2023-02-23

INSTITUTIONEN FÖR TILLÄMPAD IT

Öppen källkod i offentlig digitalisering

Analys av Open ePlatform ur ett öppet källkodsperspektiv

Per Persson

Johan Magnusson

Forskningskonsortiet för Digital förvaltning

Swedish Center for Digital Innovation

Institutionen för Tillämpad IT

Göteborgs universitet

SCDI/Forskningskonsortiet för Digital Förvaltning
Forskningsgången 6, Box 100, 405 30 Göteborg
031 786 00 00
www.digitalforvaltning.se

Sammanfattning

Digitalisering i offentlig sektor har i huvudsak realiserats genom etablering av vad som benämns ”e-tjänster”, det vill säga digitala versioner av existerande tjänster med varierande sofistikation. Den kommunala sektorn har till stor del byggt mycket av sitt existerande e-tjänsteutbud på en gemensam plattform, Open ePlatform. Givet avsaknaden av tillräcklig styrning har denna plattform ej underhållits och utvecklats i linje med teknikens utveckling, varvid den nu i praktiken är att betrakta som en proprietär produkt och inte den lösning i öppen källkod som var ambitionen. Rapporten beskriver möjligheterna och kraven associerade med tilltagande användning av öppen källkod i offentlig digitalisering, samt de brister som föreligger i Open ePlatform. Rapporten föreslår insatser för att dels kortsiktigt säkerställa kontinuiteten i den befintliga plattformen och dels långsiktigt säkra fortsatt ändamålsenlig digitalisering bland kommuner genom öppen källkod.

Innehållsförteckning

1. Bakgrund 4
2. Öppen källkod i offentlig sektor 6
Fokusera på mjukvaran 7
Sätt rätt organisation 7
Var aktiv 8
3. Resultat 9
3.1. Plattformen Open ePlatform 9
3.2. Hinder för leverans av kod till Open ePlatform 9
4. Analys och rekommendationer 12
4.1. Rekommendationer 12
4.2. Förslag – etablera OpenMunicipality 13
5. Avslutande kommentarer 15
Bilaga A: Inventering av teknisk skuld i Open ePlatform 16
Kod 16
Design och arkitektur 16
Utvecklings- och driftmiljö 17
Kunskapsspridning och dokumentation 17
Test 18

1. Bakgrund

För att möjliggöra omställning av offentlig sektor lyfts digitalisering[1] fram som ett alternativ. Det digitala möjliggör en helt annan skalning och spridning än tidigare välfärdstjänster, samt öppnar upp för en ny logik i termer av hur offentlig förvaltning bör fungera. Centralt för den nya logiken är öppenhet, transparens, proaktivitet och invånarfokus[2].

För att möjliggöra digitalisering har offentliga organisationer framgångsrikt etablerat vad man kallar ”e-tjänster”, dvs tjänster som ersätter och/eller kompletterar tidigare traditionella tjänster levererade av offentliga aktörer. Dessa tjänster kan vara av varierande grad av sofistikation, från enkla formulär som endast sparas som pdf till mer avancerade tjänster där till exempel bygglovsprocessen helt flyttas över till en digital kanal.

För att möjliggöra snabb etablering av e-tjänster nyttjar offentliga aktörer olika typer av digital infrastruktur. Vanligt förekommande är olika typer av plattformar som agerar mellanprogramvara och möjliggörare av snabbare utveckling än om tjänsterna skulle utvecklas rakt in mot existerande system. Inom kommunal sektor är Open ePlatform vanligast förekommande, med c:a 180 användande kommuner i dagsläget. Genom att nyttja en gemensam plattform möjliggörs återanvändning av utvecklade e-tjänster i en kommun i andra kommuner, lärande av varandra och enklare inlemmande av nya kommuner som önskar nyttja plattformen som bas för att återanvända och utveckla e-tjänster.

Open ePlatform är en lösning byggd som öppen källkod. Öppen källkod innebär att programvaran är fri att använda, undersöka och ändra, samt distribuera vidare givet licensens specifika villkor. Ingen enskild organisation har med andra ord ägarskap över plattformen som helhet (skaparen till koden har sin upphovsrätt, men ger obegränsad rätt till användning av densamma under licensens villkor), och ansvarsfrågan om plattformens funktionalitet och livscykel faller antingen på den organisation som ”tar hem” och nyttjar den eller den leverantör som paketerar ett erbjudande runt den.

Tidigare forskning visar på både styrkor och svagheter i användningen av öppen källkodslösningar för kritisk infrastruktur[3]. Styrkorna innefattar att man genom att dela plattformen med andra får till en gemensam utveckling och ansvarstagande för lösningens fortsatta funktionalitet. Svagheterna visar på risker relaterade till att just förvaltningsrelaterade insatser kan glömmas bort (att man inte avsätter nödvändiga resurser för detta) i takt med att resurser flyttas över till att utveckla på toppen av plattformen för att säkerställa direkt verksamhetsnytta.

Denna rapport är skriven på uppmaning av kommunala intressenter direkt beroende av Open ePlatform för tillhandahållande av sina e-tjänster. Studien bygger på tidigare genomförda tekniska analyser av Open ePlatform[4], intervjuer med enskilda utvecklare som tidigare lyft de tekniska problemen i olika

[1] Digitalisering definieras som ”en metod för verksamhetsutveckling där digitala lösningar används för automatisering eller innovation” i linje med Forskningskonsortiet Digital förvaltning.

[2] https://www.oecd.org/governance/the-oecd-digital-government-policy-framework-f64fed2a-en.htm

[3] https://aisel.aisnet.org/jais/vol11/iss11/3/ och https://link.springer.com/article/10.1007/s12525-022-00557-9

[4] En rapport från 2019 av Dewire som ämnade gå in som utvecklare av funktionalitet i plattformen, men som avskrev detta efter att ha bekantat sig med ramverk och kod, samt en rapport från Sundsvalls kommuns utvecklingsavdelning från 2022 som också tänkt sig bidra till utvecklingen men kom till samma slutsats som Dewire.

forum samt analys av dokumentation. I rapporten genomförs en metaanalys av tidigare analyser av plattformen utifrån perspektivet teknisk skuld som hindrar en breddning av utvecklingen av plattformen och skapar kontinuitetsrelaterade riskerar vid eventuellt frånfälle av nuvarande leverantör.

2. Öppen källkod i offentlig sektor

Öppen källkod har länge lyfts som en möjlig del av lösningen för en tilltagande digitaliserad offentlig sektor. En EU-rapport[5] från 2020 lyfter nedan lärdomar efter att ha studerat hur 28 länder adresserar öppen källkodsfrågan:

- Betydelsen av programvara i öppen källkod inom offentlig sektor över hela Europa bekräftas av att Europeiska regeringar alltmer införlivar programvara i öppen källkod som en del av sitt lands politiska och juridiska ram, där 26 av de 28 studerade länderna har infört rättsliga och politiska initiativ med hänvisning till programvara i öppen källkod. Det vanligaste är att policys och lagstiftning för öppen källkod är inbäddade i bredare digitaliseringsinitiativ inom dessa länder.
- Av de 100 initiativ som identifierades i rapporten fokuserade 25 politiska och 6 juridiska initiativ uteslutande på öppen källkod (antaget av 19 länder) medan de återstående 50 politiska och 19 juridiska initiativen hänvisar till öppen källkod som en del av ett bredare digitaliseringsinitiativ.
- Initiativ runt öppen källkod fanns i alla europeiska länder, även i de som saknade en specifik policy och rättslig ram för öppen källkod, eller en statlig aktör som arbetade för att främja öppen källkods status i deras land. De flesta initiativ leds dock av länder med antingen politiska och rättsliga ramar som uttryckligen adresserar öppen källkod, eller statliga organ som har inkluderat öppen källkod i deras mandat.
- Programvara i öppen källkod är ett av flera alternativ för att driva digitalisering, men fördelarna för användarna i form av transparens, anpassningsförmåga och samarbetspotential positionerar öppen källkod som ett mycket attraktivt erbjudande tillgängligt för hela offentlig sektor. EU-rapporten belyser effekterna av det gradvisa införandet av lösningar i öppen källkod över hela Europa, som kulminerar i genomförandet av olika öppen källkodsinitiativ och inrättandet av öppen källkodsorgan.

Följande design principer sammanfattar hur ett öppet källkodsprojekt bör utformas[6]:

- Uppmuntra snarare än att bekämpa decentralisering
    - Öppen källkod är avsedd att distribueras; det är en del av det som gör det så effektfullt
    - Utnyttja det öppna förhållningssättet som en styrka snarare än centralisera styrningen
- Arbeta nära andra befintliga öppen källkodsprojekt
    - Öppen källkodsprojekt är aktiva, sammansvetsade och kommunikativa – behandla detta som en tillgång snarare än att fatta beslut bakom stängda dörrar
    - Framträdande nyckelpersoner verkar som kanariefåglar i kolgruvan när något behöver omhändertas – utnyttja detta
- Överväg ett holistiskt synsätt på projektstöd
    - Öppen källkodsprojekt behöver mer än bara kod eller pengar, och ibland behöver de varken eller – långsiktigt support handlar mer om att skapa tid än om pengar
    - Kodgranskning, teknisk dokumentation, testning, opinionsbildning och evangelisation är alla viktiga aspekter på ett öppen källkodsprojekt
- Hjälp utvecklare och förvaltare att planera i förväg

[5] https://joinup.ec.europa.eu/sites/default/files/inline-files/OSOR_Status%20of%20OSS%20Policies%20in%20Europe_2020_0.pdf

[6] https://www.fordfoundation.org/media/2976/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure.pdf

    - Insatser för att stödja digital infrastruktur tenderar att vara reaktiv och ad hoc och bör försöka undvikas – förutom redan pågående projekt kan det finnas nya projekt som behöver stödjas
    - För befintliga projekt kommer förvaltare att ha stor nytta av att kunna planera för de kommande tre till fem åren, inte bara sex månader till ett år
- Se möjligheter, inte bara risker
    - Modern öppen källkodsförvaltning handlar inte bara om att förhindra incidenter (till exempel säkerhetsincidenter), utan snarare om att ge fler människor möjlighet att bygga fler saker – detta koncept är kännetecknande för dagens öppen källkodskultur och hjälper också till med att bygga upp ett starkt stöd för ditt projekt
    - Fundera på hur du kan inkludera fler människor med olika bakgrund, färdigheter och förmågor i din strategi, snarare än att begränsa arbetet till att bara gynna befintliga deltagare

För att lyckas med en etablering av ett väl fungerande öppen källkodssamarbete inom offentlig sektor måste dessutom ett antal specifika framgångsfaktorer beaktas, nedan sammanfattat genom EU-organet ISA´s mest signifikanta riktlinjer[7].

Fokusera på mjukvaran

Mognadsgraden och kvalitén för den mjukvara som levereras är avgörande för samarbetets hållbarhet över tid och är den grund som samarbetet bygger på. Det finns flera element som behöver övervägas innan och när man släpper och underhåller öppen källkod.

- Välj rätt licensform för den öppna källkoden
- Välj rätt programmeringsspråk och ramverk
- Etablera en leveransprocess
- Prioritera kvalité framför kvantitet – automatisera test samt kodgranskning i leveransprocessen
- Dokumentera all mjukvara
- Beakta GDPR
- Exponera all kod öppet (på GIThub) redan vid första kod-raden

Sätt rätt organisation

Med tanke på offentlig sektors hierarkiska natur blir det särskilt viktigt att organisera samarbetet på ett sätt som främjar att öppen källkodsprojekts framgång och hållbarhet tydligt exponeras i alla nivåer. Tydlig styrning och operativ vägledning är därför en förutsättning för att samarbetet skall fungera väl och kunna växa fritt.

För att organisationen ska fungera smidigt krävs en tydlig rollfördelning och besittning av nyckelroller

- Programnivå
    - Styrelse - har det övergripande strategiska ansvaret och ska säkerställa en positiv utveckling av samarbetet
    - Affärsansvariga – ansvarar för drift, samordning mellan projekt och styrelse, leverantörsdialoger, projektportföljen samt kommunicerar med medlemmar och

[7] https://joinup.ec.europa.eu/sites/default/files/inline-files/2021%20Updated%20Guidelines%20for%20creating%20sustainable%20OS%20communities_1.pdf

intressenter, hanterar projektmodeller och mallar och organiserar och deltar i konferenser

- Projektnivå
    - Styrgrupp – nyckelpersoner som ansvarar för samarbetet och tar beslut om funktioner, leveranser och andra aktiviteter samt fungera som en bro mellan samarbetet och de offentliga förvaltningarnas politiska hierarki.
    - Teknisk kommitté – en teknisk ledningsgrupp som ansvarar för att verifiera och godkänna föreslagna ändringar av koden samt fatta de slutliga besluten om projektens utveckling tillsammans med projektledare.
    - Förvaltare – medlemmar i samarbetet som ansvarar för att underhålla och hantera vissa aspekter av projekten (t.ex. säkerhet). Medlemmar som har en stark känsla för ansvar och riktning är bäst lämpade för denna roll.
    - Committers (”Levererare”) – Medlemmar som har visat hängivenhet och är frekventa bidragsgivare kan med tiden erkännas som committers. De kan också ansvara för att granska nya kodbidrag från andra medlemmar.
    - Bidragsgivare – alla utanför kärnteamet i organisationen som levererar kod, deltar i forum, kommenterar frågor, anordnar evenemang eller är aktiv på annat sätt.

Dessutom bör vissa medlemmar av kärnteamet även ansvara för att främja samarbetet och ansvara för kommunikation, marknadsföring och hantering av sociala medier. Organisationen bör verka fullt ut transparant och säkerställa:

- En tydlig vision och mission
- Tydliga riktlinjer
    - Uppförandekod
    - Roller och ansvarsområden
    - Arbetssätt (agilt)
    - Resurser (dokumentation och forum t ex)

Var aktiv

Transparens (samtliga beslut bör exponeras i en öppen beslutslogg t ex) och kommunikation är nyckeln för att hålla ett samarbete aktivt – här är det viktigt att främja detta via effektiva verktyg (som t ex Slack), samt anordna regelbundna möten där status på projekt mm gås igenom. Uppmärksamma medlemmars leveranser, och träffas fysiskt en eller ett par gånger per år.

3. Resultat

3.1. Plattformen Open ePlatform

Open ePlatform etablerades baserat på OpenHierarchy i RIGES-projektet (ett gemensamt projekt med fem kommuner finansierat av Europeiska Unionen). OpenHierarchy är ett Java-ramverk publicerat som öppen källkod och som först etablerades som en del i ett examensarbete 2005. OpenHierarchy underhölls som ett hobbyprojekt till lansering av ramverket 2012. Målet med RIGES var att göra det enklare för företag och medborgare att få tillgång till information och kommunicera med kommunen i frågor som främst rörde stadsplanering och bygglovsfrågor, och Open ePlatform implementerade tolv e-tjänster för att stödja delar av detta.

> *”Då projektet inte kunde hitta någon befintlig leverantör som kunde uppfyllda den kravställning som togs fram, återstod endast att utveckla något nytt. Open ePlatform blev resultatet! Det fina med Open ePlatform är att den nu utgör en komplett och mycket flexibel e-tjänstplattform där projektkommunerna på ett enkelt sätt kan bygga nya e-tjänster inom kommunernas alla verksamhetsområden. På så sätt hoppas vi att RIGES effekter kommer att sträcka sig långt utöver projektets fokusområde. I och med att plattformen är skapad med öppen källkod kan även andra intressenter; i Sverige eller internationellt, också använda det vi skapat.”*[8]

Open ePlatform är publicerad under licensformen AGPLv3 vilket t. ex. innebär att leverantörer som använder kodbasen och erbjuder e-tjänsteplattformen till kunder som molntjänst måste bidra tillbaka till öppen källkodsprojektet med den vidareutveckling de gör i plattformen.

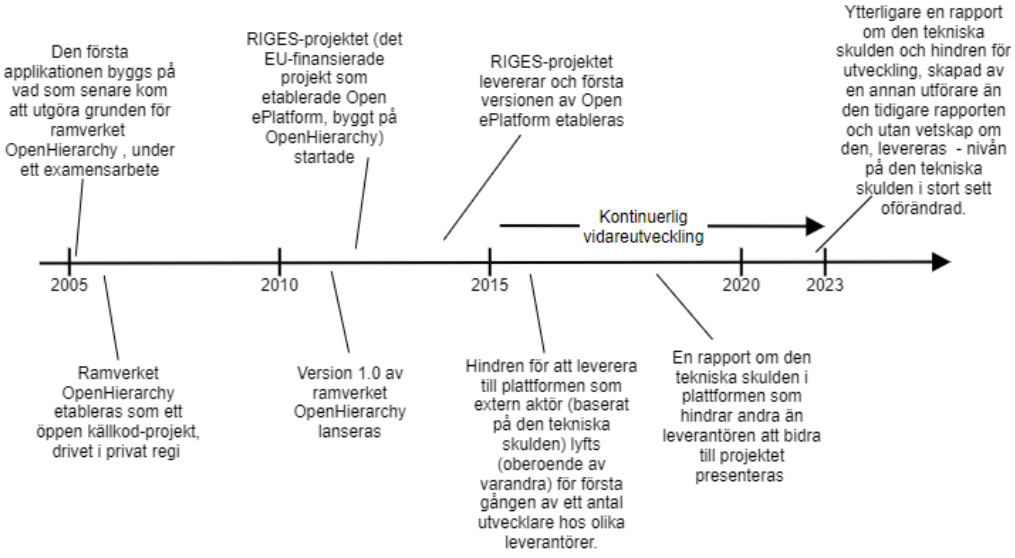
Efter projektet produktifierade leverantören Nordic Peak AB Open ePlatform och erbjuder i dagsläget drift och förvaltning till ett fast pris, samt vidareutveckling via traditionell upphandling. Nordic Peak AB är de enda som levererar Open ePlatform, av skäl som beskrivs under ”Hinder för leverans av kod till Open ePlatform” nedan. Nordic Peak AB omsätter ca 25 MSEK (2021) och har en relativt hög och stabil vinstmarginal på 60%. Nordic Peak AB ingår i Vismakoncernen.

Open ePlatform är i dag den populäraste e-tjänsteplattform i kommunsfären och används i dag av ca 180 kommuner som samverkar i en aktiv och välfungerande användarförening. Det finns ett antal proprietära e-tjänsteplattformar som konkurrerar på marknaden, men ingen som tillhandahåller samma flexibilitet eller möjligheter för samverkan och återanvändning.

3.2. Hinder för leverans av kod till Open ePlatform

Figur 1 ger en överblick av den historiska utvecklingen av Open ePlatform från första etableringen av det som kom att leda till OpenHierarchy till dagens Open ePlatform, samt när hindren för ytterligare aktörer att bidra till utvecklingen uppmärksammats av flera av varandra oberoende aktörer.

[8] Projektslutrapport Riges 1.0.pdf



Figur 1: Tidslinje över etableringen av Open ePlatform samt identifieringen hinder

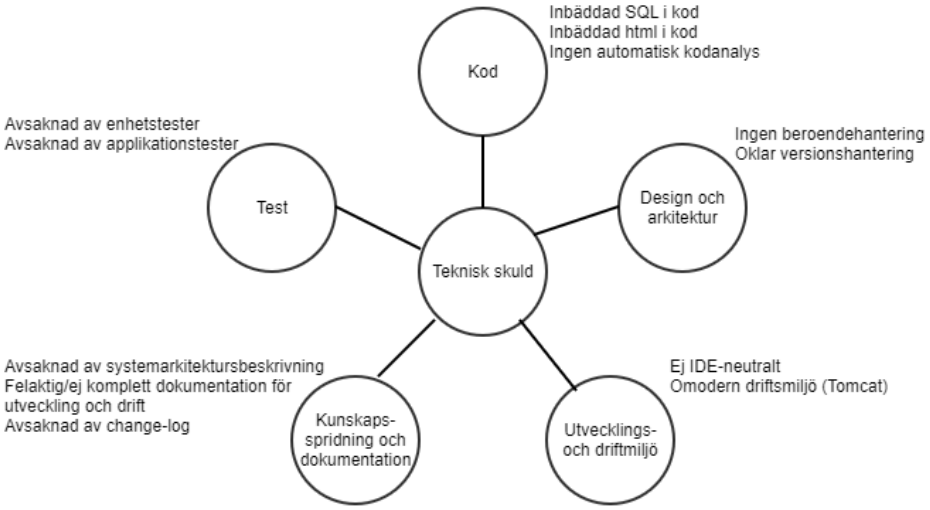
Beskrivningen av hindren utgår primärt från två oberoende tekniska rapporter, en från 2019 och en från 2022. Många av de hinder (sprungna ur den tekniska skuld som byggts upp i plattformen sedan 2005) som dokumenterats har dock uppmärksammats även tidigare i olika former sedan åtminstone 2016.

Med teknisk skuld menas problem under en mjukvara/systems livscykel som direkt härrör till mjukvarans/systemets utveckling[9]. Den tekniska skulden beskrivs i kategorierna kod, design och arkitektur, utvecklings- och driftmiljö, kunskapsspridning och dokumentation samt testning[10]. Teknisk skuld är ofta en konsekvens av snabb utveckling och medför betydande kostnader längre fram i livscykeln. För vidare detaljer, se Bilaga A.

De två rapporterna sammantaget påvisar betydande brister i form av teknisk skuld i samtliga kategorier – en teknisk skuld som i omgångar presenterats för leverantören, som dock valt att inte åtgärda det. Figur 2 presenterar en summering av dessa brister.

[9] https://doi.org/10.1016/j.giq.2022.101706

[10] https://doi.org/10.1016/j.jss.2012.12.052



Figur 2. Översikt av teknisk skuld i Open ePlattformen

Den tekniska skulden behövs liksom vilken monetär skuld som helst betalas av då den medför räntekostnader om man inte amorterar. Räntekostnaderna utgörs av den utökade tid och således pengar det tar att förädla applikationen som blir mer komplex för varje dag man avstår från att refaktorisera och modernisera lösningen. Utvecklingskostnaderna kommer alltså att stiga i takt med att man vidareutvecklar i plattformen i dess befintliga skick, och det blir i praktiken omöjligt att sprida utvecklingen på fler aktörer.

Kommuner är sålunda 100% beroende av leverantören vad avser förvaltning och vidareutveckling av plattformen, vilket skapar och vidmakthåller en imperfekt marknad. Med andra ord kullkastas själva logiken med öppen källkodslösningar. Då antal kunder dessutom ökat kraftigt de senaste åren, och leverantören har ett ändliga resurser, får de svårare och svårare att tillgodose kommunernas krav på vidareutveckling – något som inte vore ett problem om fler leverantörer kunde leverera till plattformen.

Den samlade bedömningen av en av Sundsvalls kommuns seniora Java-utvecklare lyder:

> *”Att för en extern aktör korrigera all skuld[11] är givetvis möjligt men skulle ta väldigt mycket tid i anspråk även för erfarna systemutvecklare.”*

Man kan alltså inte till en rimlig kostnad skala upp utvecklingstakten i Open ePlatform genom att lägga till externa resurser från andra leverantörer och är 100 % beroende av leverantören – som en utvecklare, som fristående från de större rapporterna undersökt Open ePlatform, uttryckt sig i en presentation om plattformen:

> *”Källkoden presenteras som öppen källkod och kan indirekt nås från websidan (...). Problemet är dock att denna källkod är allt annat än öppen.”*

[11] Som är nödvändigt för att nå en grundnivå i vad man kan definiera som modern systemutveckling – mycket lite, om ens något av det som Fowler och Foemmel presenterade i "Continous Integration" 2006 och som sedan dess gäller som standard vad avser modern systemutveckling, känns möjliga i plattformens nuvarande skick.

4. Analys och rekommendationer

För att säkerställa bärkraften i infrastrukturella komponenter byggda på öppen källkod krävs två saker. Först krävs en ständig översyn och återkommande analys av infrastrukturkomponenterna i sig. Att använda lösningar byggda på öppen källkod är i grunden bra och ett naturligt vägval för offentliga organisationer, men det kräver medvetenhet kring risk och sårbarhet relaterat till komponenterna i sig. Här behöver organisationer som nyttjar öppen källkodsbaserade infrastrukturkomponenter vara medveten om att resurser krävs för att säkerställa hållbarhet över tid. Man behöver avsätta resurser för att säkerställa att komponenten vidareutvecklas i rätt riktning.

Därefter krävs insatser kring deltagande i säkerställandet av komponentens fortsatta värde över tid. Man behöver avsätta resurser för utvecklare att kunna engagera sig i gemenskapen och bidra till komponentens vidareutveckling. Även detta kräver med andra ord resurser, varvid öppen källkod ej skall ses som ett låg-kostnadsalternativ till annan försörjning. Öppen källkod är en logik som kräver en annan typ av resurssättning än traditionell försörjning från leverantörer som tar (eller snarare borde ta) ansvar för sin produkt över tid och säkerställa kontinuerlig modernisering för fortsatt relevans och funktionalitet.

I en studie av affärsmodeller för öppen källkod analyserar Estelle Duparc med kollegor[12] just vilka konfigurationer av ansvar som fungerar för vilken typ av leverans. I relation till Open ePlatform har det centrala misstaget varit att förutsätta att leverantören tar sitt ansvar över tid. Frågan rörande om det som föreligger är en fråga om att vald licensform för Open ePlatform ej följts från leverantören eller helt enkelt en fråga om valhänthet i kravställan är ännu ej analyserad och avgjord.

4.1. Rekommendationer

På basis av den genomförda studien kan nedan rekommendationer ges till svenska kommuner:

- Då något bättre alternativ till Open ePlatform idag inte finns tillgängligt så rekommenderas Sveriges kommuner att **fortsätta använda** plattformen i sitt nuvarande utförande för att driva digitalisering.
- Dock bör **vidareutveckling** av plattformen **minimeras** för nu, för att ge leverantören en möjlighet att implementera stabiliserande förbättringar och förutsättningar för fler leverantörer att bidra till plattformen. För att uppnå en adekvat miniminivå på kvalité och säkra kontinuitet **behöver följande åtgärder vidtas**:
    - Inför kodberoendehantering (Maven)
    - Flytta koden (det som krävs för att bygga helheten, alltså även de bibliotek som ligger på unlogic) till GIThub (och anpassa utvecklingsprocess därefter och så att den främjar utveckling av fler leverantörer)
    - Avveckla det hårda IDE-beroendet (Eclipse) i utvecklingsprocessen
    - Producera en adekvat och korrekt dokumentation
        - Systemarkitekturbeskrivning
        - Utvecklingsguide
        - Driftguide
    - Inför automatiska test för allt nytt som byggs och påbörja implementation av tester för redan befintlig kod

[12] https://link.springer.com/article/10.1007/s12525-022-00557-9

På basis av den genomförda studien kan följande rekommendationer ges till offentlig sektor i stort:

- Undvik att ge en specifik leverantör ansvar att driva ett öppet källkodprojekt då detta inte fungerar (det finns fler exempel på det än detta specifika fall). Ett öppet källkodsprojekt inom offentlig sektor behöver hanteras av en fristående organisation som ansvarar för projektet och som säkerställer produktens kvalité och fortlevnad (som t.ex. Origo-samarbetet[13]).
- Utveckla en standard för hur öppen källkodsprojekt skall bedrivas inom offentlig sektor samt etablera OpenMunicipality som en ram för detta (se förslag nedan).

4.2. Förslag – etablera OpenMunicipality

Där ett Open ePlatform version 2 (arbetsnamn) är det första projektet som startas.

Det finns internationellt flera goda exempel på öppen källkodsprogram som drivs av offentlig sektor, som till exempel OS2[14] i Danmark som håller ihop utvecklingen av 23 lösningar i öppen källkod. Flera andra länder, som till exempel Italien[15] och Nederländerna[16], arbetar även de aktivt med öppen källkod i offentlig regi.

I Sverige har vi i dag ett flertal organisationer som på ett eller annat sätt verkar inom området öppen källkod eller på andra sätt delbara lösningar inom offentlig sektor, som till exempel Sambruk[17], DIGG[18] och Municipio[19]. Ett flertal kommuner delar också med sig av sin kod som öppen källkod, som till exempel Sundsvalls kommun[20] och Helsingborgs kommun[21]. Men ingen organisation har aktivt tagit ansvaret att etablera en organisation som på ett nationellt plan håller ihop och driver öppen källkodsutveckling likt till exempel OS2 i Danmark, vilket leder till att det blir väldigt svårt att skapa sig en samlad bild om vilka öppen källkodslösningar som finns tillgängliga.

Förslaget är att etablera en organisation som samlar ihop och driver öppen källkodsprojekt inom offentlig sektor i enlighet med de riktlinjer som beskrivs i kapitel 2 ovan. En mogen organisation kan se ut som i figur 5 nedan, och förslaget är att SKR/Inera eller något annat offentligt organ ansvarar för organisationen i stort.

[13] https://github.com/origo-map/origo

[14] https://os2.eu/

[15] https://developers.italia.it/en/

[16] https://github.com/J535D165/PublicSectorNL

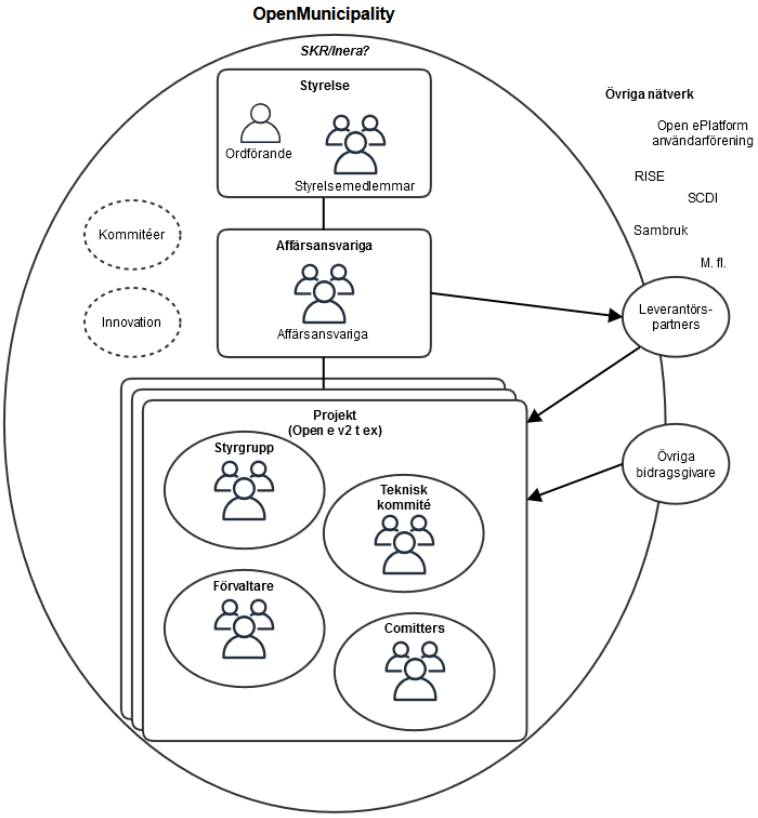
[17] https://sambruk.se/

[18] https://digg.se/

[19] https://www.municipio.se/

[20] https://github.com/Sundsvallskommun

[21] https://github.com/helsingborg-stad



Figur 5: OpenMunicipality

5. Avslutande kommentarer

Den tekniska skulden denna rapport identifierar i relation till Open ePlatform är symptomatisk för offentlig sektors hantering av digital infrastruktur. Offentliga aktörer har begått ett tankefel när de trott att digital infrastruktur är något som etableras och därefter fungerar utan ytterligare tillförda resurser. Att man ser på system och systemkomponenter på samma sätt som man ser på avlopp, bredband eller andra infrastrukturella resurser. Att man tror att digitalisering kan ske utan tilltagande kostnader i IT.

Delar av den vurm som förekommit kring öppen och fri källkod i offentlig sektor kan förklaras genom just denna syn att det är ”gratis”. Men som vi ser i relation till Open ePlatform så finns det ingenting som är gratis. Alla utvecklingsresurser som lagts in för att skapa e-tjänster på plattformen riskerar att bli förlupna kostnader. I takt med att plattformen kontinuerligt bygger ut sin tekniska skuld blir även utveckling och förvaltning dyrare och dyrare, vilket resulterar i avtagande effekt av utveckling över tid.

Open ePlatform utgör här ett varnande exempel på hur fel bristfällig styrning och resurssättning kan bli. Det bolag som drivit och förvaltat Open ePlatform har varit mycket lönsamma under åren, men givet en otillräcklig vidare- och re-investering i plattformen leder detta till inget mindre än ett fundamentalt strategiskt misstag givet att värdet av plattformen urholkats över tid. Konsekvensen är besvärlig för såväl leverantören själva som för kommun-Sverige som helhet.

I grund och botten är Open ePlatform ett sedelärande exempel kring vad som händer om man väljer att bygga en lösning på öppen källkod utan att avsätta rätt resurser för att säkerställa styrning av kontinuerlig vidareutveckling av plattformen. Konsekvenserna av detta är betydande och dyra, men samtidigt en god lärdom för fortsatt digitalisering. Öppen källkod utgör en naturlig komponent för fortsatt digitalisering av offentlig sektor, och rätt hanterat är det en överlägsen modell jämfört med många andra. Men detta ställer krav på livskraften i ekosystemet kring lösningen i sig. Det krävs en tydlighet i ansvar och mandat, och detta har än så länge inte tillräckligt belysts.

Göteborg 2023-02-23

Per Persson, doktorand och chefsarkitekt Sundsvalls kommun

Johan Magnusson, professor

Bilaga A: Inventering av teknisk skuld i Open ePlatform

Kod

> *”Överlag känns struktur, kod, ramverk, verktyg, osv. föråldrad och otidsenlig. Enligt utvärderarens personliga åsikt är bäst-före-datum passerat för länge sedan. Man får nästan intrycket av att mjukvaran precis plockats fram ur en tidskapsel från 2005. Det kan låta raljant, men uttrycker väldigt väl den spontana känsla som infinner sig.” Rapport 2022.*

Studien finner betydande brister i Open ePlatforms kod. Två exempel på dessa tveksamheter som betraktas som avsteg från god kodstandard är:

- Blandning av presentationslogik och affärslogik. Bilder, HTML-sidor, css-filer, etc. ligger blandat med Java-kod.
- SQL (för databas-operationer) ligger inbäddat i javakoden. Det har länge varit praxis att i stället använda ”persistence-API:er”, t.ex. JPA.

För att öka objektiviteten i analysen genomfördes en kodanalys med verktyget SonarLint[22] på en delmängd av plattformen. Detta verktyg analyserar koden och rapporterar sårbarheter, buggar och övriga avsteg från god programmeringssed. Analysen på denna delmängd[23] gav upphov till fler än 3 000 kommentarer vilket anses indikativt för bristande kodkvalitet.

Enligt rapporten från 2019 saknar Open ePlatform vidare:

> *”någon form av automatisk analys av källkodens kvalité eller vanliga fel. Det finns inte heller någon konvention som beskriver hur källkoden ska konstrueras. Genom att inte kontinuerligt mäta och förbättra källkodens kvalité ökar underhållskostnaden i mjukvaruprojekt, en kostnad som generellt redan är 40-80% av den totala kostnaden under projektets livstid”.*

Sammantaget är bedömningen att betydande kod-relaterad teknisk skuld föreligger.

Design och arkitektur

Open ePlatform använder sig av en uppsjö tredjepartsbibliotek. Detta är inget konstigt i sig, utan något man ofta brukar kunna förvänta sig av system byggda i t.ex. Java. Det som däremot är anmärkningsvärt är att man inte använder sig av något verktyg för beroendehantering. Idag är Maven, Gradle och liknande verktyg att betrakta som industristandard. Verktygen hjälper utvecklare att hämta och hålla reda på såväl direkta som transitiva beroenden, helt automatiskt.

Kombinerar man Maven/Gradle med verktyg som t.ex. Dependabot får man en automatiserad och regelbunden kontroll av sårbarheter i tredjepartsberoenden. Man får även hjälp att hålla dem uppdaterade. Det kunde inte hittas några sådana inbyggda mekanismer under någon av utvärderingarna av Open ePlatform. I stället förefaller samtliga beroenden ligga utspridda i subversion-repositories som

[22] https://www.sonarsource.com/products/sonarlint/

[23] endast ramverket OpenHierarchy som Open ePlatform bygger på analyserades

egna projekt (ett projekt per beroende – projekt som i många fall också måste kompileras innan de kan användas, i stället för att levereras som körbara binärer). Många beroenden tillhandahålls dessutom via en subversion-server på en domän av hobbykaraktär.

Då det inte finns någon beroendehantering kan man inte veta vilka versioner av respektive tredjepartsbibliotek (drygt 40 stycken) som är aktuella. Detta innebär en väldigt hög associerad kostnad i form av tid som krävs för att lista ut vilka versioner som är kompatibla med varandra och projektet. Detta innebär en förhöjd risk att versioner väljs som introducerar fel och/eller säkerhetsbrister i projektet. Även om applikationsramverket började byggas strax innan dessa verktyg fanns (Maven v2 släpptes 2005), så kan man fråga sig varför man har valt att inte modernisera denna hantering?

Sammantaget är bedömningen att betydande design och arkitektur-relaterad teknisk skuld föreligger.

Utvecklings- och driftmiljö

I dagsläget saknas korrekt dokumentation eller automatisering för uppsättning av utvecklingsmiljön. Detta innebär en förhöjd associerad kostnad i form av tid som krävs av nya utvecklare för att sätta upp deras lokala utvecklingsmiljö. Givet att detta måste göras manuellt är risken dessutom stor att miljön inte är enhetlig mellan utvecklare och produktion. Genom att automatisera detta kan hela processen bli reproducerbar och helt eliminera kostnaden och risken associerad med manuell hantering. De vanligaste verktygen för detta är Docker och Vagrant, som används för att skapa en virtuell utvecklingsmiljö. I princip alla större aktörer (Google, Amazon, Microsoft, Oracle) och större öppen källkods-projekt använder ett eller båda av dessa verktyg för att automatisera uppsättningen av utvecklingsmiljöer.

Det är även tydligt att det är IDE:n Eclipse som måste användas om man vill utveckla i plattformen då alla publicerade projekt är skapade som Eclipse-projekt (dokumentationen förutsätter även att denna IDE används). Att konstruera sina projekt och utvecklingsprocess mot en specifik IDE bör ses som ett tankefel. Det skapar onödiga tredjepartsberoenden och en försvårad utvecklingsprocess. i synnerhet om man (som många gör) använder andra IDE:er.

Installation av Open ePlatform görs i en separat servletcontainer (t.ex. Tomcat). Detta är idag förlegat och numera brukar man vanligtvis använda Tomcat och liknande servrar i form av en “embedded servletcontainer” vid användning av Java-applikations-ramverk, (som Open ePlatform utger sig för att vara). Ett exempel på en sådan tillämpning av Tomcat finns i t.ex. SpringBoot.

> *”Open ePlatform är i grunden en Java webbapplikation baserad på OpenHierarchy plattformen (http://openhierarchy.org) som i sin tur baseras på Servlet 2.4 specifikationen”. Open ePlatform – Dokument för utvecklare.*

Den nuvarande version av Servlet-specifikationen är 6.0, varvid man ej följt den tekniska utvecklingen.

Sammantaget är bedömningen att betydande teknisk skuld relaterad till utvecklings- och driftmiljö föreligger.

Kunskapsspridning och dokumentation

Vad gäller dokumentation slår rapporten från 2019 fast att:

> *”i dagsläget finns det kortare dokumentation på Open ePlatform webbsidan som förklarar vilka API-ändpunkter som finns, vart källkoden finns tillgänglig och övergripande vilka beroenden projektet har. Dokumentationen förklarar dock inte projektets övergripande arkitektur (SAD), hur uppsättning av utvecklingsmiljö sker, vad som har ändrats i varje version (changelog) eller hur projektet installeras och konfigureras för drift.”*

Sedan den rapporten skrevs har en guide för att sätta upp utvecklingsmiljön tagits fram. Detta dokument är dock fullt av fel och kan inte användas som en guide för att sätta upp utvecklingsmiljön.

Sammantaget är bedömningen att betydande kunskapsspridnings och dokuments-relaterad teknisk skuld föreligger.

Test

Även om det naturligtvis är möjligt att arbeta med kodstandard av lägre kvalitét upptäcktes saker som är mer besvärande i den senaste analysen av plattformen. Den viktigaste av dessa upptäckter är att inbyggda enhets- och applikationstester helt saknas.

Detta väcker en serie frågor:

- Hur säkerställer man bibehållen funktionalitet vid refaktorering?
- Hur förhindrar man regression, det vill säga hur undviker man att introducera buggar vid ny- och vidareutveckling?

Att skriva tester i samband med ny- och vidareutveckling av funktionalitet hör till en av de saker vi tar för givet idag. Det är någonting man bara gör. Att försumma detta är en allvarlig avvikelse vilket naturligtvis minskar förtroendet för mjukvaran.

Sammantaget är bedömningen att betydande test-relaterad teknisk skuld föreligger.