av urets arkitektur. Men antag att så inte varit fallet. Då hade det valda urverket utgjort en dålig implementering av arkitekturen. Men även det problemet hade gått att lösa. En urmakare hade helt enkelt kunnat byta ut antingen delar av urverket eller hela urverket för att lösa problemet. Ett lyckosamt sådant byte hade inneburit att implementeringen hade ändrats medan arkitekturen förblivit oförändrad.

Urets arkitektur kan alltså upplevas från en position på utsidan om uret medan dess implementering är inkapslad inne i själva uret. Man kan alltså se uret som en svart låda, även om dess färgskala är vitt, svart och guld. Det man kan uppleva från utsidan är klockans gränssnitt mot mig som användare.

Serviceorienterad arkitektur

Du kan se även en IT-lösning som en svart låda, vars funktionalitet kan utnyttjas via ett eller flera gränssnitt. I en tjänsteorienterad lösning finns det i varje fall två nivåer av sådana gränssnitt.

Den ena nivån är själva användargränssnittet som låter slutanvändaren interagera med lösningen. Den andra nivån ligger på de enskilda tjänster som utgör lösningen, där varje tjänst dels har ett eller flera gränssnitt, där varje gränssnitt har ett av både tjänsten och konsumenten känt

kontrakt, dels har en implementering av dessa gränssnitt och dessa kontrakt.

Om du förändrar ett gränssnitt, eller ett gränssnitts kontrakt, då förändrar du tjänstens arkitektur. Jag kan påstå det eftersom varje sådan ändring påverkar konsumentens beteende eller upplevelse av att använda tjänsten. Om du låter gränssnitt och kontrakt vara oförändrade och "bara" ändrar i tjänstens insida, då är det inte en arkitektonisk ändring utan en ändring av implementeringen.

Det handlar alltså inte om att stora ändringar skulle vara arkitektoniska och att små ändringar inte skulle vara det. Om du i ett datakontrakt ändrar ett enda fälts datatyp från integer till string, då är det en arkitektonisk förändring eftersom den direkt påverkar konsumenten. Om du ändrar från att i tjänstens insida till exempel byta ut hela din domändrivna modell mot att låta data representeras av datakontraktens objekt, då är det *inte* en arkitektonisk ändring eftersom den inte påverkar konsumentens beteende. Så den pyttelilla ändringen är i detta fall arkitektonisk medan den rejält stora ändringen inte är det.

Mot detta kan man invända att ett felaktigt gjort utbyte av insidans dataobjekt kan komma att påverka konsumenten eftersom det som fungerade igår då kanske inte fungerar idag. Men i så fall är det ju bara så att utbytet av implementering var misslyckat. Arkitekturen förblir oförändrad, men implementeringen fungerar inte längre.

Vari ligger värdet?

Man kan fråga sig vari värdet ligger av att skilja på arkitektur och implementering. Båda behövs ju ändå!

Värdet ligger i att det som påverkar konsumentens beteende och upplevelse – det vill säga arkitekturen – bör avhandlas med uppdragsgivaren, medan det som *inte* påverkar konsumentens beteende och upplevelse bör hållas *ifrån* uppdragsgivaren. Detta gäller både när tjänsten skall tillverkas från början och när du skall vidareutveckla en befintlig tjänst.

Det finns två skäl till att undvika att diskutera implementeringsfrågor med uppdragsgivare. Det

första är att det inte tillför diskussionen något som är av värde för denne. Det han eller hon behöver veta är "vad måste vi göra" och "vad får vi i utbyte".

Det andra skälet är att du själv som implementatör av en tjänst vill ha full frihet att implementera tjänsten på det sätt som passar dina ändamål bäst. Det finns ingen poäng för dig i att diskutera tekniska implementeringsfrågor med en uppdragsgivare som både kan antas ha begränsad insyn i din tjänst totala uppdrag och i de tekniska faktorer som begränsar din frihet att välja utformning av implementeringen, och som kan antas ha begränsat intresse av dem.

Ibland är det så att en tjänst är implementerad på ett sådant sätt att den inte uppfyller sina kontrakt. Då måste du antingen ändra implementeringen eller få tjänstens ägare att gå med på att kontraktet ändras. I båda fallen är det av värde att veta vad som ingår i tjänstens arkitektur och vad som utgör dess implementering, det vill säga var gränsen går mellan dessa storheter.

Så mitt råd är att du skall vara noga med att dra denna gräns på ett bra ställe. Dokumentera sedan arkitektur för sig och implementering för sig. Du kommer att finna att det lönar sig bra att göra det. Om du själv är både arkitekt och teknisk designer (som vi gärna kallar för mjukvaruingenjör), då kan det bli litet svårare att dra gränsen, men du bör ändå göra det. Du kan ju alltid trösta dig med att alla diskussioner mellan arkitekt och ingenjör är fruktbara, angenäma och problemfria.

ⁱ Frederick P. Brooks, Jr. The Mythical Man-Month, Addison-Wesley. ISBN 0-201-83595-9