ChatGPT

Arhitektuuri kihid ja detsentraliseeritus: Süsteem tuleks kavandada kihilise arhitektuurina, eristades servatasandi (kohapealsed sensorid ja droonisüsteemid), piirkondliku tasandi (kohalikud juhtimiskeskused) ning üleriigilise tasandi (keskne situatsioonipilt ja koordineerimine). Niisugune arhitektuurijaotus vastab ka NATO standardites (nt STANAG 4586) kirjeldatud põhimõtetele, mis rõhutavad ühiste liideste ja definitsioonidega modulaarset ülesehitust [1]. Iga tasand saab tegutseda optimaalse latentsuse ja koormusega: servasõlmed töötlevad andmeid reaalajas sündmuste toimumiskohal, piirkondlikud sõlmed integreerivad mitme sensori info piirkondlikuks operatiivpildiks ning üleriigiline kiht koondab ülevaate ja suunab strateegilist otsustamist. Selline kihtideks jaotamine parandab skaleeritavust ja töökindlust, kuna süsteem ei sõltu ühestainsast kesksõlmest ning võib vajadusel hõlpsasti laieneda uute serva- või piirkonnasõlmede lisamisega. (Tugev argument, kriitiline ühilduvus Anduriliga)

Võrgutaluvus ja autonoomsus: Detsentraliseeritud kihilise arhitektuuri oluliseks eeliseks on vastupidavus sidekatkestustele ning autonoomne toimetulek olukorras, kus keskne sõlm (nt riigi tasandi juhtimiskeskus) ajutiselt puudub. Serva- ja piirkonnasõlmed suudavad jätkata kriitiliste funktsioonide täitmist ka ilma pideva ühenduseta kõrgemate tasanditega – näiteks jätkavad andurid ja droonisüsteemid kohaliku avastuse, klassifitseerimise ja jälgimisega ning jagavad olulisi hoiatusi lokaalsete kanalite kaudu [2]. Selline lähenemine tagab, et üksikute sõlmede või sidekanalite rikke korral ei muutu kogu süsteem pimedaks; operatsioonid degradeeruvad graatsiliselt, säilitades põhivõimekused kohapeal kuni ühenduse taastumiseni. Tulemusena suureneb süsteemi usaldusväärsus ka vaenulikes tingimustes, kus vastane võib sihtida keskseid juhtimispunkte või kommunikatsioone. (Tugev argument, kriitiline ühilduvus Anduriliga)

Sündmustel põhinev pub/sub infovahetus: Andmevahetus arhitektuuris peaks toimuma avaldaja-tellijana (publish-subscribe, pub/sub) mudeli alusel, mitte dokumentide saatmisega läbi ühe keskse sõlme. Pub/sub lähenemine tähendab, et infoallikad (nt sensorid) avaldavad sündmuseid või teateid teemadesse ning huvipooled (teised süsteemi komponendid) tellivad neid teemasid, saades teavituse kohe, kui uus sündmus on saadaval [3]. Selline asünkroonne, lõdvalt seotud kommunikatsioon elimineerib vajaduse tsentraalseks andmekogumiseks ja -jagamiseks igas etapis – süsteemi osad ei pea teadma üksikasjalikult üksteisest, mis lihtsustab integreerimist ja muudab lahenduse paindlikumaks. Lisaks võimaldab pub/sub mudel paralleelset infotöötlust ja filtrite rakendamist: iga sõlm saab valida, millist infot ta vajab, selle asemel et läbi närida suurt hulka ebaolulisi andmeid. Tulemuseks on väiksem võrgu- ja arvutuskoormus ning parem skaleeritavus, kuna uusi tarbijaid saab lisada ilma, et peaks muutma andmeallikate loogikat. (Tugev argument, kriitiline ühilduvus Anduriliga)

Standardiseeritud liidesed ja adapterid: Süsteemi laiendatavuse ja partnerite kaasamise huvides on kriitiline rakendada standardiseeritud liideseid ning adapterite põhimõtet koos avalikult kättesaadava SDK-ga (tarkvaraarenduse komplektiga). Ühtne ja hästi dokumenteeritud liides võimaldab erinevatel sensoritel, droonidel ja alamsüsteemidel süsteemiga liituda ilma ad-hoc arenduseta iga uue integratsiooni jaoks. Näiteks on Suurbritannia SAPIENT arhitektuuris rõhutatud moodulite plug-and-play ühilduvust ja avatud standardeid, mis loovad konkurentsi soodustava ökosüsteemi erinevatele tarnijatele [4]. Kui meie süsteemi tuum defineerib selged API-d ja pakub ametlikku SDK-d, saavad nii sise- kui välispartnerid luua uusi võimekusi (adaptereid) kiiremini ja väiksema riskiga, vähendades tulevikus sõltuvust ühe tarnija lahendustest. (Tugev argument, kriitiline ühilduvus Anduriliga)

Konformsustestid ja sertifitseerimine: Standardiseeritud liideste tõhusaks toimimiseks on vajalik ka range konformsuse testimine – igale uuele adapterile või moodulile tuleb kehtestada nõue läbida vastavustestid, mis kinnitavad protokollide ja andmeformaatide järgimist. Selline testraamistik (nagu

SAPIENT initsiatiivis väljatöötatud automaatne Test Harness) annab kindluse, et uued komponendid suudavad suhelda süsteemis vigadeta [5]. Konformsustestide olemasolu lihtsustab integratsiooni: arendajad saavad juba arendusfaasis avastada kõrvalekaldeid standardist ning tellija saab enne vastuvõtmist veenduda, et kolmandate osapoolte lahendused ei kahjusta süsteemi stabiilsust. Lõpptulemusena tõuseb süsteemi usaldusväärsus ja väheneb integreerimisprojektide ajakulu. (Tugev argument, toetab arhitektuurilist lähenemist)

Pilvevalmidus ja topoloogiapariteet: Kuigi süsteemi arhitektuur peaks olema pilvevalmis – kasutades kaasaegseid pilvetehnoloogiate põhimõtteid (nt konteineriseeritus, mikroteenused, automaatne skaleerimine) –, ei tohi pilv olla ainsaks toimimiskeskkonnaks. Oluline on saavutada topoloogiapariteet: sama lahendus arhitektuuriliselt töötab ühtviisi nii avalikus/erapilves kui ka lokaalses andmekeskuses või eesliini (edge) sõlmes. See tähendab, et pilves kasutatavad tehnoloogiad (näiteks sõnumijärjekorrad, andmevood, konteinerorkestratsioon) peavad olema saadaval ka lahenduse iseseisval paigaldusel kliendi taristus. Niisugune paindlikkus tagab, et süsteem on kasutatav ka olukordades, kus pilveteenuste kasutamine pole võimalik või soovitav – ilma funktsionaalsuse vähenemiseta. Lähenemine vähendab sõltuvust konkreetsest taristust ning lihtsustab testkeskkondade ülesseadmist, sest arendus- ja tootmiskeskkond võivad olla identsed sõltumata asukohast. (Keskmise tugevusega, toetab arhitektuurilist lähenemist)

NATO standarditega ühilduvus: Rahvusvahelise koostöö ja tulevikukindluse huvides peab süsteemi arhitektuur tuginema NATO standarditele (STANAGidele) ning laialt levinud andmeformaatidele. Näiteks STANAG 4586 defineerib droonide juhtimissüsteemide ühtse arhitektuuri ja liideste komplekti, mis võimaldab erinevate tootjate UAV-d ja maajaamad omavahel tööle saada [1]. Samuti on oluline toetada NATO luure- ja seiresüsteemides kasutatavaid standardeid nagu STANAG 4609 (täisliikumisvideo metaandmete formaat) jmt, et integreeruda sujuvalt liitlaste infosüsteemidega. Eraldi tähelepanu väärib Suurbritannia poolt arendatud SAPIENT protokoll, mis on kujunenud avatud standardiks autonoomsete sensorite ja efektorite integreerimiseks ning on juba Briti kaitseministeeriumi poolt võetud kasutusele droonitõrjes [6]. NATO plaanib SAPIENT protokolli adopteerida NATO standardina C-UAS valdkonnas [7], seega SAPIENTiga ühilduva arhitektuuri poole liikumine annaks meie süsteemile olulise eelise liitlasraamistikes kasutuselevõtul. (Tugev argument, kriitiline ühilduvus Anduriliga)

ASTERIX ja andmeformaatide standardid: UAV seiresüsteem peab suutma vastu võtta ja edastada andmeid standardiseeritud formaatides, et hõlbustada infovahetust teiste süsteemidega. Eriti oluline on ASTERIX (All-Purpose Structured Eurocontrol Surveillance Information Exchange) – üldine lennunduse järelevalve andmete vahetamise standard, mida kasutatakse laialdaselt nii radari- kui ka lennuliiklusandmete edastamisel üle maailma [8]. ASTERIX hõlmab erinevaid kategooriaid alates primaar- ja sekundaarradari sihtmärgiteadetest kuni töödeldud koondsüsteemi rajajälgedeni, võimaldades meie süsteemil esitada tuvastatud droonid ja muud objektid formaadis, millest teised õhuvalve või lennujuhtimise süsteemid aru saavad [9]. Toetades ASTERIX-i (ning vajadusel teisi NATO kokkuleppelisi vorminguid), tagame, et süsteemi poolt loodud ja tarbitav info on universaalses keeles, vähendades integratsioonibarjääre tsiviil- ja sõjalise taristuga. (Keskmise tugevusega, toetab arhitektuurilist lähenemist)

Õppetunnid Ukraina konfliktist: Hiljutine sõjakogemus Ukrainas on rõhutanud detsentraliseeritud, võrgukindla arhitektuuri tähtsust lahinguväljal. Süsteemid peavad suutma toimida ka siis, kui side on häiritud või keskne juhtimissõlm (C2) on rivist väljas – üksused eesliinil peavad säilitama situatsiooniteadlikkuse ja jätkama tegutsemist olemasoleva info põhjal. Ukrainas on täheldatud, et hajutatud ja alt-üles (kaasav) tehnoloogialahendustega saavutati kiire innovatsioon ning vastupidavus: näiteks on seal droonivõimekusi arendatud suurel määral rohujuuretasandil, kaasates arvukalt väikseid arendajaid ja vabatahtlikke, mis andis eelise paindlikkuses [10]. Sellest lähtudes peaks meie süsteemi analüüs arvestama lahendusi, kus iga sõlm või taktikaline üksus omab teatud autonoomiat otsuste

tegemisel ja info jagamisel lokaalselt, ilma et kogu operatsioon sõltuks haavatavast kesksest infrastruktuurist. Selline lähenemine suurendab oluliselt vastupanuvõimet reaalses konfliktis, kus vastane püüab esmalt just keskseid sõlmi häirida. (Tugev argument, toetab arhitektuurilist lähenemist)

Andme- ja juhtimistasandi eristamine: Süsteemi kavandamisel tuleb lahus hoida reaalaja andmevood (operatiivinfo liikumine) ning juhtimis- või haldustasandi toimingud (andmehaldus, konfiguratsioon, õiguste haldus). See tähendab, et situatsioonipilti ja sensoriteateid vahendav andmetasand toimib maksimaalse kiiruse ja minimaalse latentsusega, samal ajal kui governance-tasandil toimub näiteks kasutajaõiguste kontroll, poliitikate rakendamine ja süsteemi ümberseadistamine eraldi kanalites. Selline jaotus väldib seda, et bürokraatlikud protsessid või raskepärane andmehaldus aeglustaks kriitilisi reaalaja funktsioone. Näiteks võib droonide reaalajas jälgimise voog toimida pub/sub mudelis otse vajalikele tarbijatele, aga andmete ligipääsuõigusi või säilitamispoliitikat hallatakse paralleelselt teises kihis, mis sekkub ainult erandolukorras. Nii tagatakse, et operatiivotsused ja -tegevused ei jää kunagi ootele administratiivsete protseduuride tõttu, säilitades samas süsteemis kontrolli ja järelevalve võimaluse kõrgemal tasandil. (Keskmise tugevusega, toetab arhitektuurilist lähenemist)

Turvalisus ja identiteedihaldus servas: Detsentraliseeritud arhitektuuri puhul tuleb rakendada nullusaldusel põhinevat turvamudelit, kus igal seadmel ja sõlmel on tugev krüptograafiline identiteet ning õigused, mis kehtivad ka autonoomses töös. Soovitav on kasutada standardeid nagu IEEE 802.1AR Secure Device Identity (DevID), mis defineerib seadmele krüptograafiliselt seotud unikaalse tunnuse – tehasest kaasa antud esmane identiteet, mida saab täiendavalt kohaliku halduri poolt uute sertifikaatidega laiendada [11] . Selline lahendus võimaldab igal servasõlmel tõendada oma identiteeti ka siis, kui keskset autoriteeti parasjagu pole kättesaadav. Juurdepääsu kontrolliks peaks süsteem toetama attribuudipõhist ligipääsukontrolli (ABAC), kus otsuseid tehakse mitme teguri (kasutaja roll, seadme tüüp, operatsiooni kontekst jms) alusel; oluline on, et need reeglid ja poliitikad jooksutatakse lokaalselt serva- või piirkonnasõlmes, vältides vajadust küsida luba keskselt serverilt iga tegevuse puhul. Kohapealne võtmehaldus ning volituste kontroll (näiteks eel-laaditud või piirkondlikult väljastatud võtmed, ajutised sertifikaadid ja õiguste vahemälud) tagavad, et isegi sidekatkestuse korral saab süsteemi turvalisus jätkuvalt kehtida ning uued seadmed/automaatsõlmed saab vajadusel operatiivselt liita vastavalt eelmääratud poliitikatele. (Tugev argument, kriitiline ühilduvus Anduriliga)

Partitsioonitaluvuse prototüübikatsed: Soovitatav on juba eelanalüüsi faasis kavandada prototüübid ja katsestsenaariumid, mis demonstreerivad arhitektuuri toimivust võrgupartitsiooni (ühenduskatkestuste) tingimustes. Näiteks tuleks testida olukordi, kus piirkondlik kesksõlm läheb ajutiselt võrguühenduseta – kas servasõlmed jätkavad iseseisvalt sihtmärkide tuvastamist ja jälgimist ning kas kogutud info sünkroniseeritakse sujuvalt tagasi kõrgemale tasandile ühenduse taastudes. Sellised katsed annavad kinnitust, et süsteem vastab kriitilistele vastupidavusnõuetele ega kaota funktsionaalsust ka äärmuslikes tingimustes. Partitsioonitaluvuse tõestamine prototüübis suurendab sidusgruppide usaldust valitud arhitektuurilahenduse vastu ning aitab varakult avastada võimalikke puudujääke (nt andmete kooskõlastamise probleemid taasühinemisel), mille lahendamine on varases staadiumis oluliselt lihtsam. (Keskmise tugevusega, toetab arhitektuurilist lähenemist)

Isetervenemine ja veaautomaatika: Detsentraliseeritud süsteemis peaks olema sisseehitatud võimekus nn isetervendamiseks – kui mõni sõlm või teenus ebaõnnestub, tuvastavad ülejäänud komponendid selle ning võtavad automaatselt üle kriitilised funktsioonid või käivitavad tõrkunud komponendi uuesti uues asukohas. See nõuab arhitektuurilt staatust mittesäilitavat (stateless) teenuste disaini ja orkestreerimisvõimekust, mis suudab dünaamiliselt ressursse ümber jaotada. Näiteks võib piirkondliku sõlme rivist väljalangemisel mõni teine piirkond või pilvekomponent ajutiselt üle võtta selle rolli või servasõlmed jaotavad omavahel otse andmete jagamise, kuni tavapärane struktuur taastub. Isetervenev arhitektuur vähendab inimsekkumise vajadust intsidentide lahendamisel ja parandab süsteemi kasutatavust, kuna talitlushäired ei väljendu pikaajalise teenusekatkestusena. Sellise

dünaamilise vastupidavuse saavutamine on oluline osa modernsete kaitsesüsteemide nõuetest ning haakub hästi eelnimetatud partitsioonitaluvuse ja skaleeritavuse põhimõtetega. (Keskmise tugevusega, toetab arhitektuurilist lähenemist)

Skaleeritavus vs monoliit: Et süsteem suudaks ajas kasvada ilma jõudluse ja hooldatavuse probleeme tekitamata, tuleb vältida monoliitse arhitektuuri kujunemist. Monoliitne lahendus, kus kõik funktsioonid ja andmed koonduvad ühte suuremasse süsteemi, võib algselt olla lihtne teostada, kuid aja jooksul muutub see raskesti skaleeritavaks – igasugune uuendus või lisavõimekus eeldab terve süsteemi muutmist ning koormuse kasv ühes komponendis mõjutab kogu platvormi. Detsentraliseeritud, teenustepõhine arhitektuur lahendab seda: funktsionaalsused jagatakse iseseisvateks teenusteks või sõlmedeks, mis suhtlevad selgelt defineeritud liideste kaudu. Nii saab iga komponenti skaleerida horisontaalselt (lisades ressursse vaid sinna, kus koormus suureneb) ning uuendada või asendada seda sõltumatult teistest. Such approach prevents the uncontrolled expansion of a “one big system” and avoids a situation where the development of the entire solution slows down due to bottlenecks in a single core module. (Tugev argument, kriitiline ühilduvus Anduriliga)

---

1 STANAG 4586 - Wikipedia
https://en.wikipedia.org/wiki/STANAG_4586

2 4 5 6 SAPIENT autonomous sensor system - GOV.UK
https://www.gov.uk/guidance/sapient-autonomous-sensor-system

3 What is Pub/Sub Messaging? - Pub/Sub Messaging Explained - AWS
https://aws.amazon.com/what-is/pub-sub-messaging/

7 NATO “to adopt UK’s SAPIENT protocol as C-UAS standard” – Unmanned airspace
https://www.unmannedairspace.info/counter-uas-systems-and-policies/nato-to-adopt-uks-sapient-protocol-as-c-uas-standard/

8 9 ASTERIX - Wikipedia
https://en.wikipedia.org/wiki/ASTERIX

10 The Drone Revolution - Lessons from the Russia-Ukraine War and the Future of Warfare
https://www.galaxyuas.com/post/the-drone-revolution-lessons-from-the-russia-ukraine-war-and-the-future-of-warfare

11 IEEE SA - IEEE 802.1AR-2018
https://standards.ieee.org/ieee/802.1AR/6995/