Engineer

Programvare

programmene og dataene en datamaskin kjører

Hva det er

Programvare er alt i et datasystem som kan endres ved å skrive i stedet for å lodde: programmene, opplysningene de bærer med seg, og innstillingene som bestemmer hvordan de oppfører seg. Ordet brukes nesten bare i entall, så én enkelt enhet kalles gjerne et program, og det en person åpner for å få gjort en jobb, heter en app eller en applikasjon. Det engelske «software» er også vanlig på norsk, «kode» betyr som regel kildekoden en utvikler redigerer, og i spesifikasjoner blir det hele noen ganger forkortet til SW.

Det vanlige motstykket er maskinvare, selve den fysiske maskinen, med fastvare, eller firmware, i mellomrommet. NIST-ordlisten trekker grensen ved endring: programvare er dataprogrammer og tilhørende data som kan skrives eller endres mens de kjører, mens fastvare er den typen som ikke kan endres under kjøring. Operativsystemet på en bærbar datamaskin er programvare, koden i tastaturkontrolleren er fastvare, og selve tastaturet er maskinvare.

Programvare har lag. Systemprogramvare, f.eks. et operativsystem eller en enhetsdriver, styrer maskinen. Brukerprogramvare gjør noe for en person: et regneark, en bestillingsløsning på nett, en meldingsklient. Mellom de to ligger biblioteker og programvarepakker, ferdige deler som begge typene settes sammen av, og i et moderne produkt har det meste av koden i praksis kommet den veien heller enn å bli skrevet for produktet.

Ett skille til avgjør mye senere. Kildekode er det mennesker skriver og leser; en kompilator eller en tolk gjør den om til instruksjoner en prosessor utfører. Begge deler er programvare, men de kan ikke brukes om hverandre når du kjøper programvare. En leveranse med bare kompilerte filer er noe bare leverandøren kan endre.

Bruk begrepet i en beslutning

Bruk «Hva er programvare?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Formen på et system tegnet opp på papiret først, slik at delene passer sammen og kostnaden er kjent før noen skriver kode.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Én infrastrukturplan for 14 avdelinger — 14 forskningsavdelinger på én felles infrastrukturplan

Én infrastrukturplan for 14 avdelinger Ta kontakt →

Hvorfor det er viktig

Den som bestiller, kjøper eller leier programvare, velger samtidig hvem som skal bære vedlikeholdet. Det som gjør programvare fristende, at en kopi koster ingenting og at en endring etter levering er billig sammenlignet med å bygge om en maskin, er også grunnen til at den aldri blir ferdig. Komponentene den er bygd av, slipper rettelser. Operativsystemer og nettlesere fjerner grensesnitt. En svakhet blir offentliggjort i et bibliotek ingen på teamet husker å ha valgt. Hver av dem blir arbeid for den som eier koden.

De vanlige veiene fordeler det arbeidet ulikt:

- **Leid som tjeneste.** Leverandøren står for oppdateringene og serverne. Du arver utgivelsesplanen og prisene deres og bryderiet med å komme deg ut, og dataene dine ligger hos dem.
- **Lisensiert og driftet selv.** Du bestemmer når det skal oppgraderes, og du gjør også selve oppgraderingen, sikkerhetskopieringen og sikkerhetsoppdateringene.
- **Bygd for deg.** Den passer prosessen i stedet for omvendt, og vedlikeholdsbudsjettet er ditt så lenge den er i drift. Kostnaden som avgjør om det var klokt, kommer etter lanseringen.
- **Åpen kildekode.** Under en lisens som oppfyller Open Source Definition, kan du bruke, studere, endre og videreformidle den. Noen må fortsatt legge inn oppdateringene, og gratis å skaffe er en annen egenskap enn gratis å drifte.

Uansett vei er to spørsmål om resultatet verdt å stille før du skriver under. Hva får du til slutt: kildekoden, instruksjonene for å bygge den og en liste over hva den inneholder? Og hvordan skal leverandøren holde den sikker? NISTs Secure Software Development Framework finnes delvis for den andre samtalen, siden det gir kjøpere og leverandører et felles språk for utviklingspraksis. For den første definerer en rapport fra NTIA en stykkliste for programvare, på engelsk

software bill of materials eller SBOM, som en formell oversikt over komponentene programvaren er bygd av, med opplysninger om hver av dem og om hvordan de henger sammen i leverandørkjeden. Å be om en slik liste er en billig måte å se hvor mye av det du betaler for, som er skrevet av noen andre.

Slik ser det ut i praksis

Å vite hva som faktisk ble levert, er den delen team hopper over. På én plattform ble sikkerhetsarbeidet flyttet fra skjønn og over i bygget: i et løpende sikkerhetsprogram lages en stykkliste for programvaren ved hver bygging med Anchore og Syft, Dependabot overvåker 6 økosystemer, CodeQL og ZAP analyserer koden og den kjørende instansen, og hver GitHub Action fra tredjepart er låst til en commit-hash i stedet for en tagg. Ved siden av automatiseringen ligger 21 skriftlige sikkerhetsrevisjoner, bl.a. en trusselmodell. En sårbarhet som de konfigurerte sjekkene gjenkjenner, kan få byggingen til å feile. Siden er like ærlig om kostnaden: mengden funn krever en triagering som aldri tar slutt.

Å få programvare fra en endring og ut i produksjon er den andre halvdelen. Der utgivelser hadde vært avhengige av at noen husket rekkefølgen på en serie manuelle steg og gjorde det litt annerledes hver gang, sparte automatiserte leveransepipeliner for bygging, testing og produksjonssetting anslagsvis 4 000 timer og gjorde utgivelsene både raskere og mer pålitelige. Gevinsten kom av at veien var den samme hver gang, og ikke av at noen var mer forsiktige.

På et reparasjonsbord er grensen mellom programvare og maskinvare helt konkret. Reparasjoner av bærbare datamaskiner omfattet skjermer, hengsler, tastaturer og hovedkort, og programvareproblemer på tvers av Linux, Windows og Mac. En bærbar som ikke vil starte, kan skyldes laderen, hovedkortet eller batteriet, så arbeidet begynte med å finne den virkelige feilen før noe ble rørt.

Hvor det går galt

Et sårbarhetsvarsel som blir en uke med utgraving

En sårbarhet blir kunngjort i en mye brukt komponent, og ingen kan si hvilke av organisasjonens systemer som inneholder den. Oversikten har bare noen gang ligget i hodene på folk og i låsefiler som ingen leser, så den må bygges opp igjen under press, system for system. Lag en stykkliste som en del av hver bygging, og ta vare på den sammen med utgivelsen den beskriver. Spørsmålet «er vi rammet?» blir da et søk.

Utgivelser som avhenger av hvem som gjør dem

Produksjonssettingen virker når én person kjører den, og feiler når en annen gjør det, og forskjellen viser seg å være et steg som bare finnes i noens hukommelse. Hver utgivelse blir en liten improvisasjon. Legg hele veien inn i en leveransepipeline som bygger, tester og produksjonssetter på samme måte hver gang, og behandle hvert manuelle steg som er igjen, som en feil som skal bort.

Et budsjett som slutter ved lansering

To eller tre år etter lansering føles hver oppdatering risikabel, rammeverket ligger flere hovedversjoner bak, og det eneste forslaget på bordet er å skrive alt på nytt. Programvaren ble betalt som et bygg og deretter latt være i fred, mens alt rundt den fortsatte å bevege seg. Sett av en fast sum til vedlikehold, oppgrader avhengighetene i små, jevne steg mens hvert steg fortsatt er lite, og regn med at plattformene under endrer seg etter sin egen timeplan.

Et navn som står stille mens koden bak det endrer seg

Det samme bygget gir forskjellige resultater på to dager uten en eneste commit imellom. Et sted peker en referanse mot et bevegelig mål: et versjonsintervall, en `latest`-tagg, en CI-handling det vises til med en tagg som eieren kan flytte. Lås kode fra tredjepart til en eksakt versjon, eller til en commit-hash der tagger kan flyttes, commit låsefilen, og la en bot foreslå oppdateringene, slik at hver endring kommer som en diff som kan gjennomgås.

Les mer

- NIST CSRC Glossary: software — definisjonene NIST-publikasjonene bruker, også den som skiller programvare fra fastvare.
- NIST SP 800-218, Secure Software Development Framework — et kjernesett med praksiser for å bygge programvare med færre sårbarheter, skrevet slik at kjøpere kan bruke det overfor leverandører.
- NTIA: The Minimum Elements for a Software Bill of Materials — hva en liste over komponentene i programvare må registrere for å være til nytte.
- The Open Source Definition — de ti kriteriene en lisens må oppfylle før programvare som gis ut under den, regnes som åpen kildekode.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er programvare?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/software/