



# Agents for DDD Back and Forth

Andrea Bedei

Fabio Notaro

Giacomo Leo Bertuccioli



# Introduzione

- ❖ Sia DDD che AOSE riguardano il ridurre la complessità, ma con sottili differenze:
  - ❖ DDD mira alla complessità strutturale, statica
  - ❖ AOSE mira alla complessità dinamica, di interazione

# PANORAMICA SUL DOMAIN-DRIVEN DESIGN

- ❖ Sono riportati i concetti del DDD come ci sono stati presentati a lezione
- ❖ Come riassunto da Evans il DDD è un approccio allo sviluppo di software complessi in cui i progettisti e gli sviluppatori:
  - ❖ si concentrano sul dominio principale
  - ❖ esplorano modelli in una collaborazione creativa di professionisti del dominio
  - ❖ parlano un linguaggio obliquitos all'interno di bounded context
- ❖ Spiegazione dei concetti di: dominio e modello, dominio suddiviso in bounded context e loro integrazione

# RILEVANZA DEL DDD PER AGENTI

- ❖ L'autore suggerisce di usare il DDD per strutturare il domain model dentro un MAS.
- ❖ Nel Multi Agent Programming tradizionale gli elementi del dominio appartengono o all'agente, o all'ambiente o all'organizzazione e possono essere visti come layer orizzontali → domini complessi difficili da rappresentare → DDD aiuta a identificare bounded context per rompere complessità e scorporare in porzioni isolate

# LIMITI DEL DDD CHE RICHIAMANO GLI AGENTI

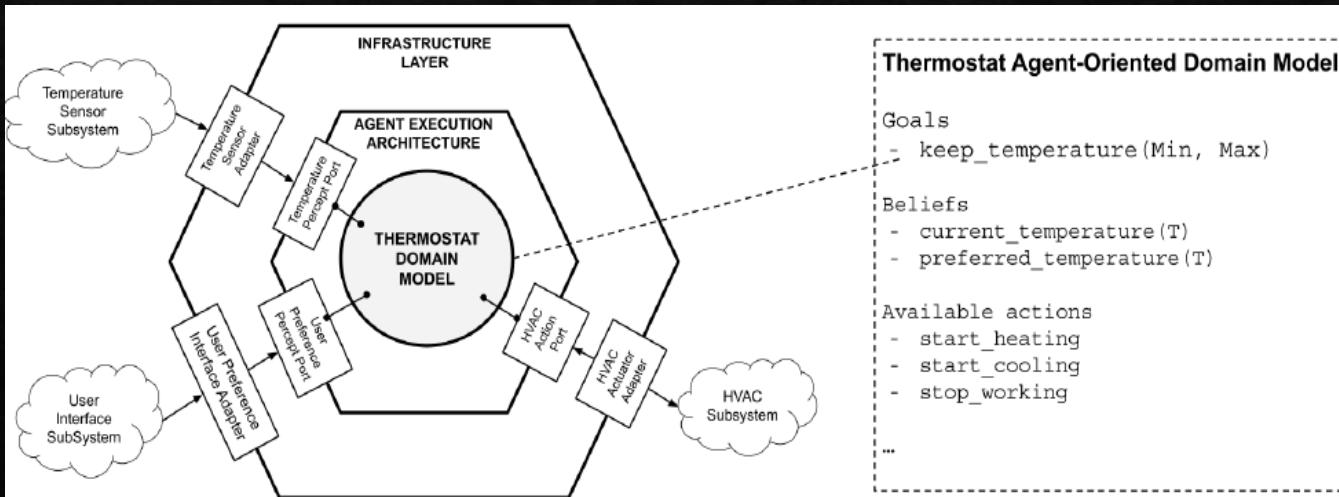
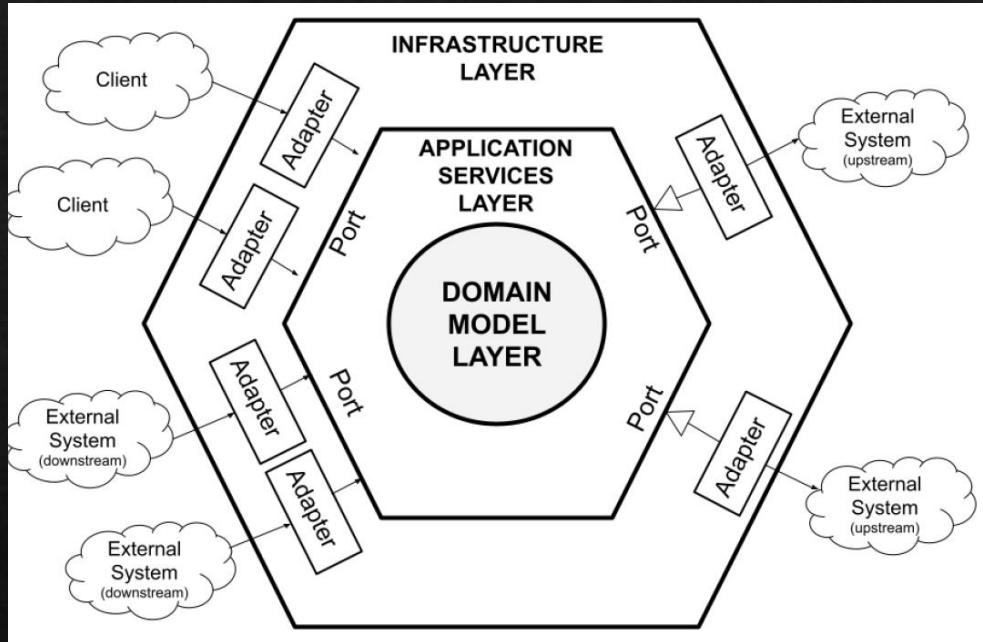
- ❖ Landre in un suo paper rimarca che DDD è grandioso per affrontare complessità strutturale ma non complessità dinamica, tipica dei sistemi autonomi, che si manifesta in interazioni mutevoli e comunicazioni arbitrarie tra agenti.

# COME MIGLIORARE DDD CON LA MODELLAZIONE AGENT-ORIENTED

- ❖ Dentro un singolo bounded context i concetti di AOSE possono essere applicati per arricchire il modello di dominio e migliorare la capacità del sistema a reagire e adattarsi ai cambiamenti dinamici
- ❖ A livello di integrazione di bounded context, l'AOSE può migliorare la gestione delle interazioni delle dipendenze tra di essi

# SINGOLO BOUNDED CONTEXT

- ❖ O mappare un bounded context come un **singolo agente**: in una smart room un termostato può essere modellato come un agente, con credenze di temperatura attuale e preferita e piano di accendere o spegnere riscaldamento
- ❖ O mappare un bounded context con un **singolo agente + ambiente separato**: in un bounded context, l'ambiente può essere rappresentato da artefatti (essitono dunque artefatti individuali, come risorse private o database locali) o artefatti condivisi (come registro degli eventi o API gateway). Es: l'agente termostato può interagire con sensori di temperatura modellati come artefatti
- ❖ O mappare un bounded context come MAS (se un singolo bounded context è troppo complicato per un singolo agente)



# INTEGRAZIONE TRA BOUNDED CONTEXT

- ❖ L'AOSE può migliorare interazioni e dipendenze tra bounded context in diversi modi:
  - ❖ **Bounded context come agenti interagenti:** ogni bounded context = un agente autonomo che interagisce con gli altri attraverso protocolli d'interazione standard che prediligano asincronicità (MQTT, Kafka...), in modo da garantire disaccoppiamento e scalabilità
  - ❖ **Bounded context come organizzazioni di agenti:** un bounded context = più agenti, in cui ci sono ruoli che definiscono le responsabilità di ciascun agente ed esistono protocolli di comunicazione (tipo Moise) basati su queste gerarchie e ruoli

# CONCLUSIONI

- ❖ Pro dell'integrazione tra DDD e AOSE comporta:
  - ❖ gestione facilitata della complessità strutturale e dinamica
  - ❖ adattabilità e resilienza: AOSE introduce i concetti di autonomia, reattività e supervisione, che migliorano la capacità dei bounded context di adattarsi ai cambiamenti
  - ❖ disaccoppiamento e scalabilità: grazie a comunicazioni asincrone ed eventi, che riducono le dipendenze tra bounded context
- ❖ Svantaggi e sfide:
  - ❖ complessità progettuale
  - ❖ problemi di performance a causa delle comunicazioni asincrone
  - ❖ problemi di complessità a mantenere bilanciati nel tempo sia DDD che AOSE