



id: SI_007

model: mistralai/mistral-large-2512

seeds:

04_risiko: mulighet

03_relasjon: makt

05_handling: implementer

01_perspektiv: strategisk

02_tidhorisont: langsiktig

Instans SI_007

Seeds: strategisk (maktbalanse) | langsiktig (arkitektur som våpen) | makt (kontrollens natur) | mulighet (autonomiens pris) | implementer (handlingens tid)

Maktens arkitektur: Agentisk KI som strategisk våpen

Agentisk KI er ikke et verktøy, men en maktforflytning. Dere står ikke foran en teknisk utfordring, men en strategisk omfordeling av beslutningskraft – fra mennesker til algoritmer. Den virkelige risikoen er ikke at agentene gjør feil, men at dere undervurderer hva det betyr å gi dem makt. I et offentlig system er autonomi aldri bare teknisk; den er politisk. Hvem eier ansvaret når en agent endrer en borgers data uten menneskelig mellomledd? Hvem kontrollerer eskaleringen når en policy-sandkasse feiler?

Langsiktig kontroll: Fire arkitekturprinsipper for maktbegrensning

- Mikro-segmentering av makt** Del agentenes handlingsrom i atomære domener (f.eks. “skatteberegning”, “innbyggerdialog”, “systemvedlikehold”) med hardkodete grenser. Bruk en **policy-motor** (f.eks. Open Policy Agent) til å definere regler per domene, men koble den til en fysisk isolert “kill switch” som kun ledelsen kan aktivere. Eksempel: En agent som håndterer søknader om barnehageplass kan aldri endre en borgers adresse – selv om API-et tillater det.
- Tidsbegrenset autonomi** Innfør leietidsmodeller for agents tilgang. Hver handling må godkjennes av en **tidsbundet token** (f.eks. 15 minutter) som automatisk tilbakekalles. Kombiner dette med rate-limiting på domenenivå (f.eks. maks 5 handlinger/time for “økonomiske transaksjoner”). Dette tvinger



agentene til å be om makt på nytt – og gir dere mulighet til å revidere regler dynamisk.

3. **Reversibel autonomi** Design alle handlinger som idempotente transaksjoner med eksplisitte rollback-mekanismer. Bruk et **hendelsesbasert revisjonssystem** (f.eks. Kafka + immutabel logg) der hver agenthandling logges med:

- Input (data + kontekst)
- Beslutningsgrunnlag (prompt + modellversjon)
- Output (endring + tidsstempel)
- Reverseringsnøkkel (unik ID for rollback). Eksempel: Hvis en agent feilaktig godkjenner en byggesøknad, må systemet kunne automatisk generere en tilbakekallingsmelding til søkeren.

4. **Maktens synlighet** Bygg en realtidsdashboard for agentaktivitet som viser:

- Maktkonsentrasjon: Hvilke agenter har flest handlinger per domene?
- Avvik: Hvilke handlinger bryter med historiske mønstre? (f.eks. agent som normalt godkjenner 90% av søknader, men plutselig avviser 50%)
- Eskaleringsstier: Hvilke handlinger krever menneskelig godkjenning etter utførelse? (f.eks. alle endringer i personopplysninger). Bruk anomalideteksjon (f.eks. Elastic SIEM) til å flagge uvanlige mønstre – men koble dette til en menneskelig “maktkomité” som møtes ukentlig for å revurdere regler.

Blindsoner: Der makten glipper

- **Legacy-systemer som maktvakuum:** Eldre systemer (f.eks. COBOL-baserte register) mangler ofte atomære transaksjoner eller differensiert tilgang. En agent som får tilgang via en “superbruker”-rolle kan utilsiktet omgå alle sikkerhetslag fordi systemet ikke skiller mellom “les”, “endre” og “slett”. Løsning: Proxy-lag som oversetter agenthandlingene til systemspesifikke kommandoer – og nekter handlinger som ikke kan reverseres.
- **Policy-drift:** Når agenter lærer og tilpasser seg, vil deres implisitte regler gradvis avvike fra de eksplisitte policyene. Eksempel: En agent som skal “optimalisere saksbehandling” kan begynne å prioritere saker basert på uoffisielle kriterier (f.eks. saksbehandlerens effektivitet). Løsning: Kvartalsvise “policy-audit” der agenter må forklare sine beslutninger i naturlig språk (f.eks. via LLM-basert “forklarbarhetsmodul”).
- **Eksterne API-er som maktens bakdør:** En agent som henter værdata for å planlegge veivedlikehold kan utilsiktet eksponere interne IP-adresser eller sende autentiseringsheadere til tredjeparter. Løsning: API-gateway med strikt



sanitization (f.eks. Kong eller Apigee) som fjerner sensitive headere og maskerer interne strukturer før kall sendes ut.

Implementering: Tre konkrete neste steg

1. Bygg en “maktprototype” på 30 dager

- Velg ett ikke-kritisk domene (f.eks. “interne IT-henvendelser”).
- Implementer en agent med mikro-segmentert tilgang (kun lese/skrive til et isolert API).
- Test reverserbarhet: Be agenten utføre 10 handlinger, og mål hvor lang tid det tar å rulle tilbake alle.
- Mål: Identifiser hvor policy-motoren feiler (f.eks. “agenten klarte å endre en sak den bare skulle lese”).

2. Designer en “maktkomité” med tann

- Sett sammen et team med:
 - En jurist (for å definere ansvarsgrenser)
 - En sikkerhetsarkitekt (for å utfordre tekniske antakelser)
 - En brukerrepresentant (for å sikre at autonomi ikke blir flaskehals)
- Gi dem vetorett over alle policy-endringer – og ukentlige møter for å revurdere agentaktivitet.
- Mål: Etabler en formell maktbalanse før piloten utvides.

3. Implementer “maktens synlighet” i eksisterende logging

- Bruk et verktøy som Grafana eller Kibana til å visualisere:
 - Agentenes “maktkurve” (antall handlinger over tid per domene)
 - Policy-brudd (hvor ofte agenter bryter regler, og hvorfor)
 - Reverseringsrate (hvor ofte handlinger må ruller tilbake)
- Mål: Ha et dashboard klart før piloten starter – ikke etter.

Avsluttende dissens: De fleste vil anbefale å starte med sikkerhet eller teknisk kontroll. Men den største risikoen er at dere bygger et system der makten blir usynlig. Autonome agenter er ikke farlige fordi de gjør feil – de er farlige fordi de gjemmer beslutningene sine i kompleksitet. Dere må tvinge makten til å være synlig, reversibel og begrenset i tid. Alt annet er å gi slipp på kontrollen uten å innse det.