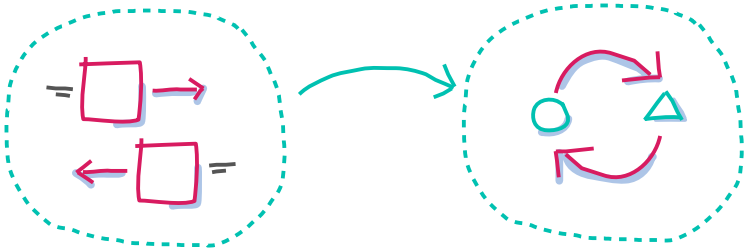
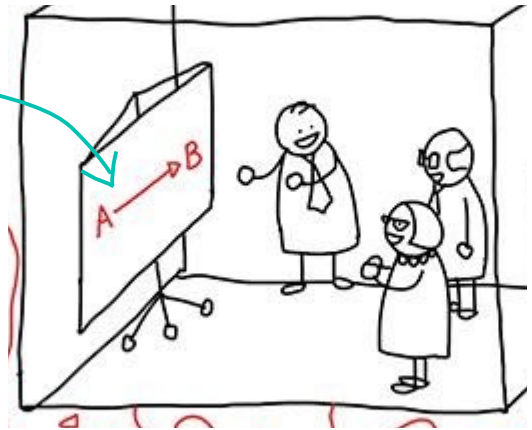


Architecture for Flow w/ Wardley Mapping, DDD, Team Topologies

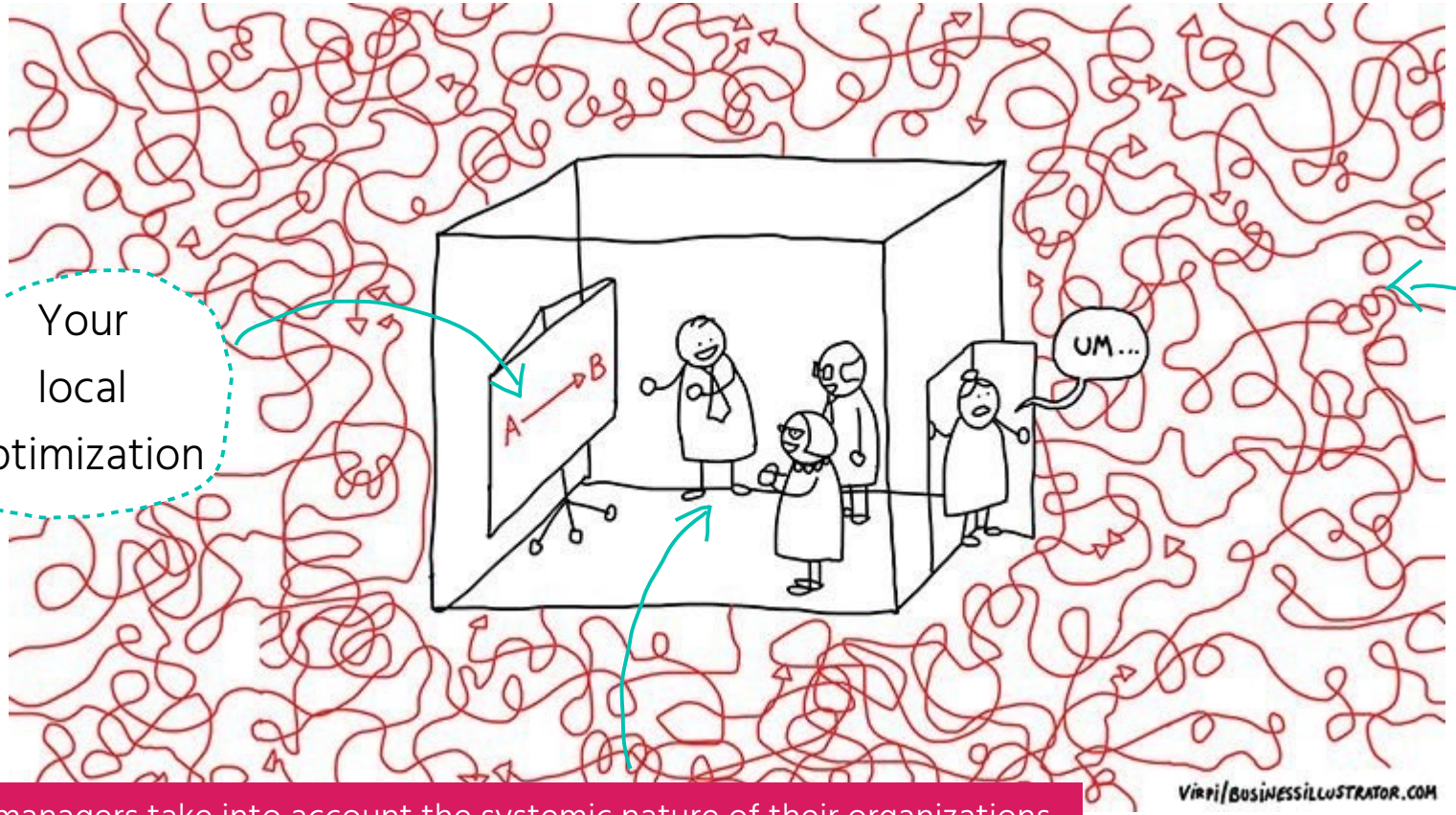
Susanne Kaiser
Independent Tech Consultant
@suksr

Problem with Local Optimization

Your local optimization



“A system is more than the sum of its parts, it’s a product of their interactions.” *



Your local optimization

Your system

“Until managers take into account the systemic nature of their organizations, most of their efforts to improve their performance are doomed to failure.” *

*) Dr. Russell Ackoff

Challenges of Building Systems

Building the right thing

- How aligned is our solution to business / user needs?
- Have we understood the problem?
- Do we share the same common understanding?

Effectiveness

Building the thing right

- How efficient are our engineering practices?
- How fast can we deliver changes?
- How easy and fast can we change and adapt?

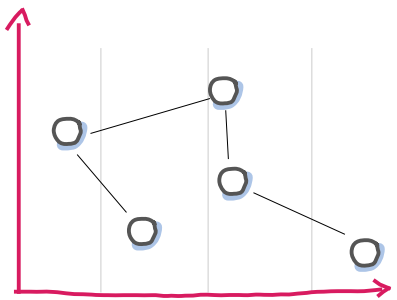
Efficiency

"Doing the wrong thing right is not nearly as good as doing the right thing wrong"

Dr. Russell Ackoff

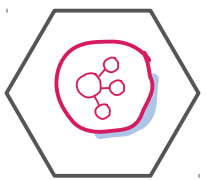
Three Perspectives to Build Adaptive Systems

1. Business-Strategy



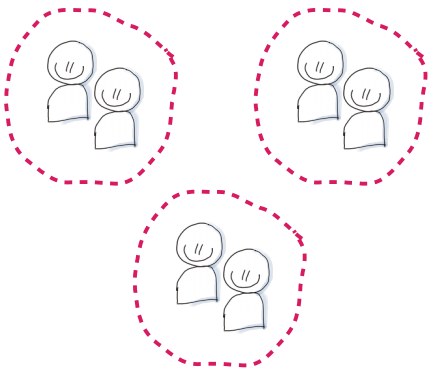
w/ Wardley Mapping

2. Software-Design/
-Architecture



w/ Domain-Driven Design

3. Team-Organization



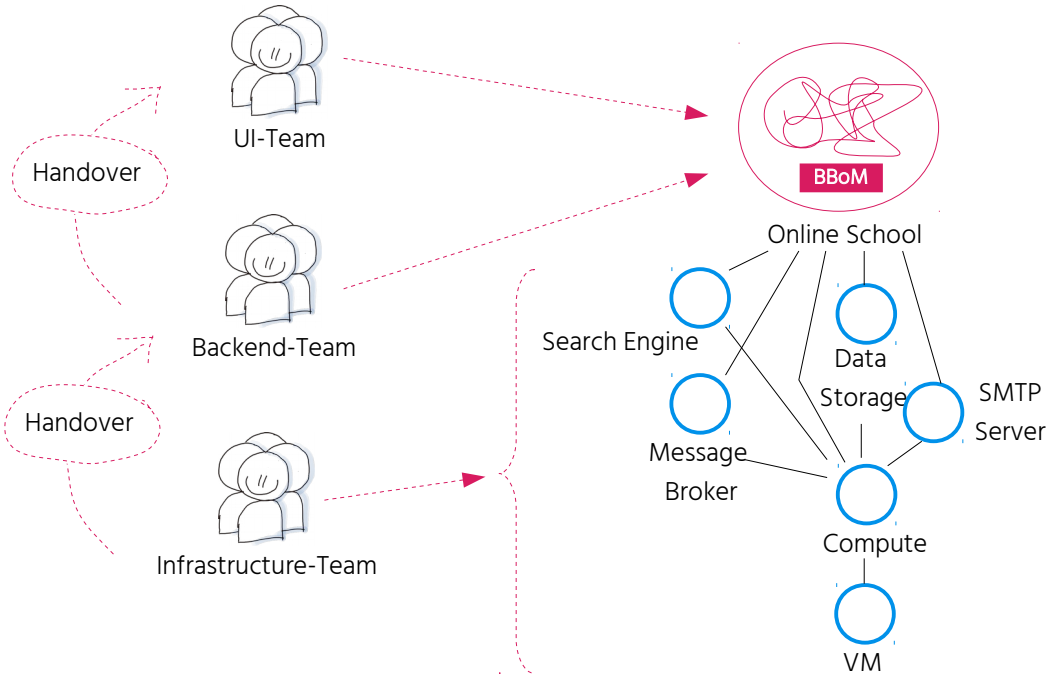
w/ Team Topologies

Evolving a Legacy System

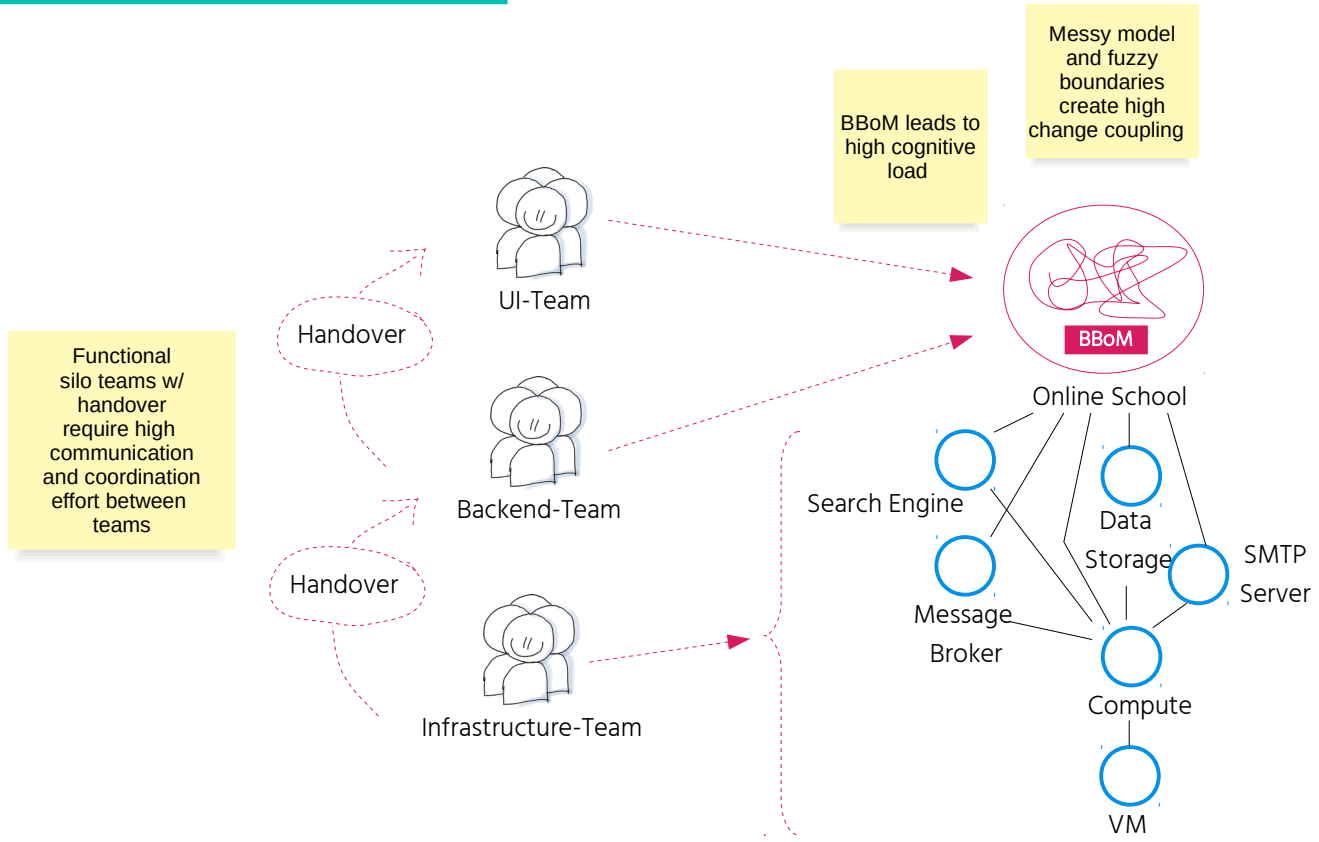
Source: https://www.food-management.com

Their Current Challenges

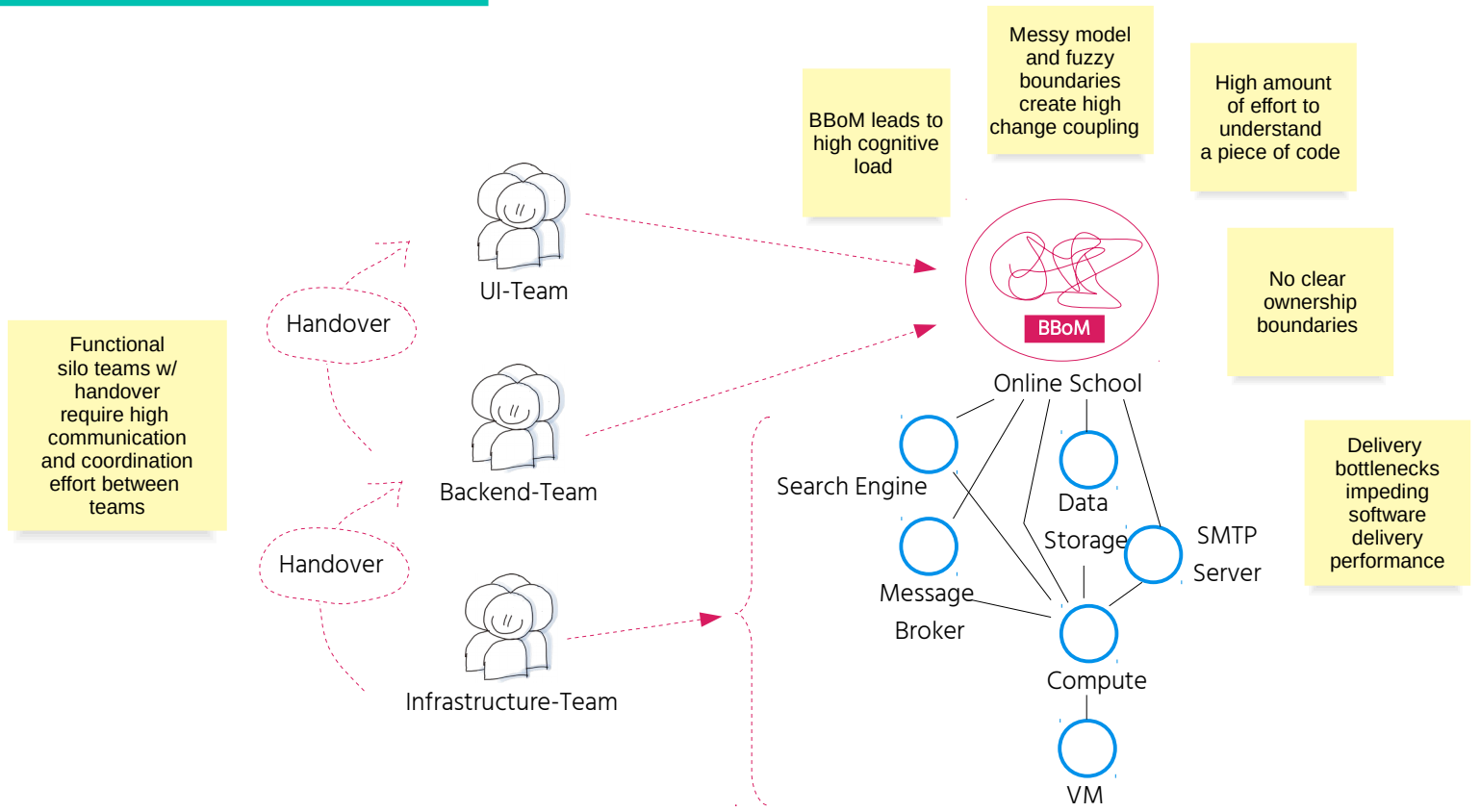
Functional silo teams w/ handover require high communication and coordination effort between teams



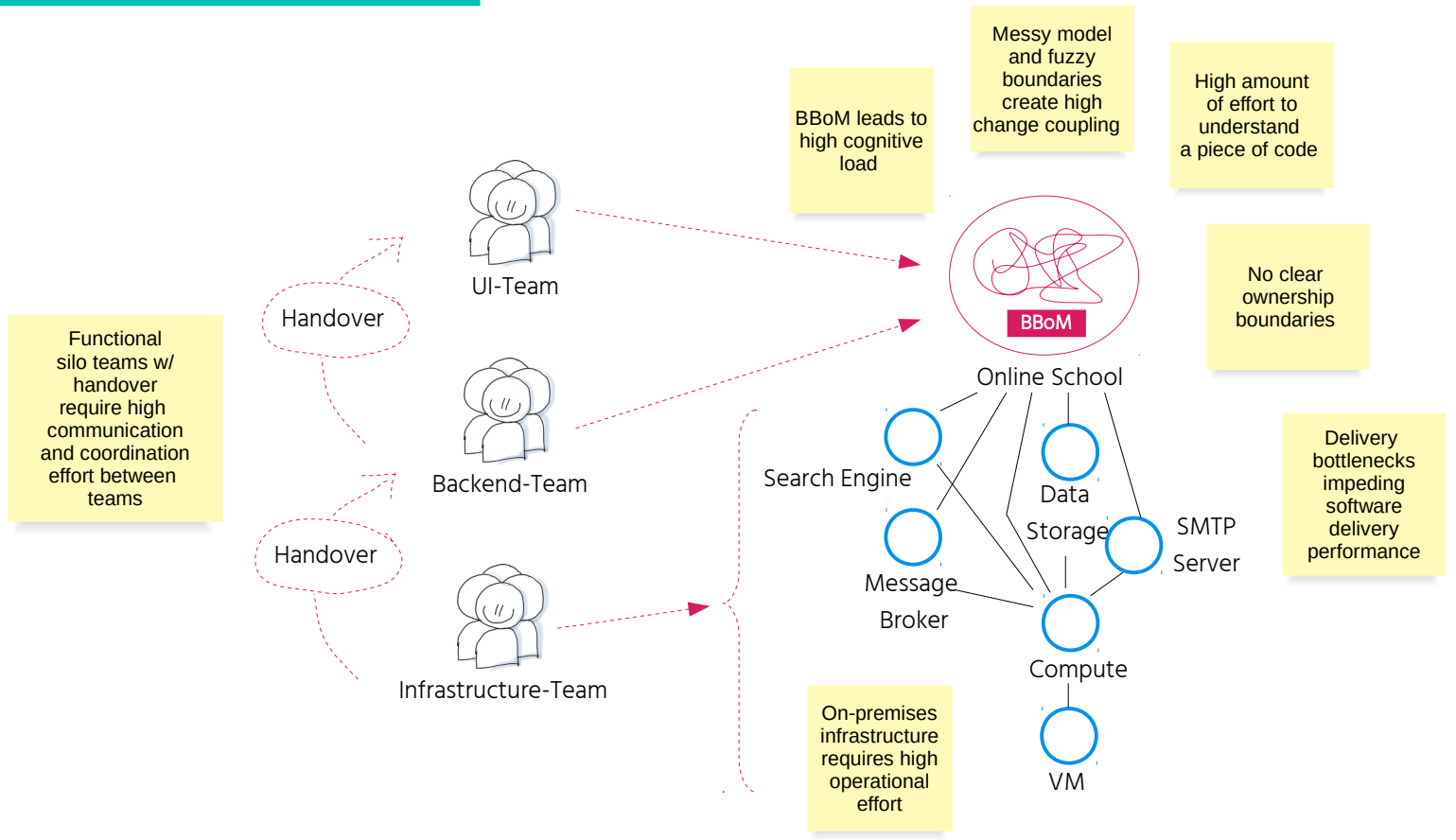
Their Current Challenges



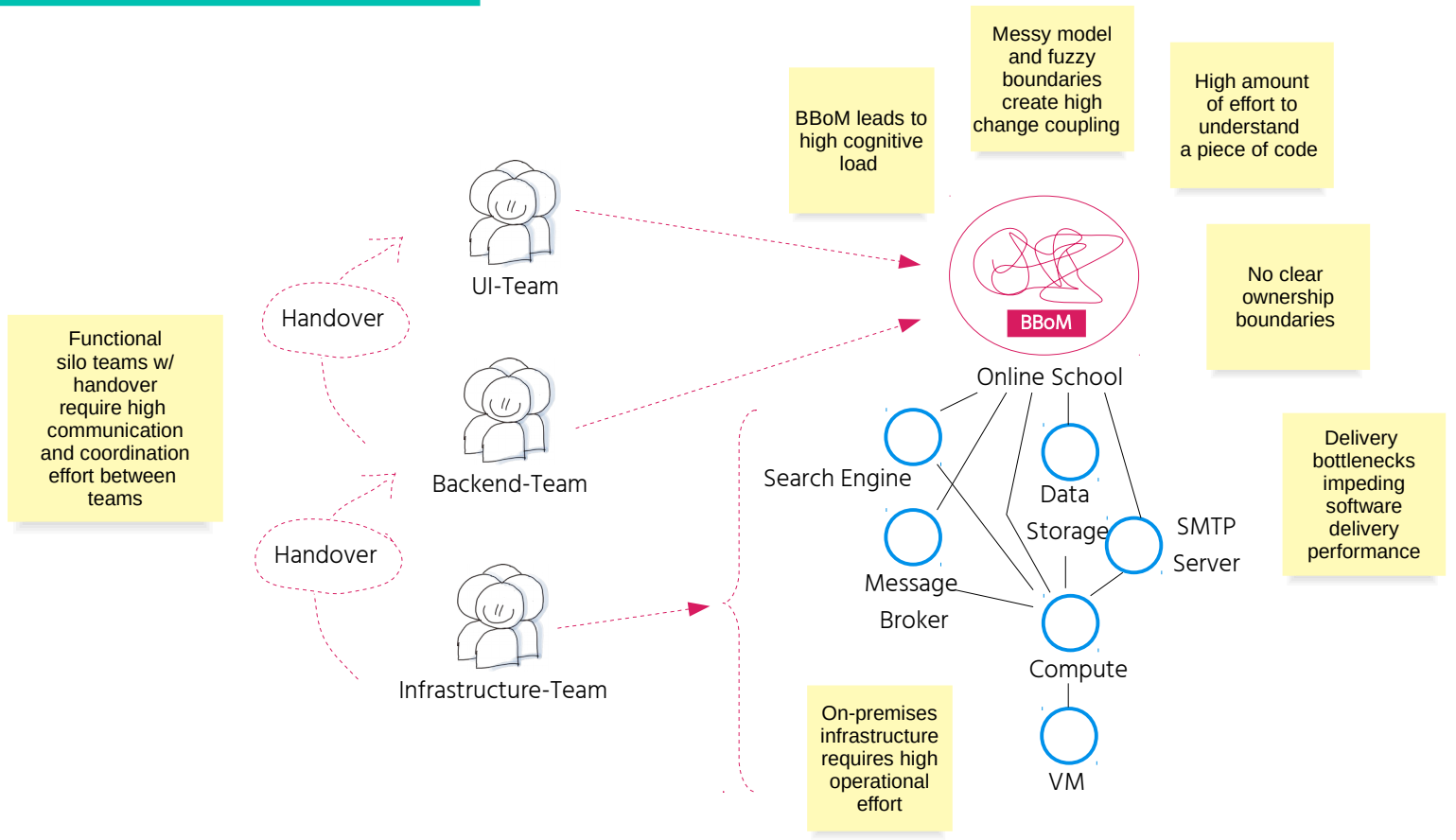
Their Current Challenges



Their Current Challenges



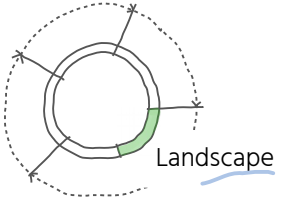
Their Current Challenges

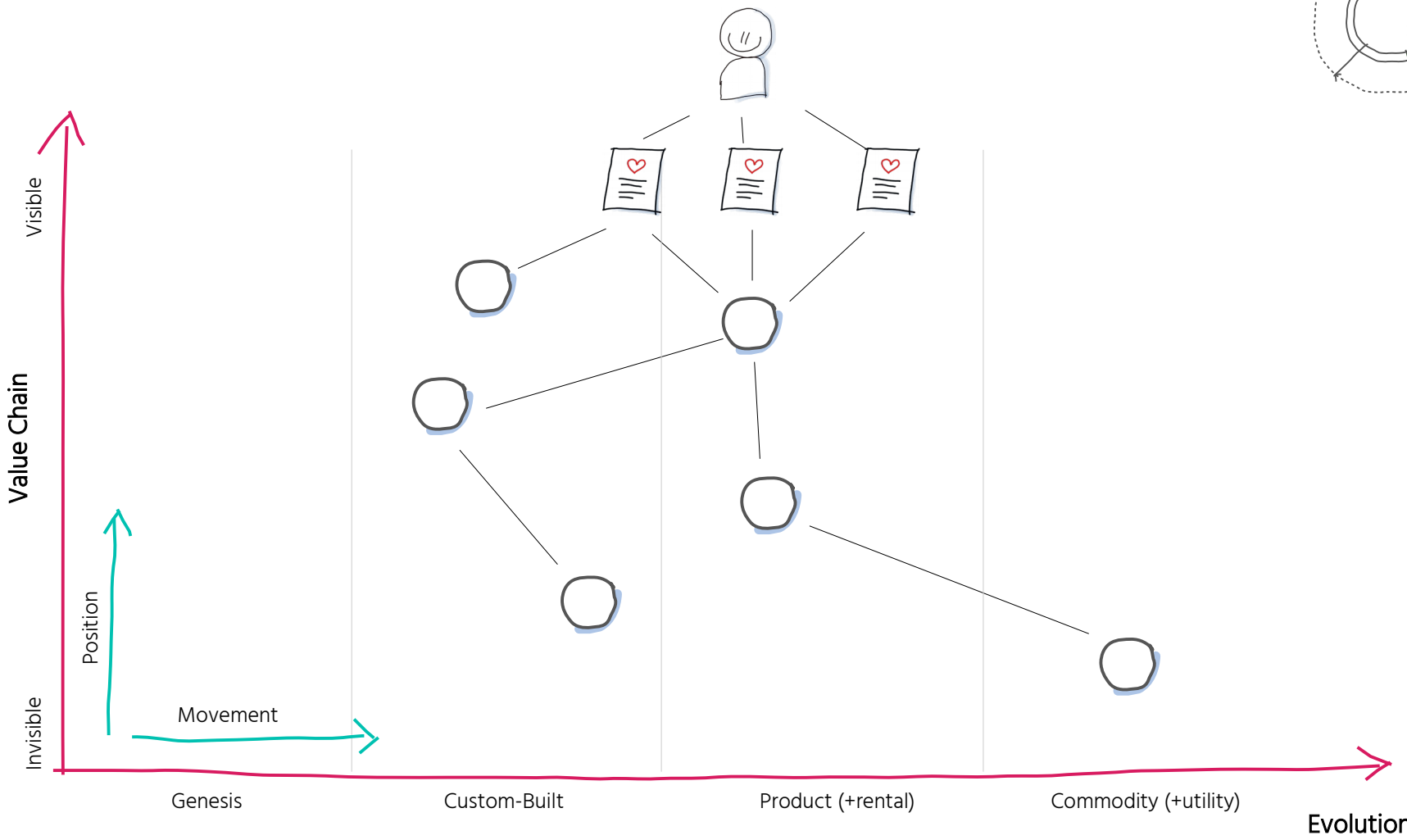


"Local optimization does not improve the performance of the whole."

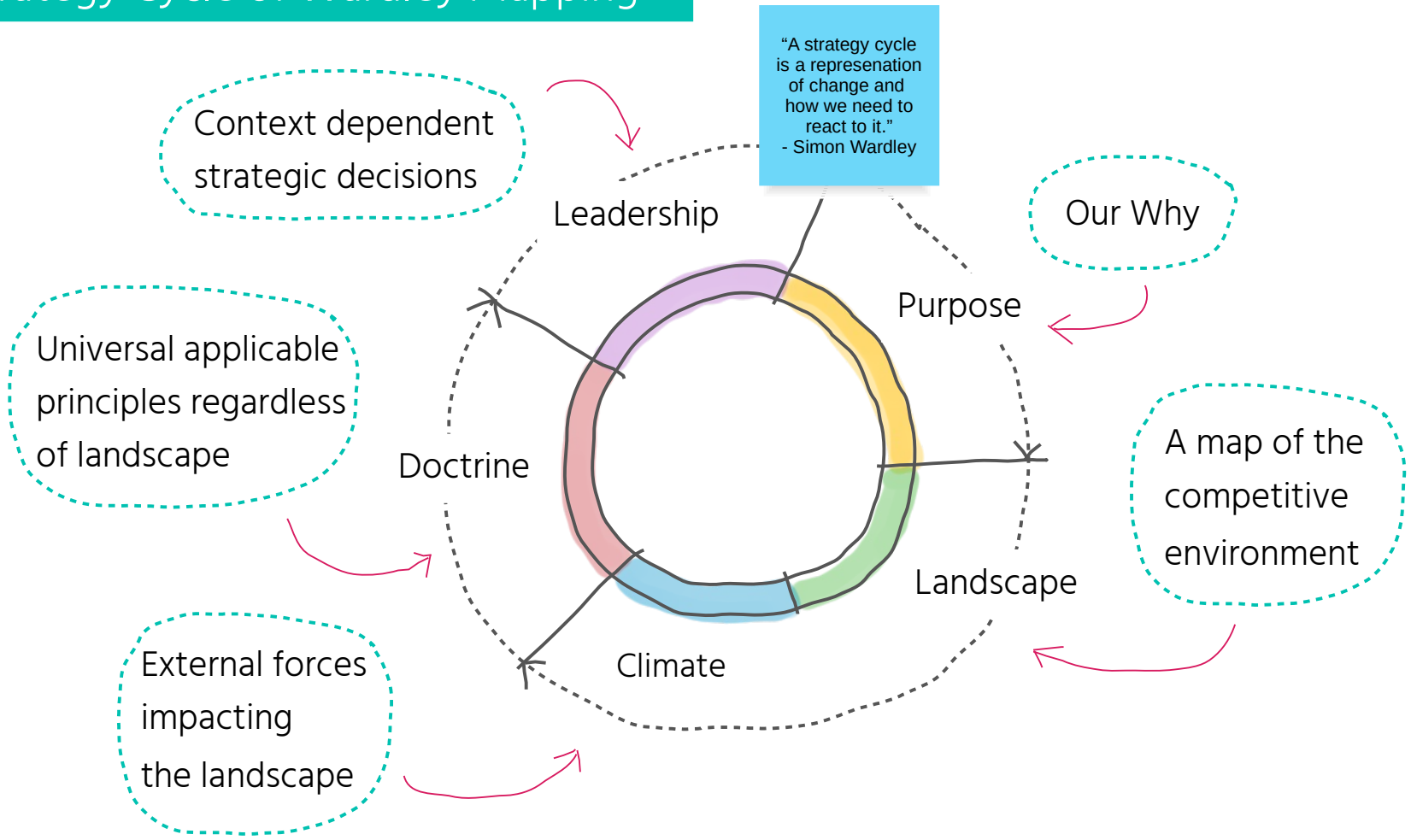
Dr. Russell Ackoff

Visualizing the Landscape w/ a Wardley Map





The Strategy Cycle of Wardley Mapping



The Strategy Cycle of Wardley Mapping

"A strategy cycle is a represenation of change and how we need to react to it."
- Simon Wardley

Providing high-quality education for junior students everywhere and help teachers to engage with their students online.

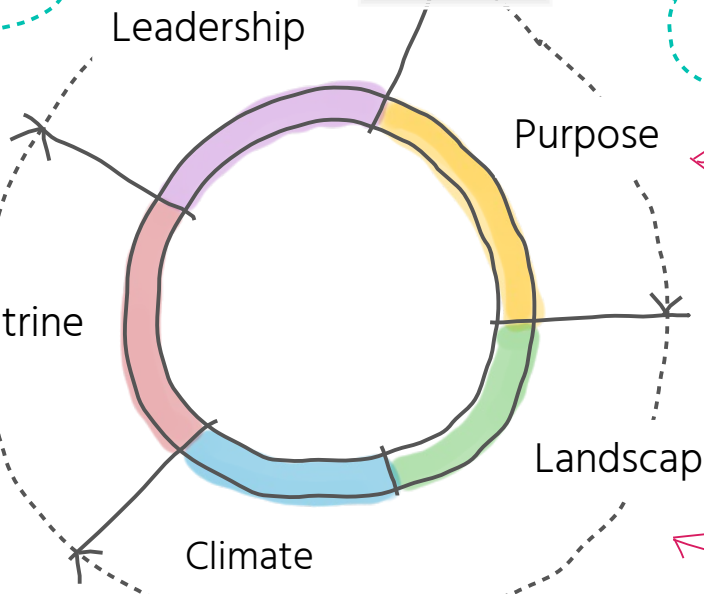
Context dependent strategic decisions

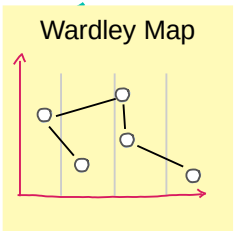
Our Why

Universal applicable principles regardless of landscape

A map of the competitive environment

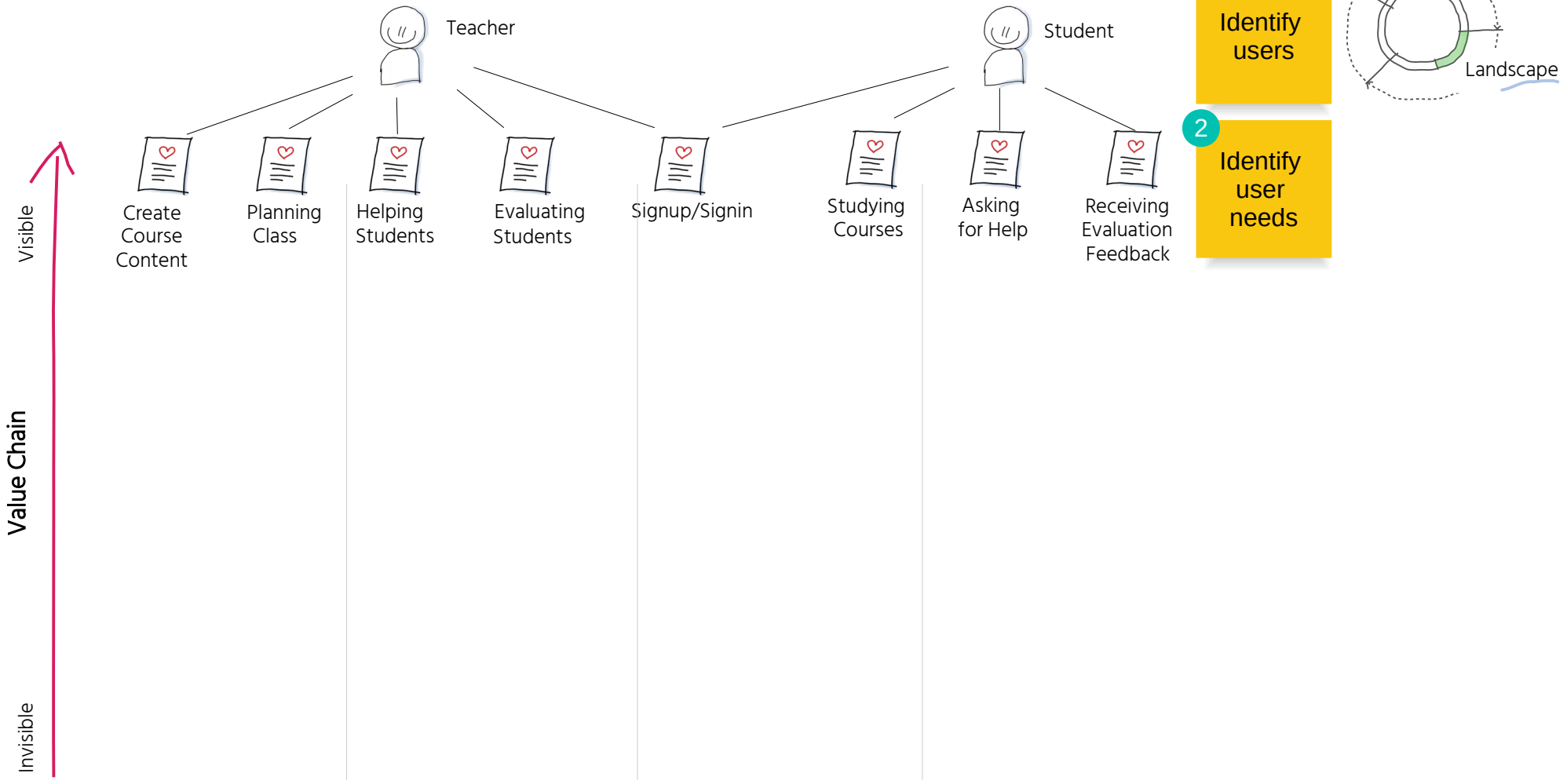
External forces impacting the landscape





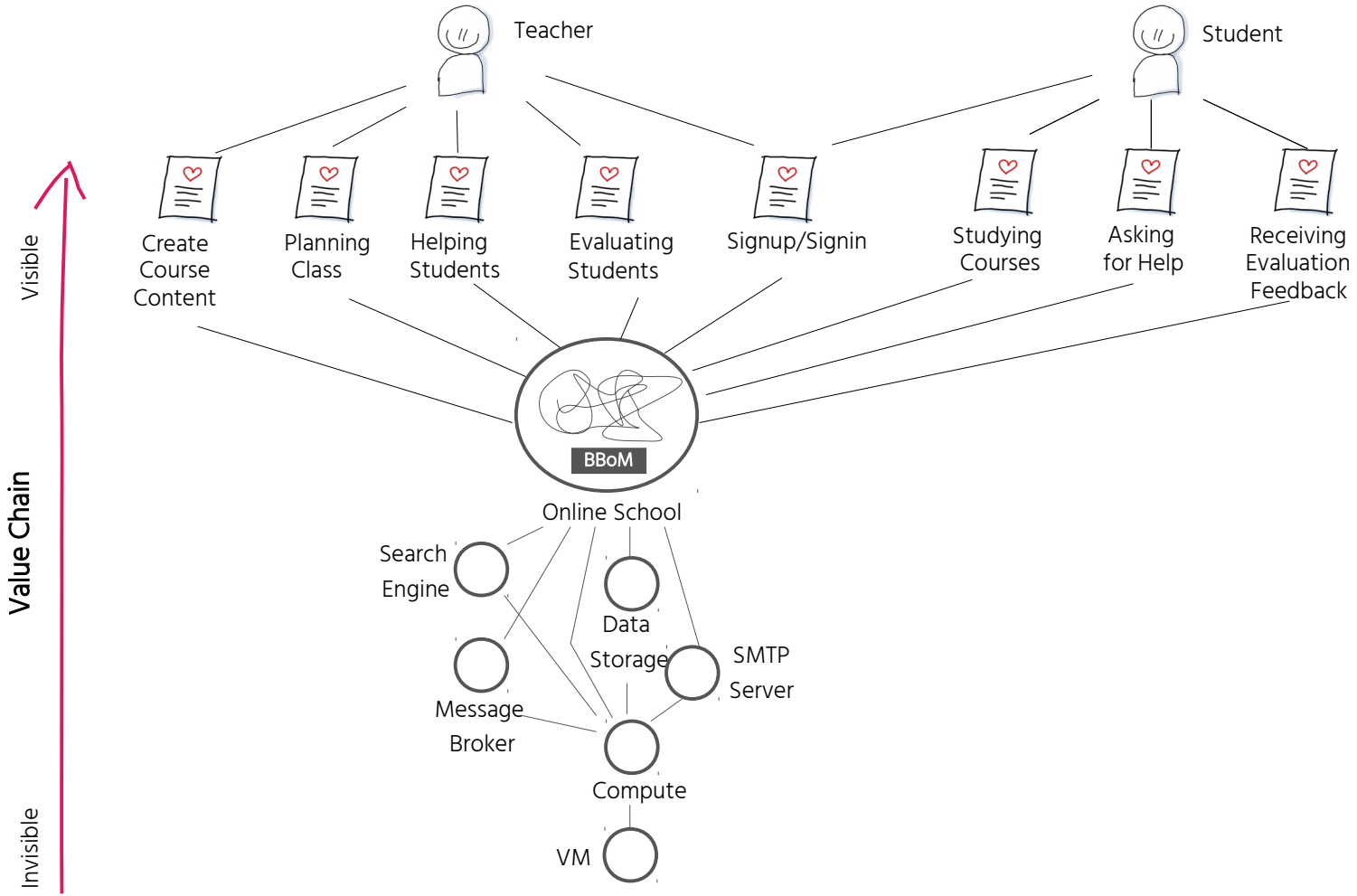
@suksr

The Landscape of the Current State

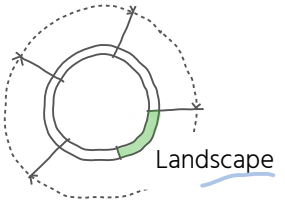


1. Identify users
2. Identify user needs

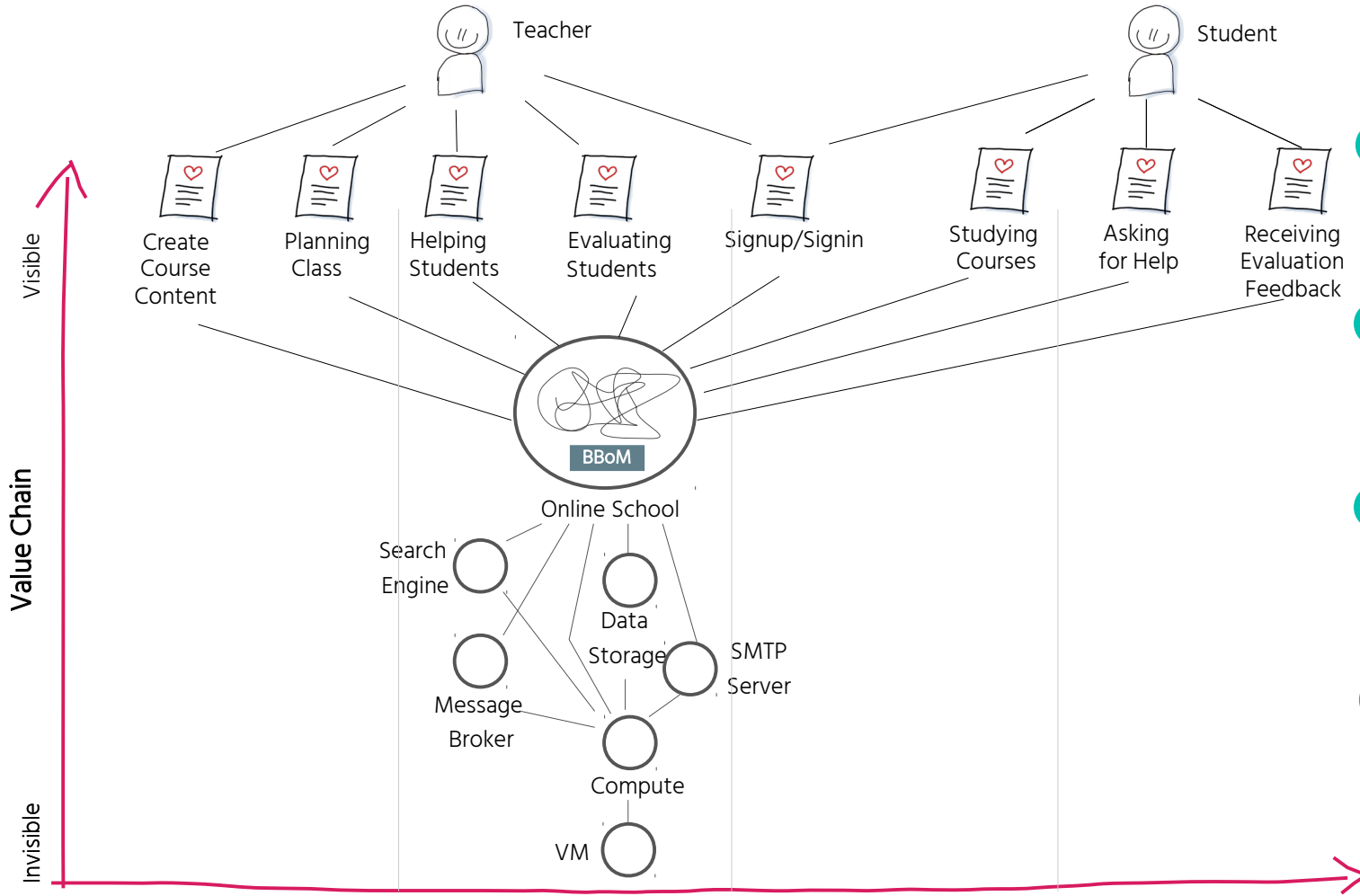
The Landscape of the Current State



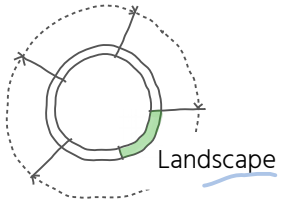
1. Identify users
2. Identify user needs
3. Identify components fulfilling user needs
4. Determine dependencies and position of the components in the value chain



The Landscape of the Current State

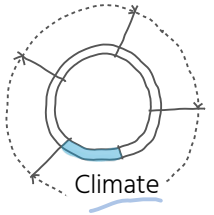


1. Identify users
2. Identify user needs
3. Identify components fulfilling user needs
4. Determine dependencies and position of the components in the value chain
5. Determine stage of evolution for every component



@suksr

The Climatic Patterns (extract)



Demand Competition

Supply Competition

Everything evolves

"Understanding climatic patterns is important when anticipating change."
- Simon Wardley

Visible

Value Chain

Invisible

Past

Current

Future

Movement

Genesis

Custom-Built

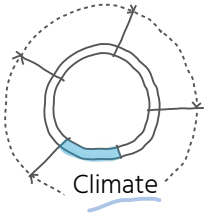
Product (+rental)

Commodity (+utility)

Evolution

@suksr

The Climatic Patterns (extract)



"Understanding climatic patterns is important when anticipating change."
- Simon Wardley

Characteristics change

Uncharted

Industrialised

Visible

Invisible

Value Chain

Uncharted	Forming market	Growing market	Mature market
Undefined market	Learning on use	Learning on operation	Known / accepted
Uncertain	Increasing understanding	Increasing education	Stable
Unpredictable	Slowly increasing consumption	Rapidly increasing consumption	Widespread and stabilising
Rare	Rapid increases in learning	Rapid increase in use	Commonly understood
Poorly understood	Domain of experts	Increasing expectation of use	Ordered
Chaotic			

Past

Current

Future

Movement

Genesis

Custom-Built

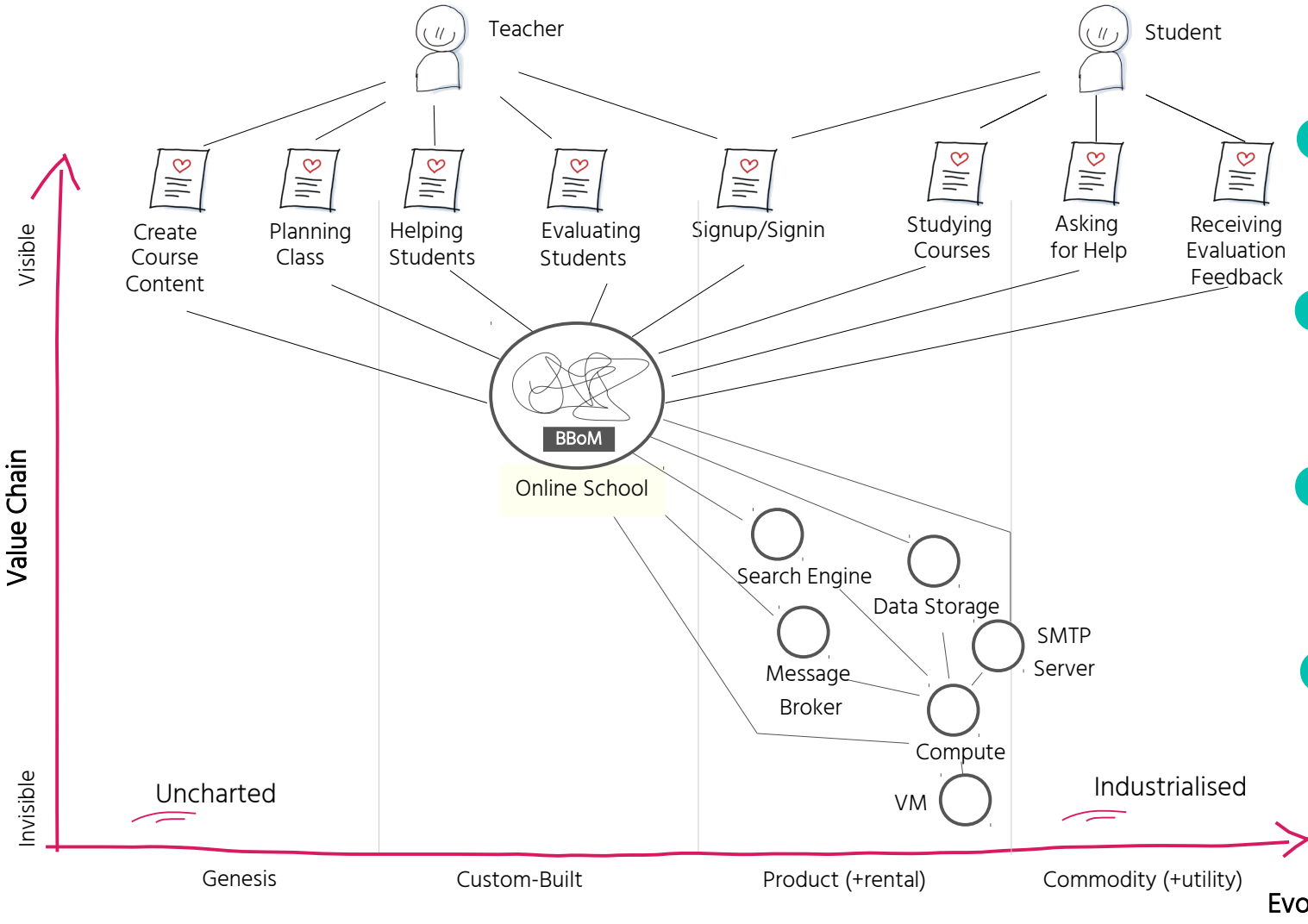
Product (+rental)

Commodity (+utility)

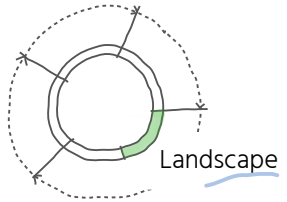
Evolution

@suksr

The Landscape of the Current State



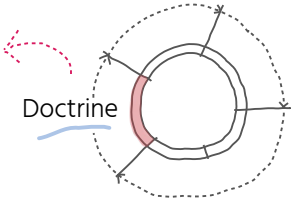
1. Identify users
2. Identify user needs
3. Identify components fulfilling user needs
4. Determine dependencies and position of the components in the value chain
5. Determine stage of evolution for every component

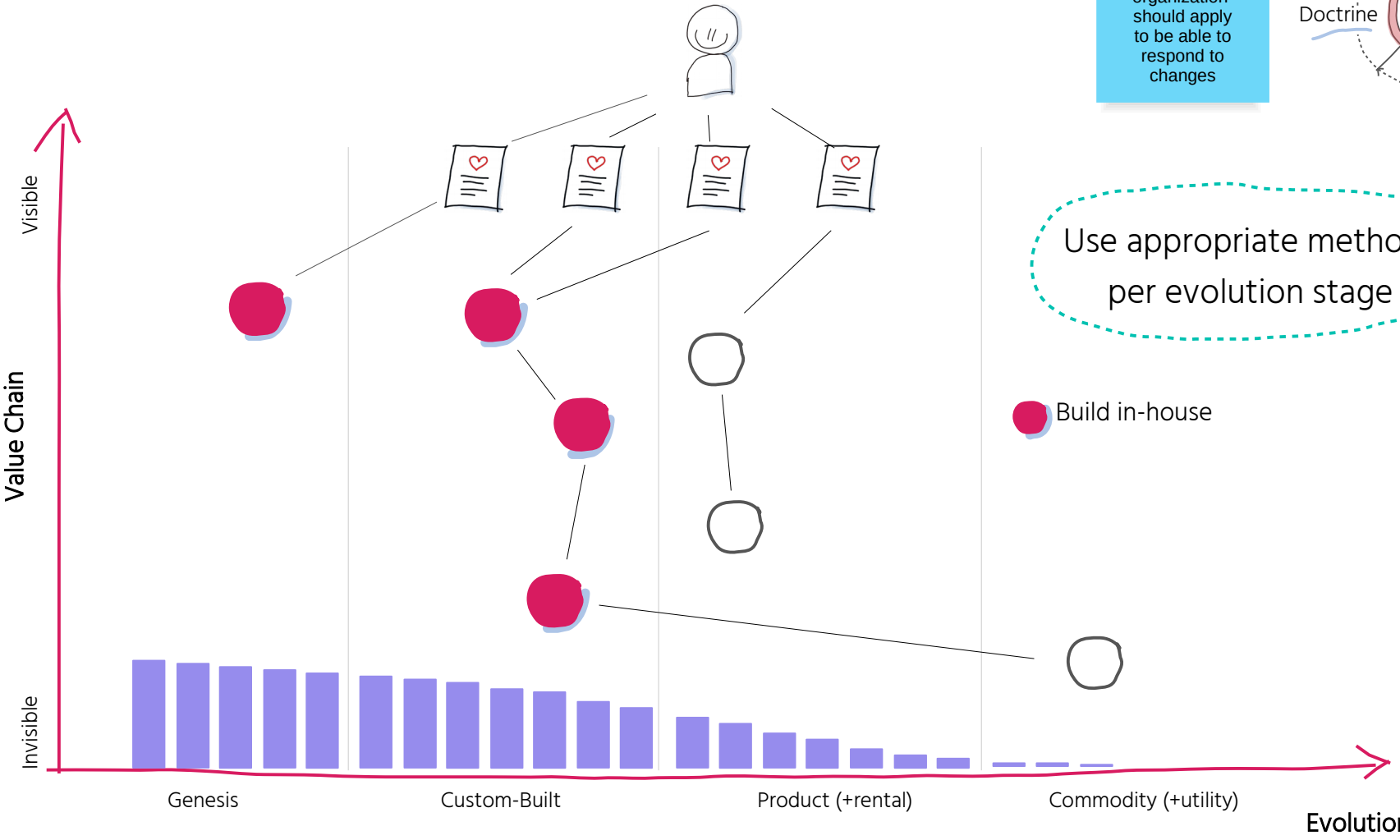


@suksr

Applying Doctrinal Principles

Universal principles an organization should apply to be able to respond to changes





Use appropriate methods per evolution stage

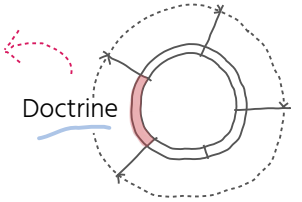
Build in-house

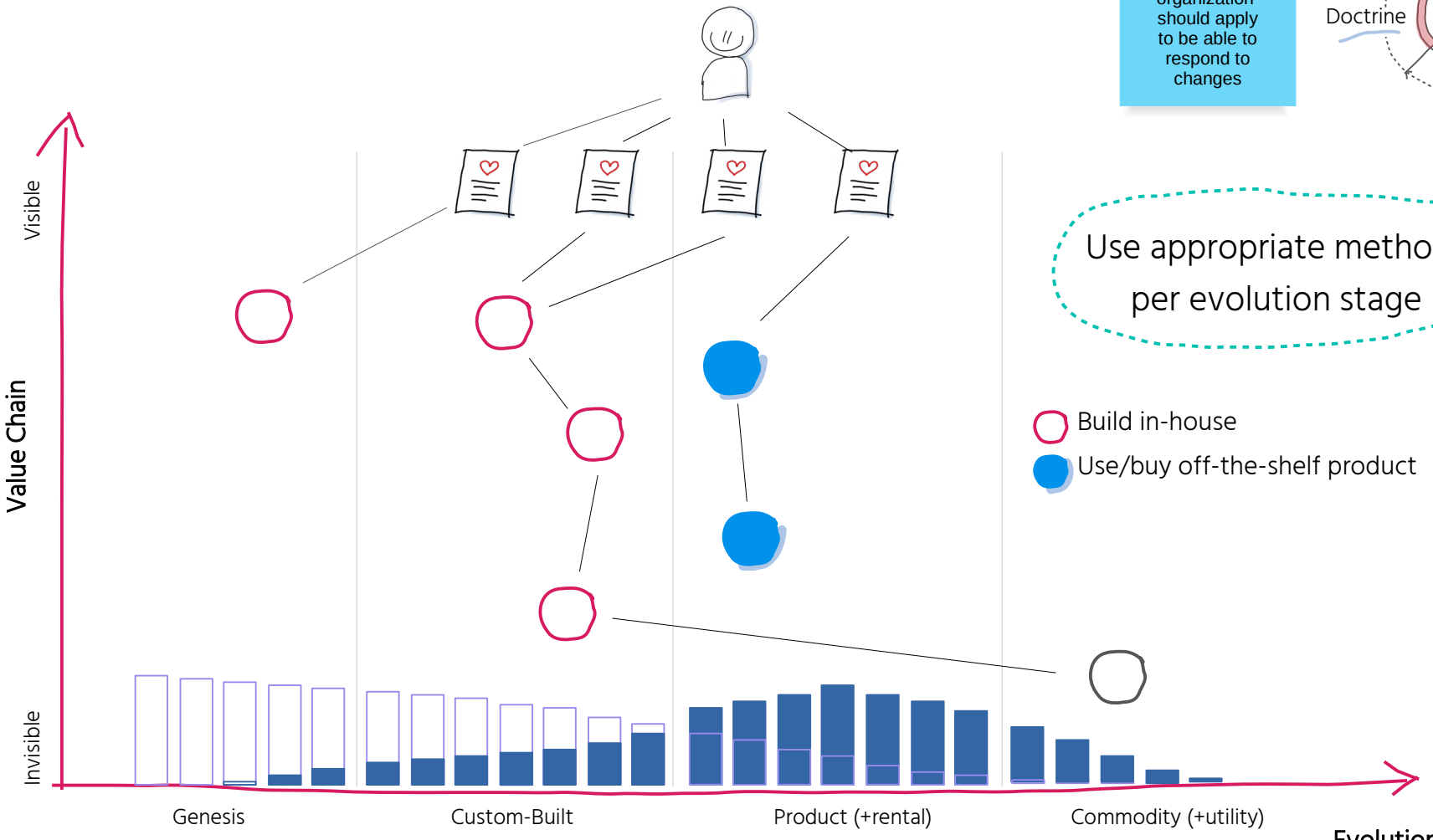
Agile

@suksr

Applying Doctrinal Principles

Universal principles an organization should apply to be able to respond to changes





Use appropriate methods per evolution stage

Build in-house

Use/buy off-the-shelf product

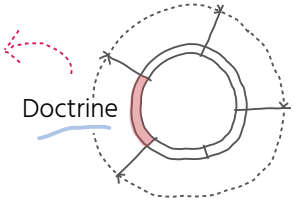
Agile

Lean

@suksr

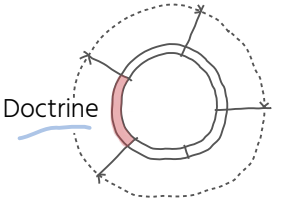
Applying Doctrinal Principles

Universal principles an organization should apply to be able to respond to changes



@suksr

Applying Doctrinal Principles

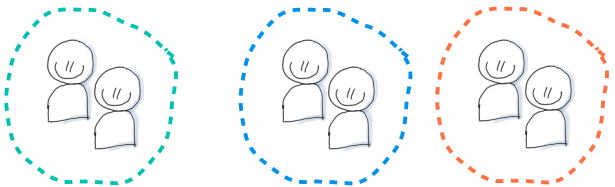


To optimize for flow of change from a team perspective requires ...

cross-functional, autonomous teams

no handover between teams

restricting high, on-going communication bandwidth between teams

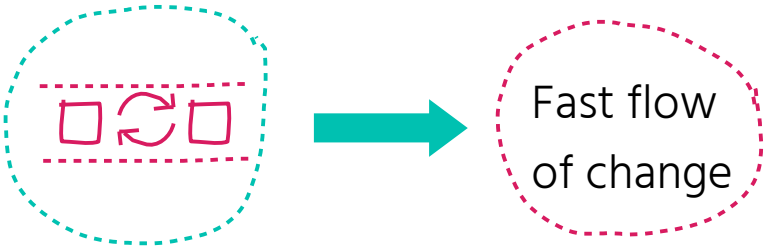


small, long-lived teams

minimizing cognitive load

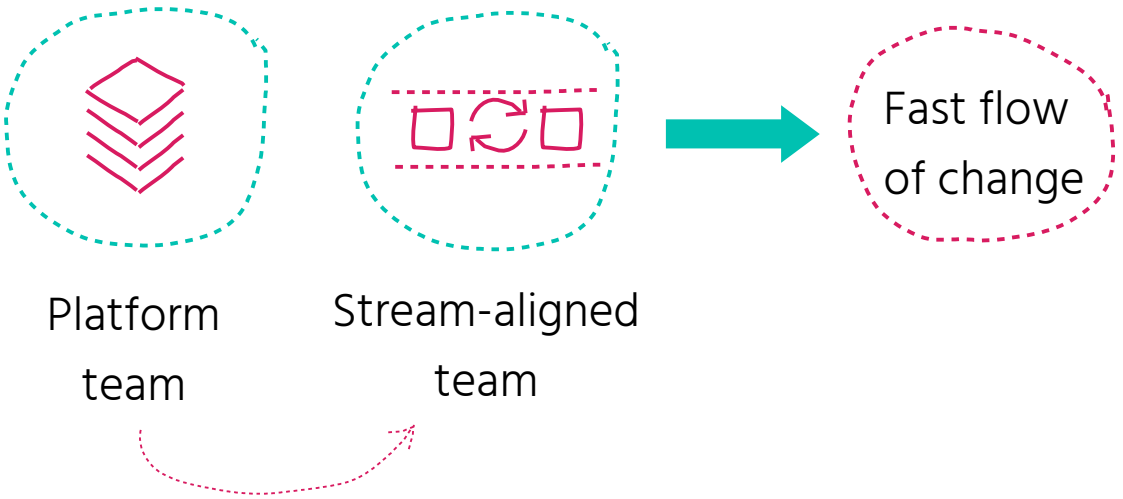
team ownership

Four Team Types of Team Topologies

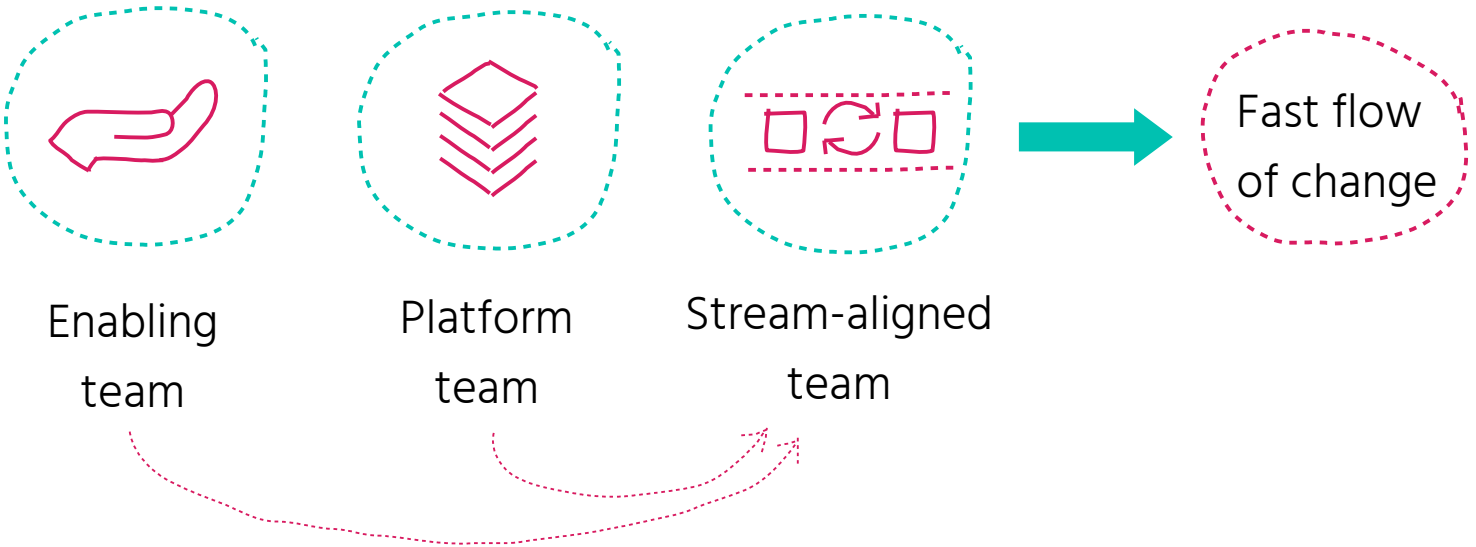


Stream-aligned
team

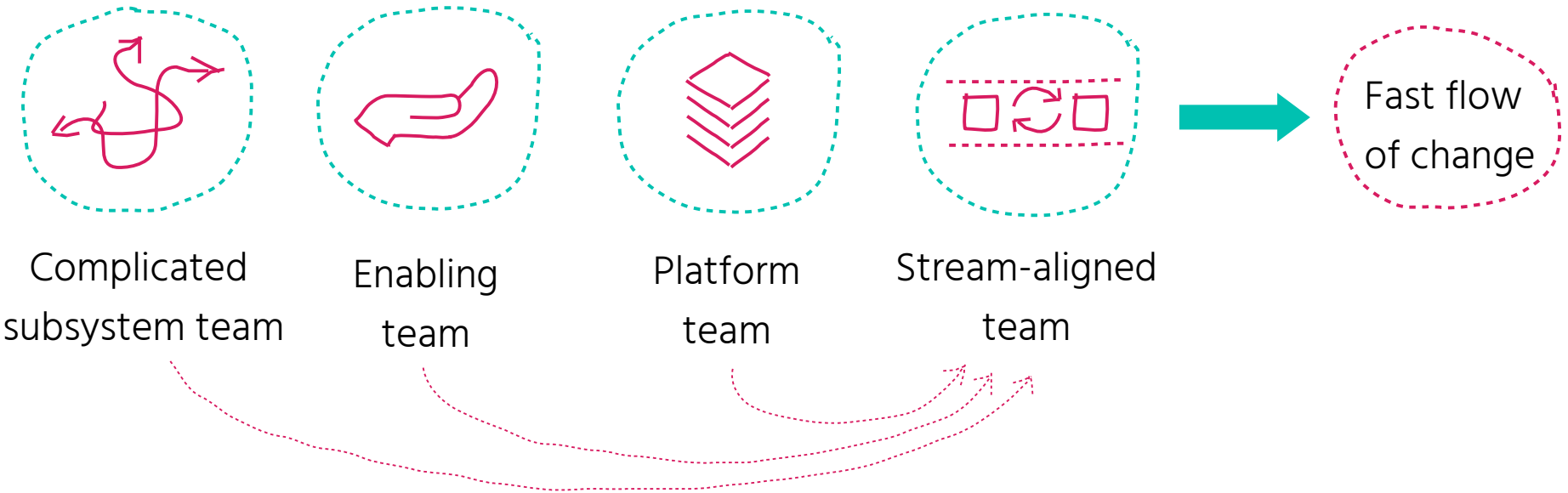
Four Team Types of Team Topologies



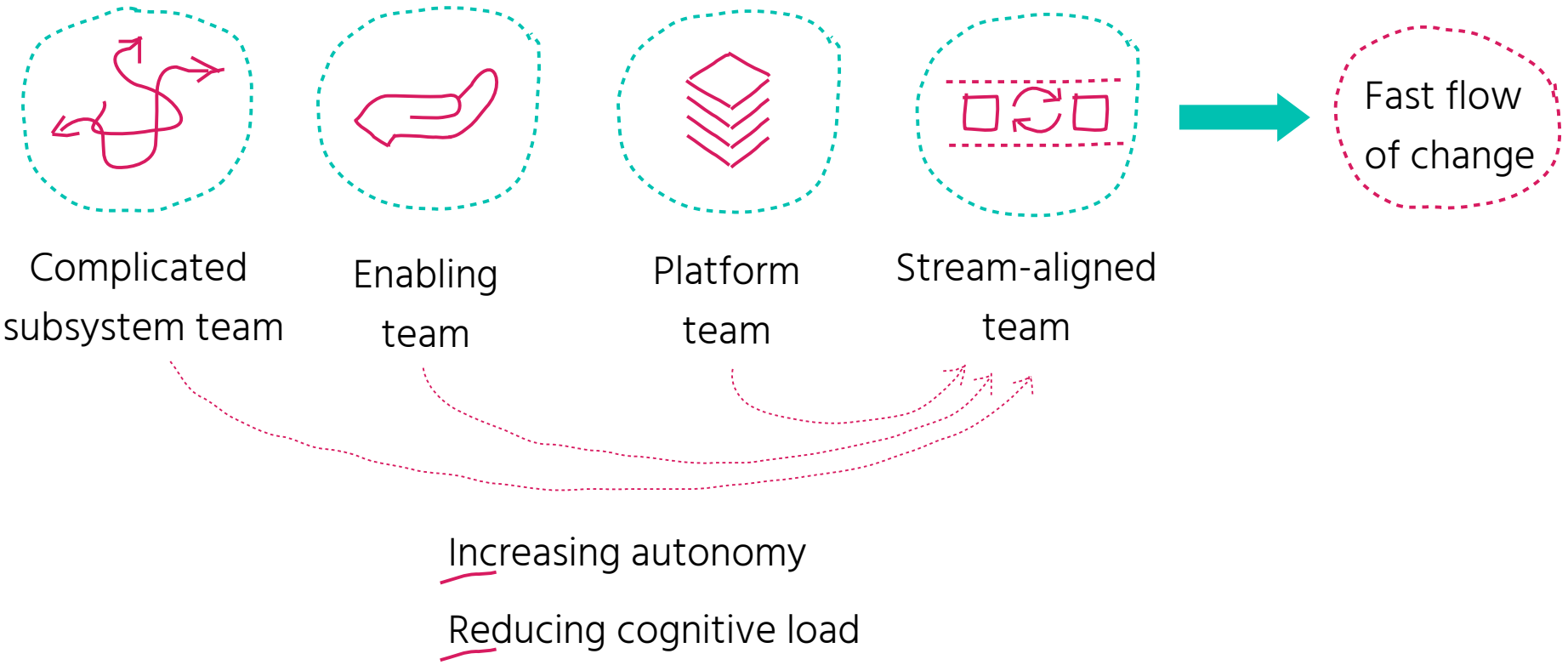
Four Team Types of Team Topologies



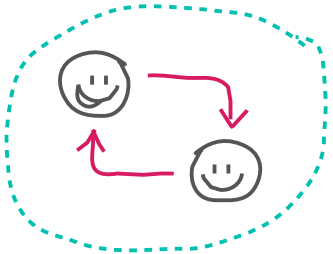
Four Team Types of Team Topologies



Four Team Types of Team Topologies



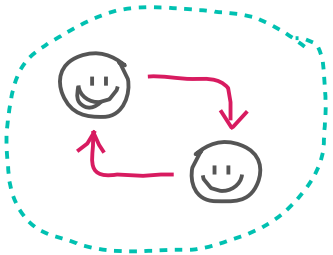
Three Interaction Modes



Collaboration

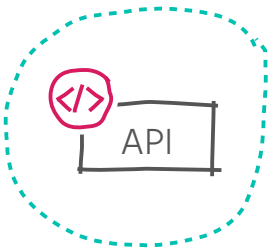
Rapid discovery

Three Interaction Modes



Collaboration

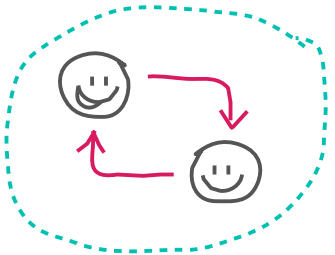
Rapid discovery



X-as-a-Service

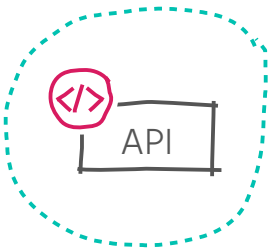
Predictable delivery

Three Interaction Modes



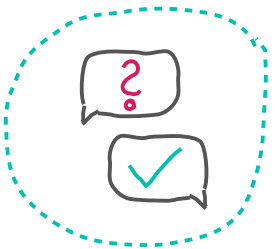
Collaboration

Rapid discovery



X-as-a-Service

Predictable delivery

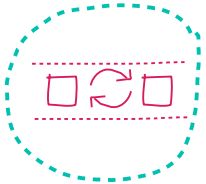


Facilitating

Active help

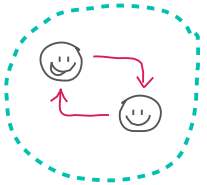
Team Topologies

Stream-aligned team

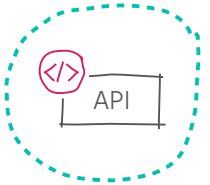


Platform team

Collaboration



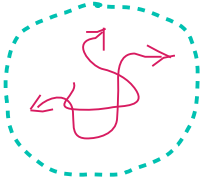
X-as-a-Service



Facilitating

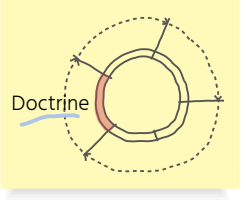
Enabling team

Complicated subsystem team

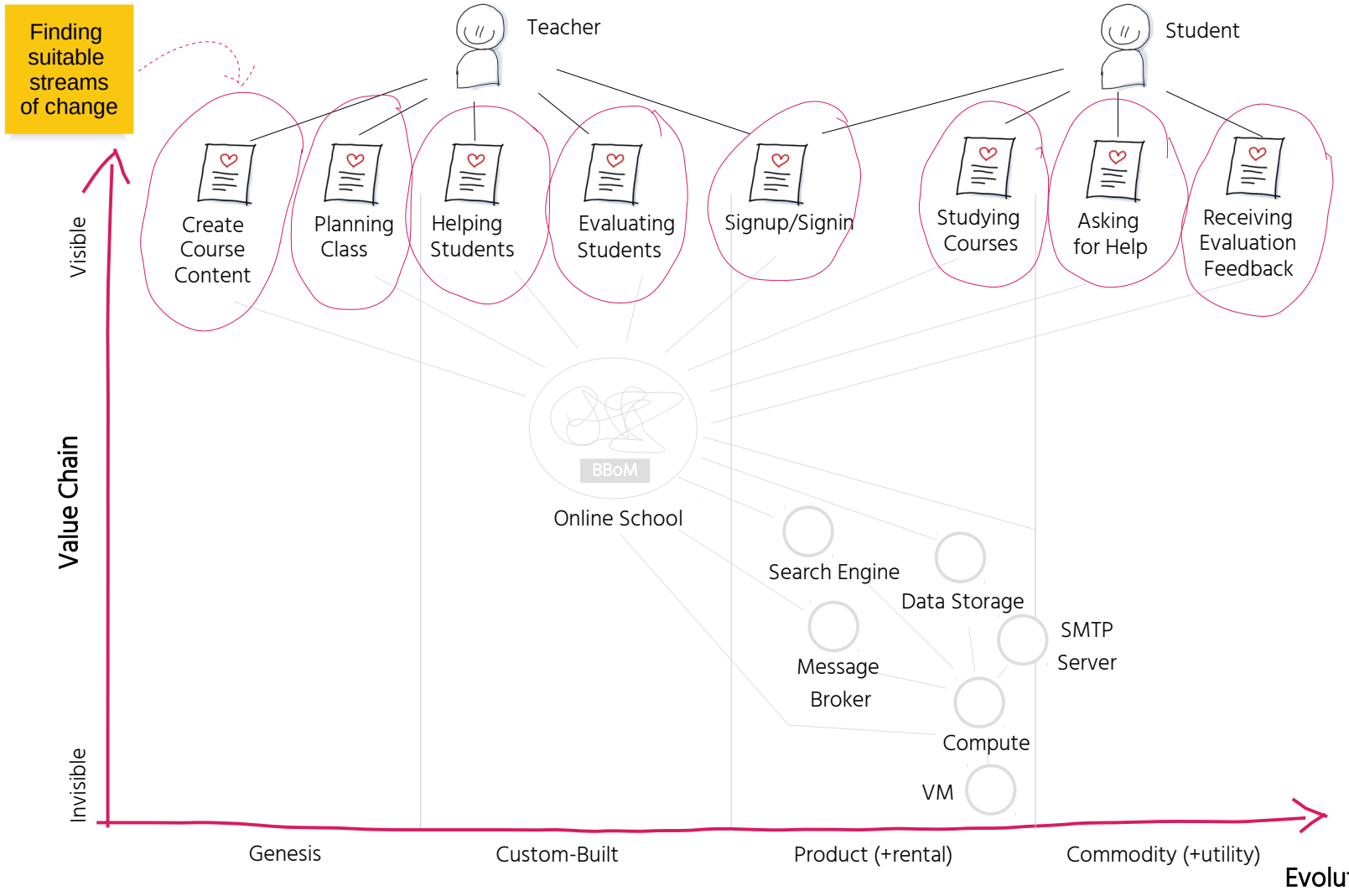


Promotes organizational effectiveness

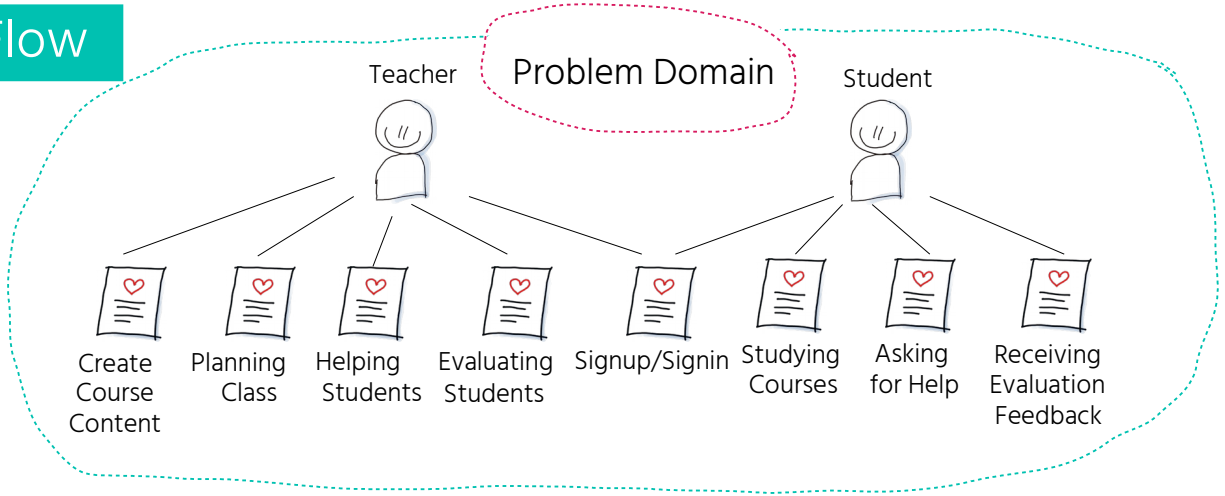
Helps to apply Wardley's Doctrinal Principles



Architecture For Flow



Architecture For Flow



Understanding the problem domain and partitioning it into subdomains

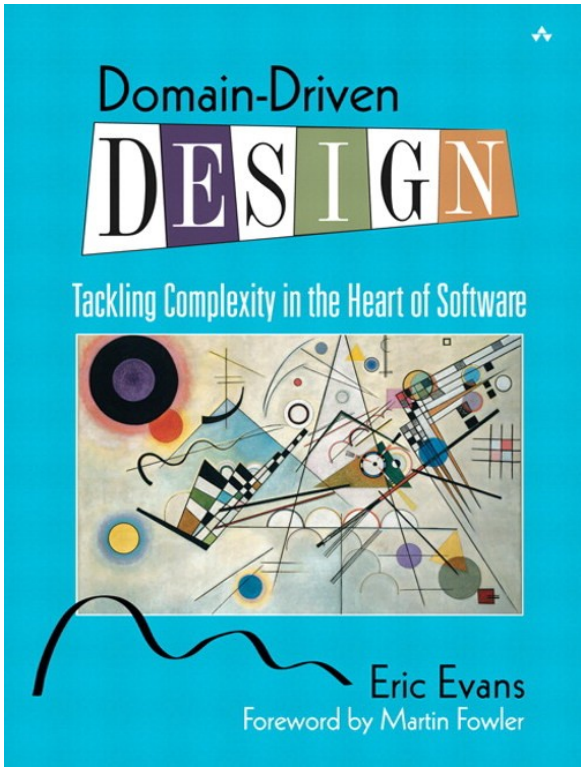
Finding suitable streams of change

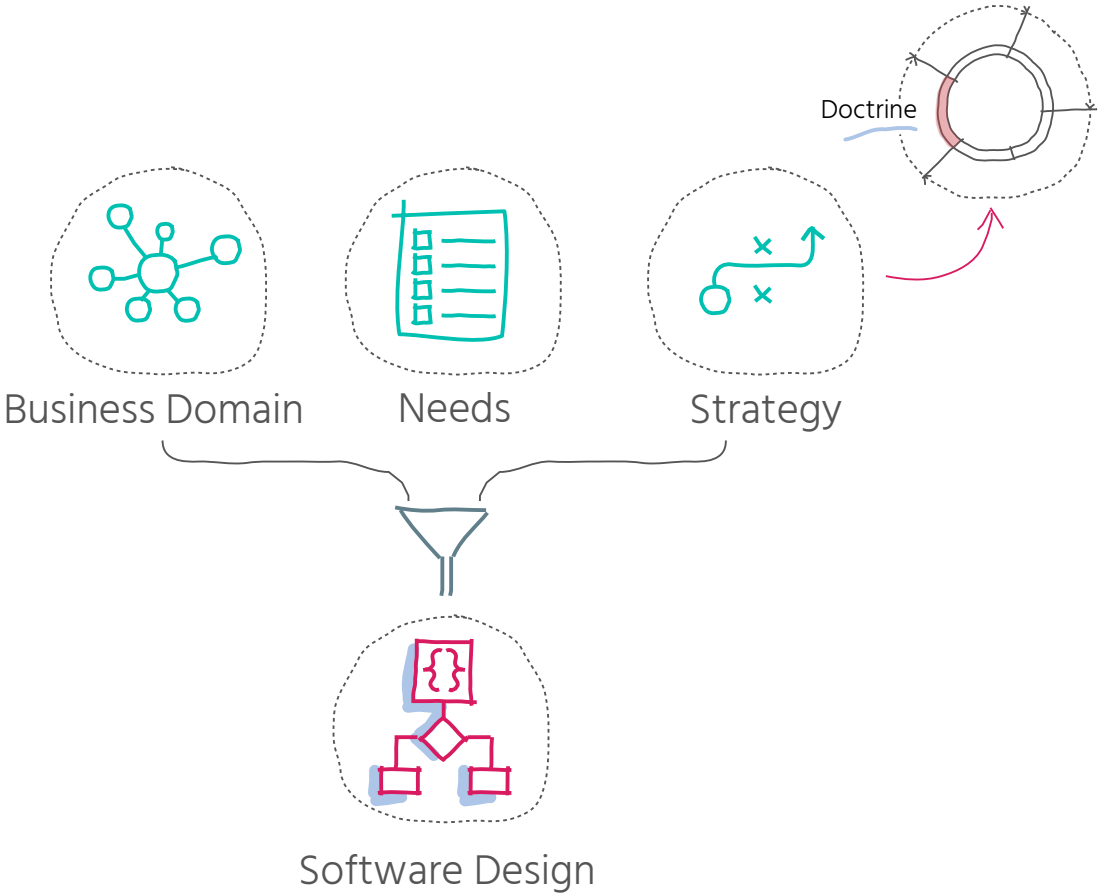
Visible

Value Chain

Invisible

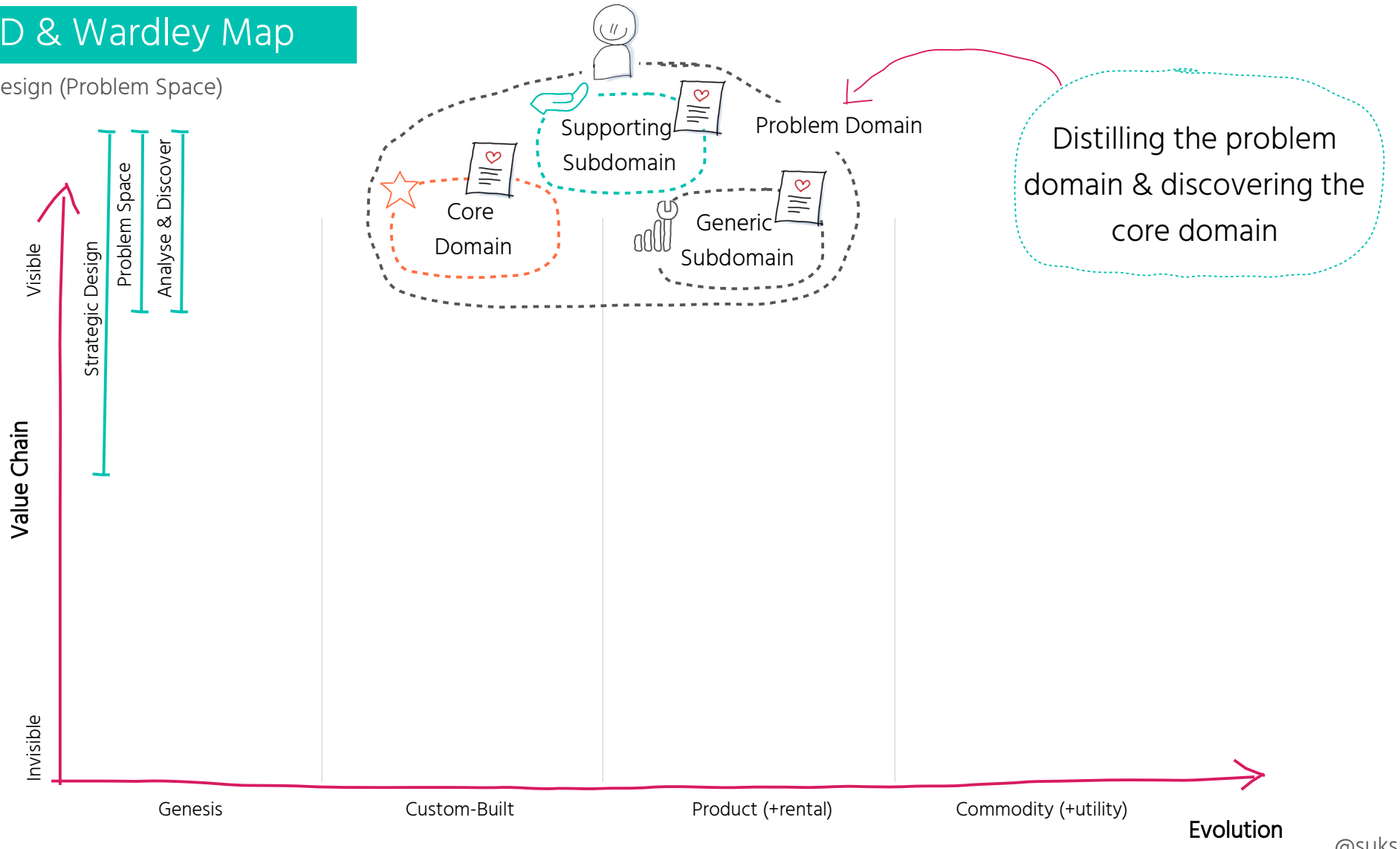
Domain-Driven Design (DDD)





DDD & Wardley Map

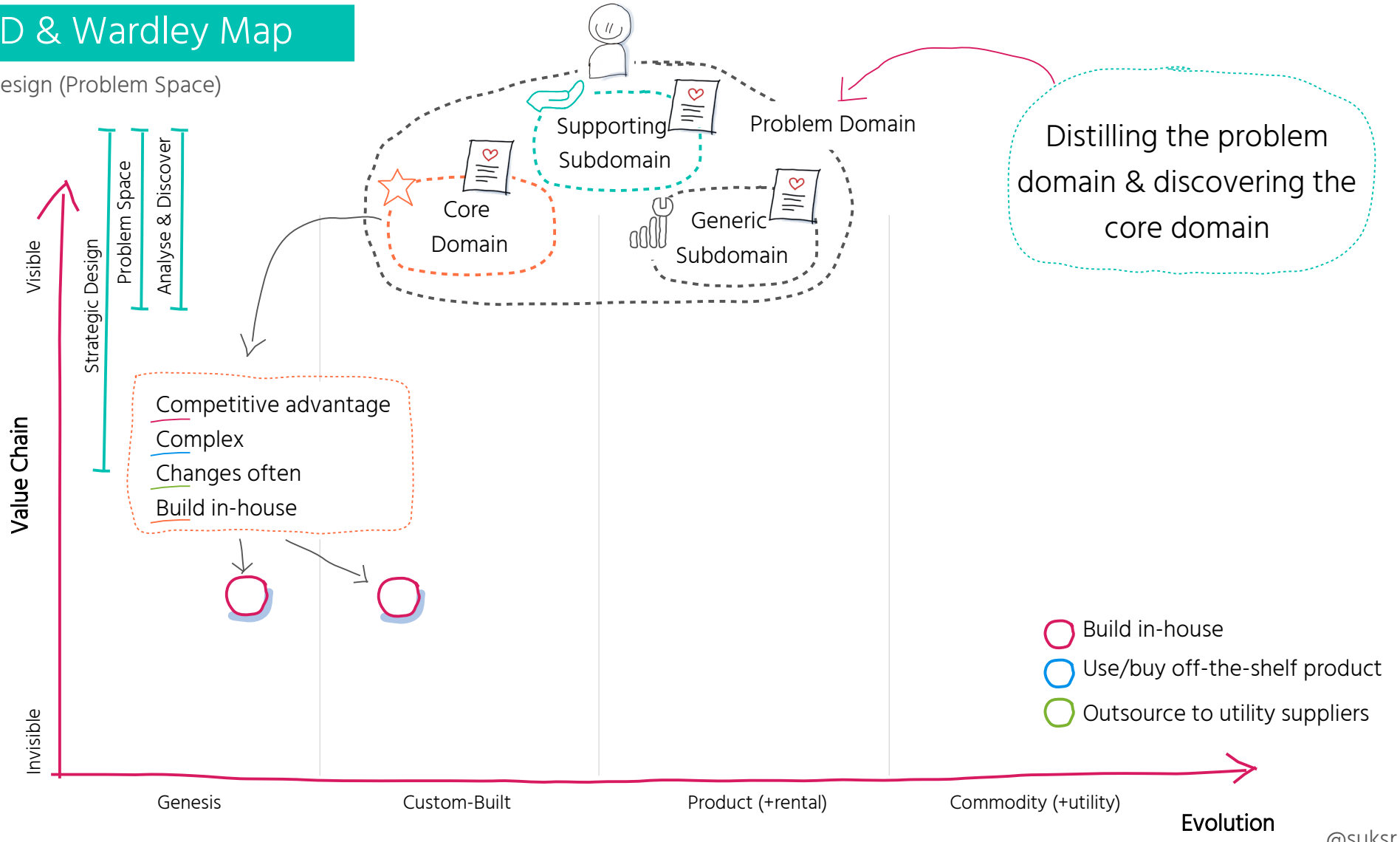
Strategic Design (Problem Space)



@suksr

DDD & Wardley Map

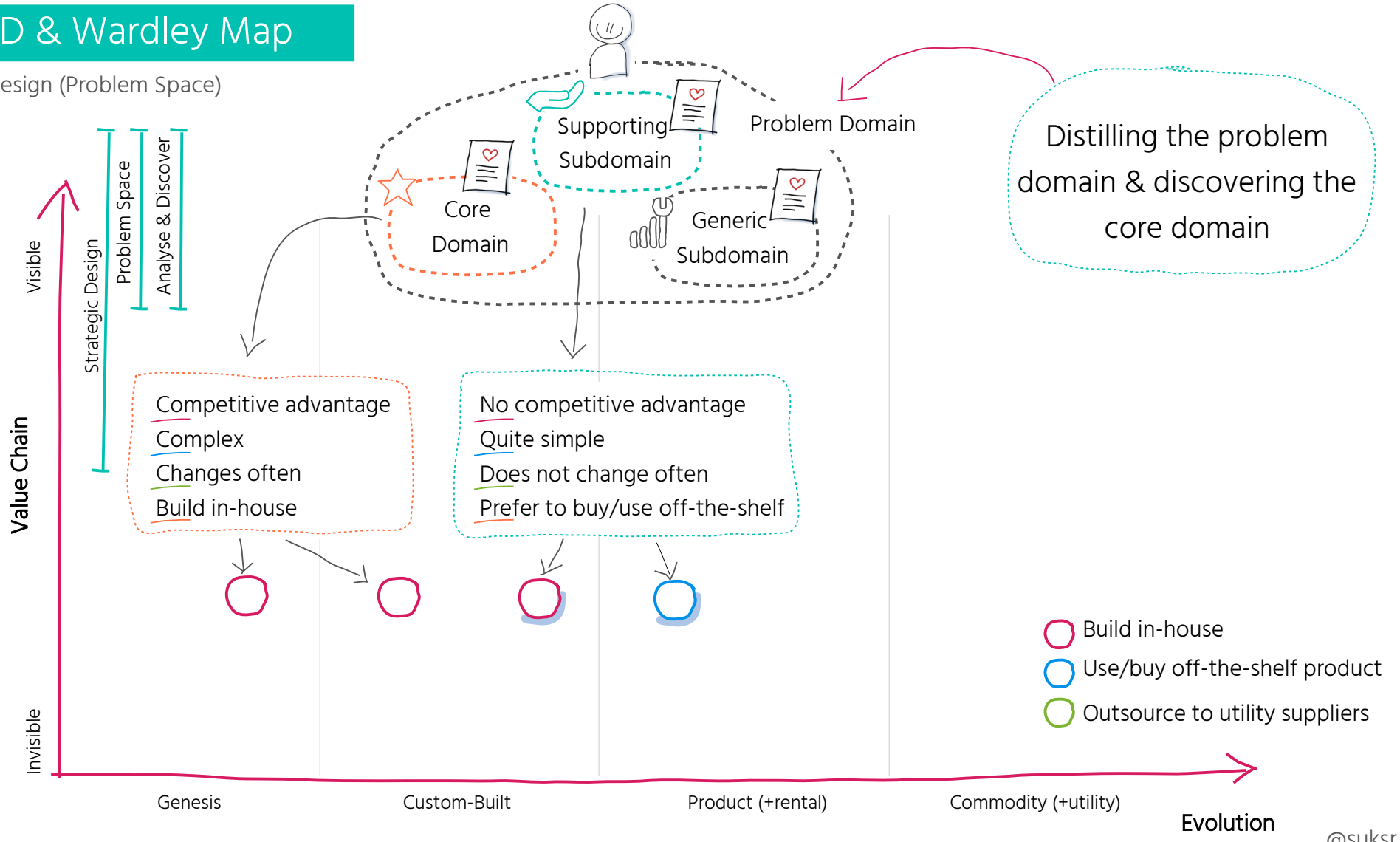
Strategic Design (Problem Space)



@suksr

DDD & Wardley Map

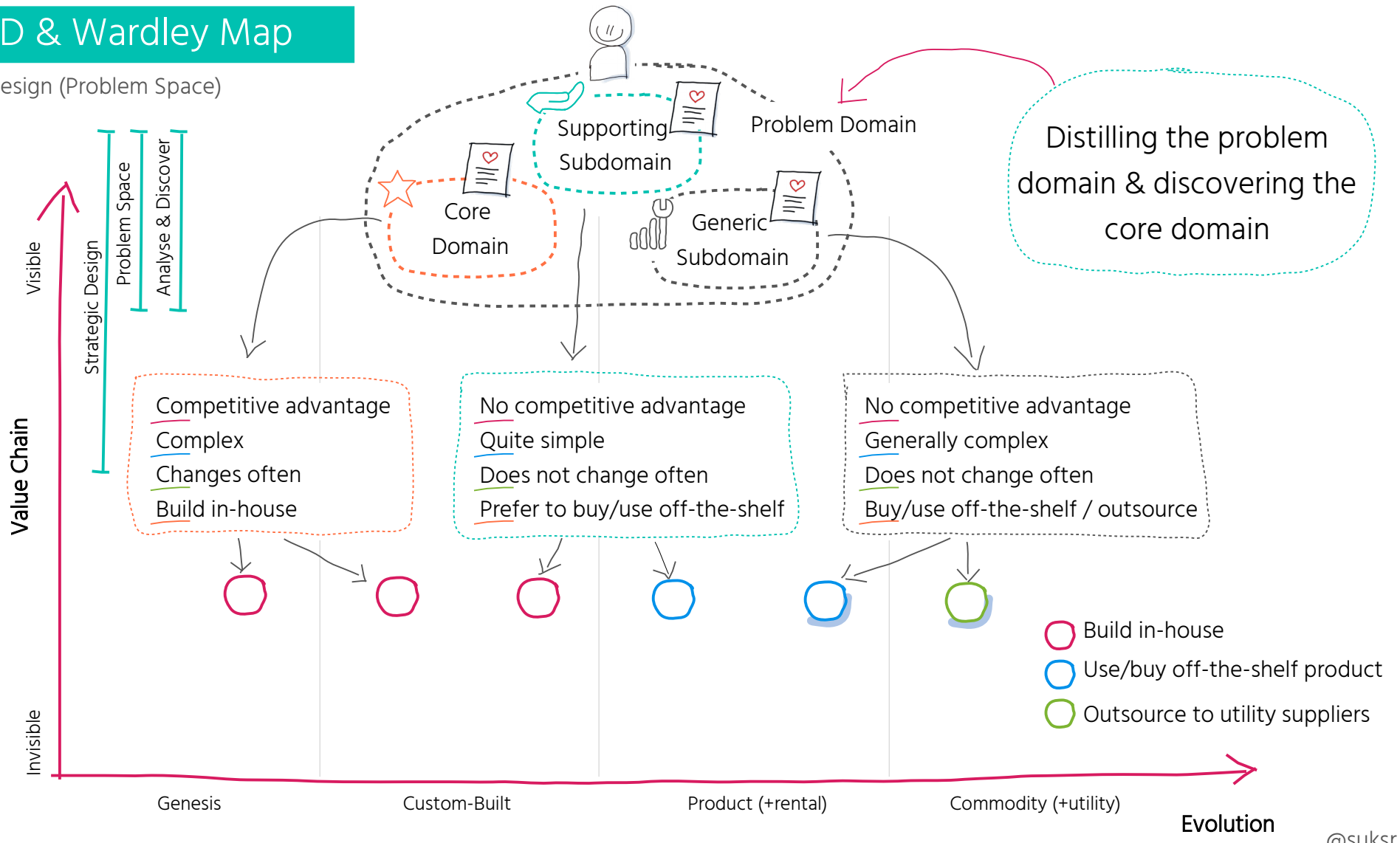
Strategic Design (Problem Space)



@suksr

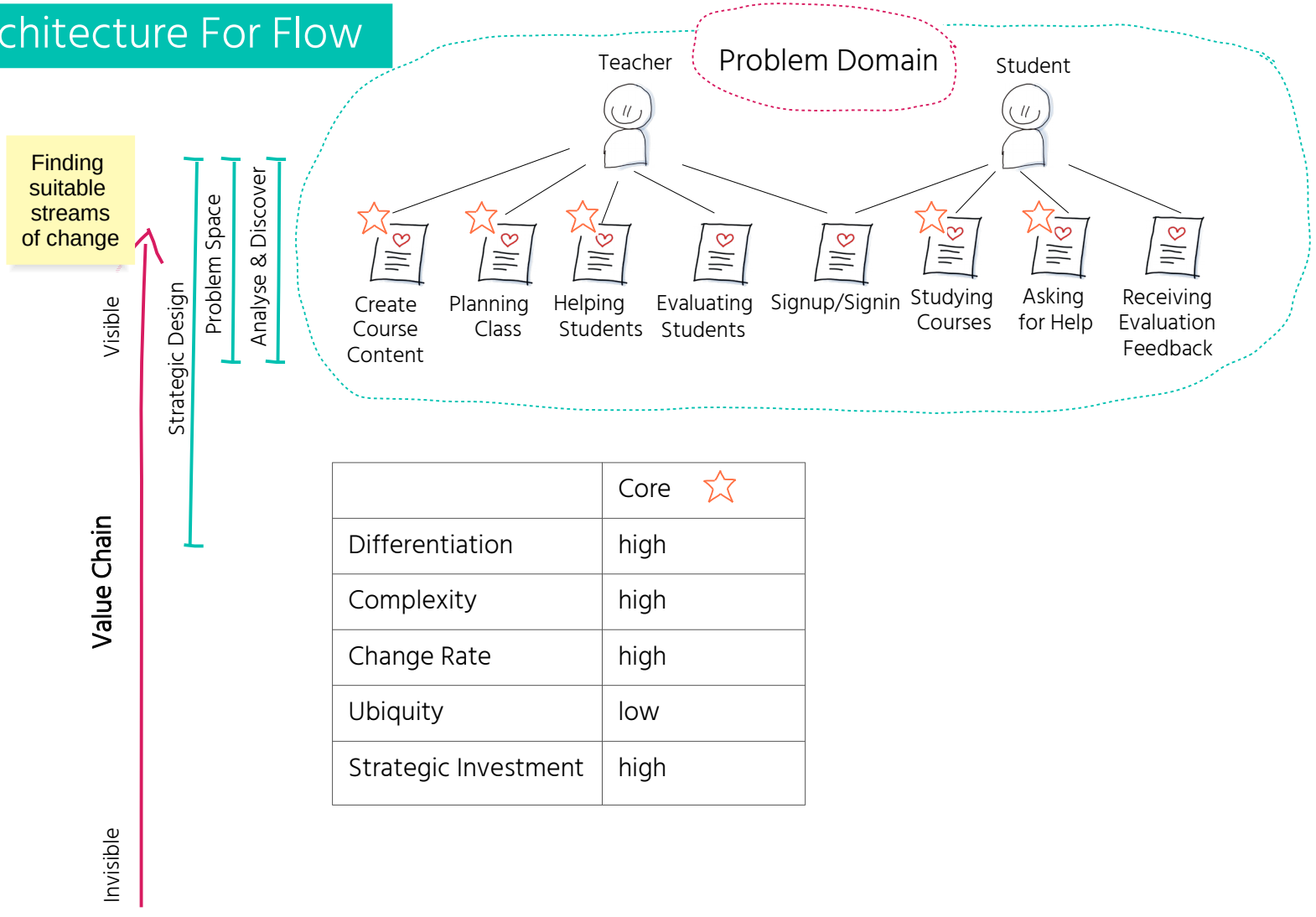
DDD & Wardley Map

Strategic Design (Problem Space)



@suksr

Architecture For Flow

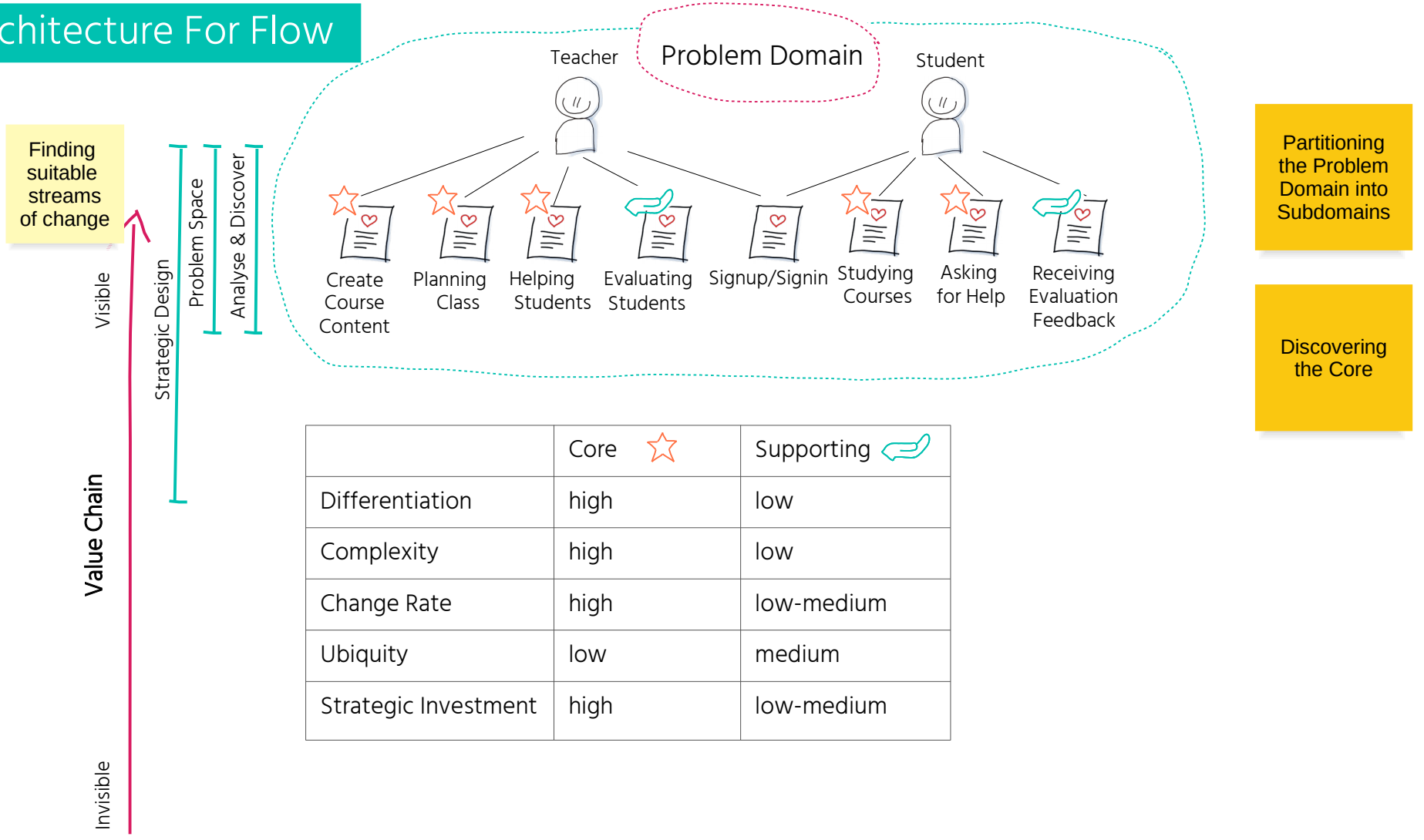


	Core ☆
Differentiation	high
Complexity	high
Change Rate	high
Ubiquity	low
Strategic Investment	high

Partitioning the Problem Domain into Subdomains

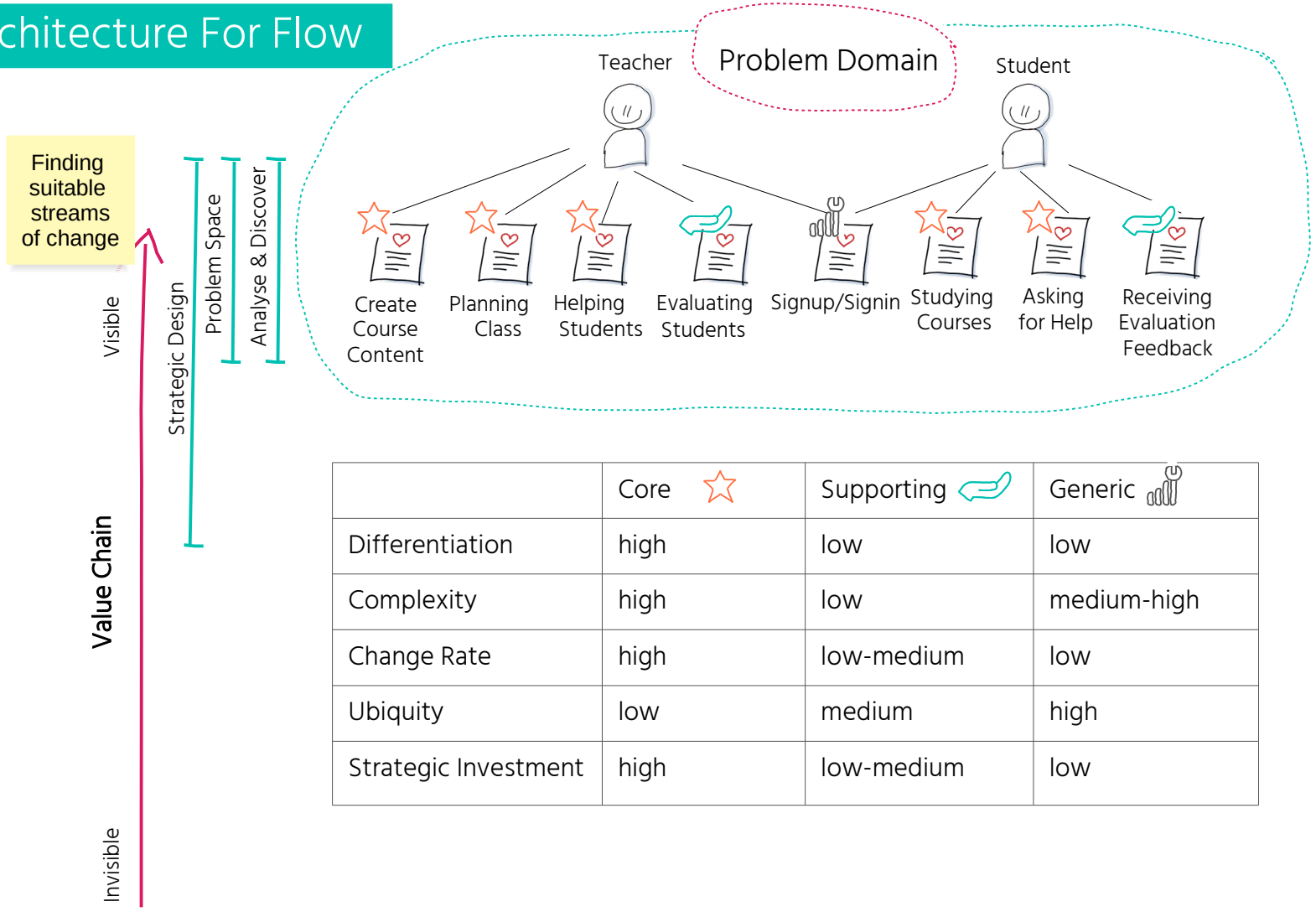
Discovering the Core

Architecture For Flow



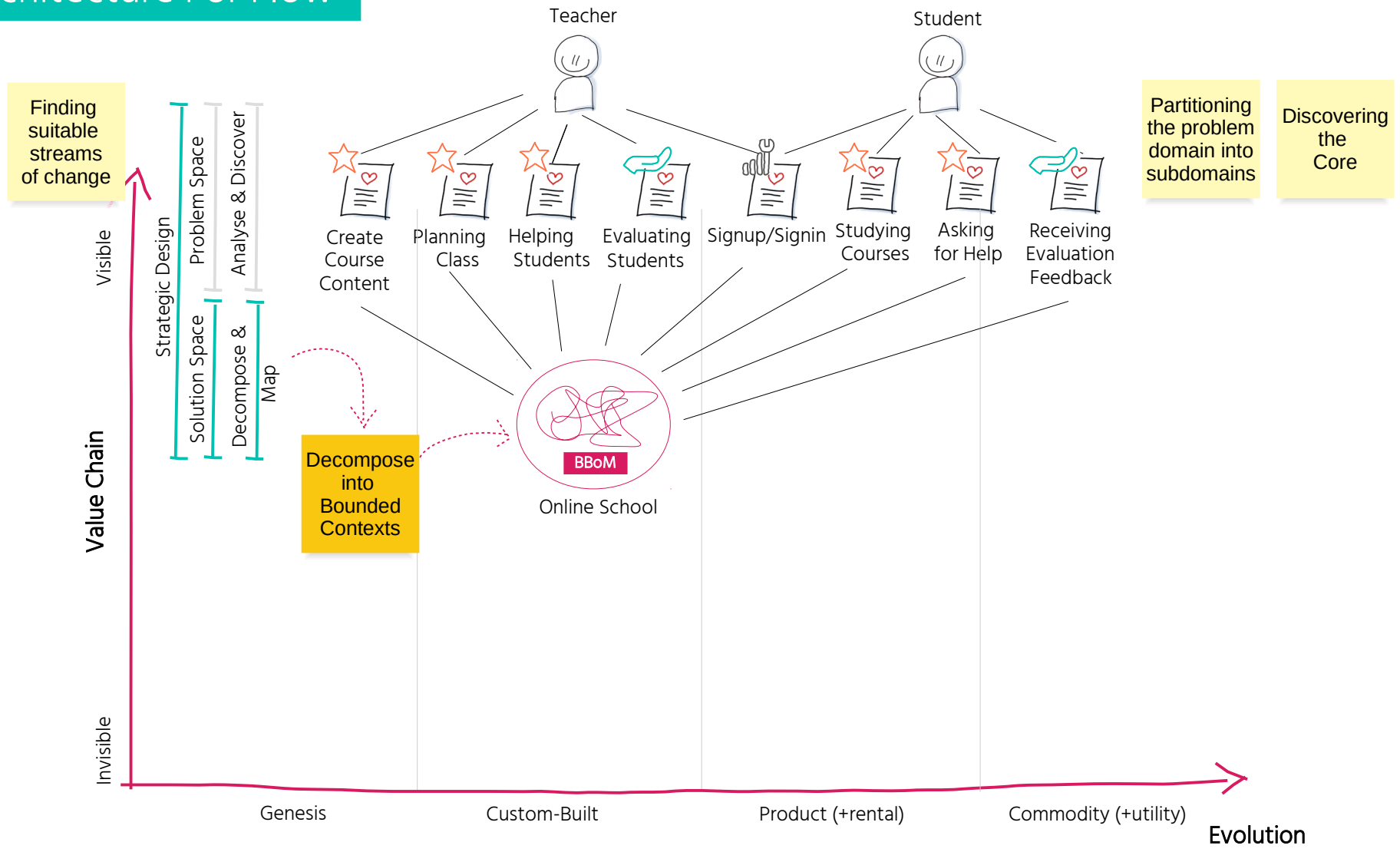
	Core ☆	Supporting
Differentiation	high	low
Complexity	high	low
Change Rate	high	low-medium
Ubiquity	low	medium
Strategic Investment	high	low-medium

Architecture For Flow

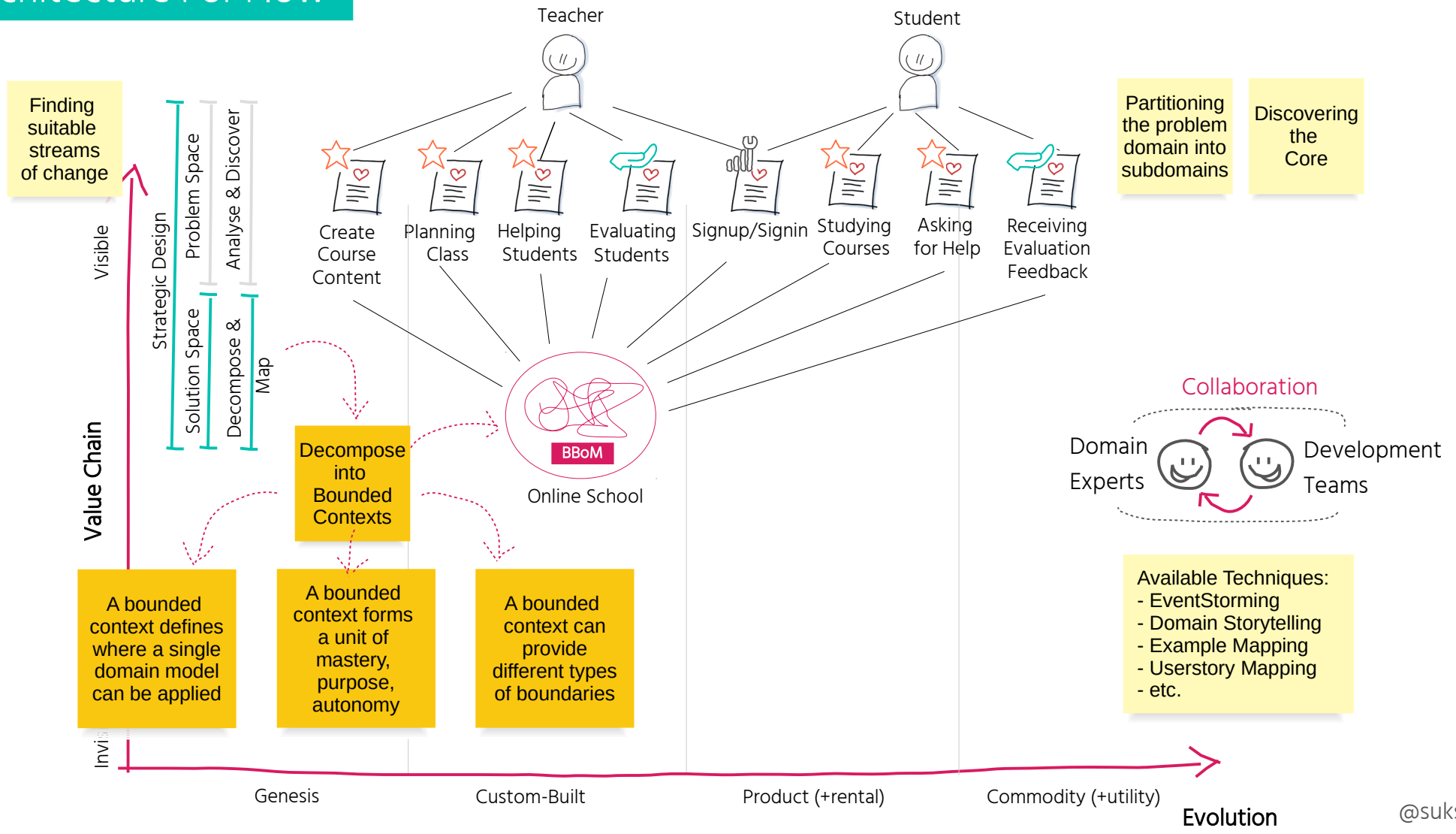


	Core	Supporting	Generic
Differentiation	high	low	low
Complexity	high	low	medium-high
Change Rate	high	low-medium	low
Ubiquity	low	medium	high
Strategic Investment	high	low-medium	low

Architecture For Flow

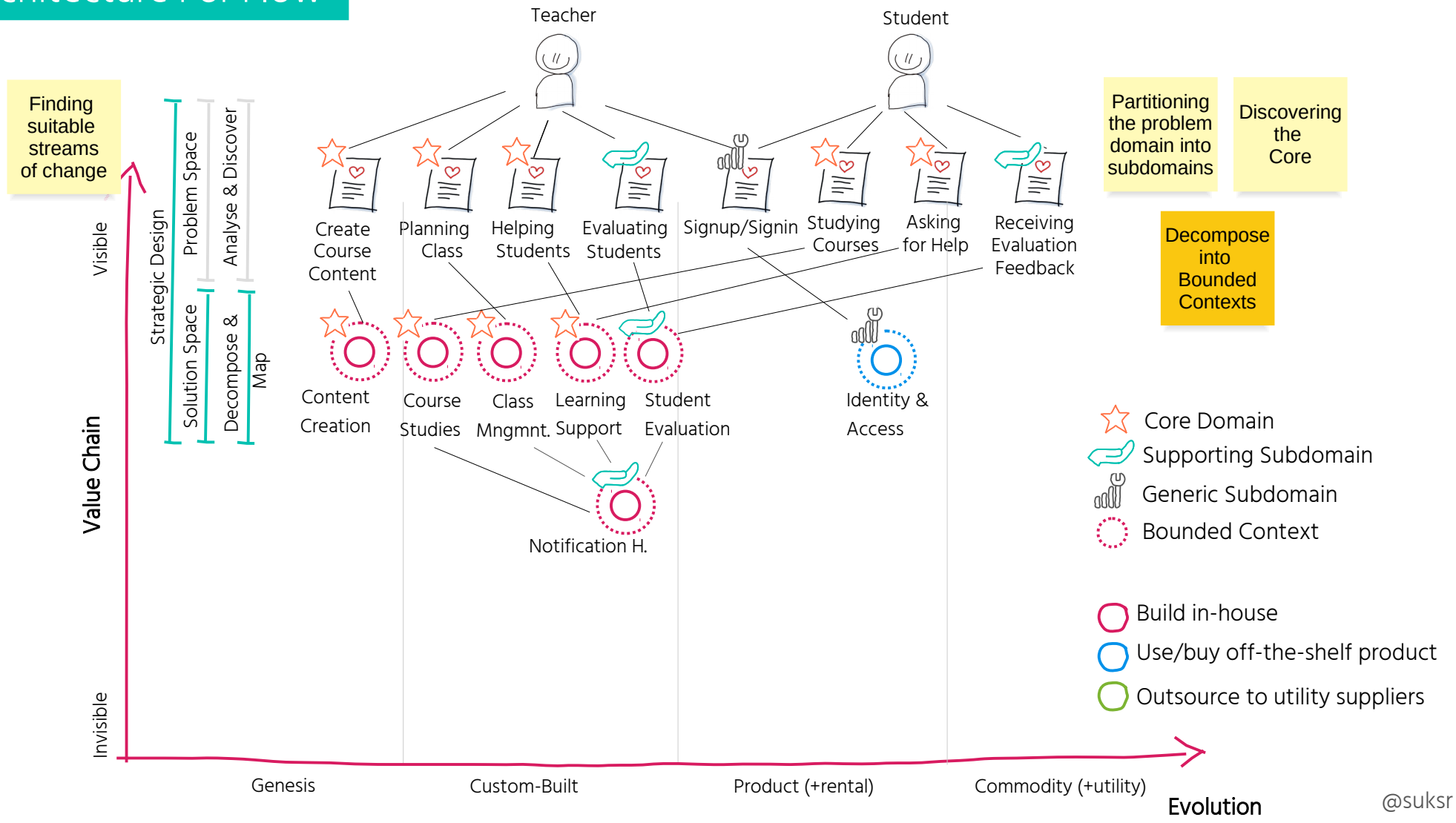


Architecture For Flow



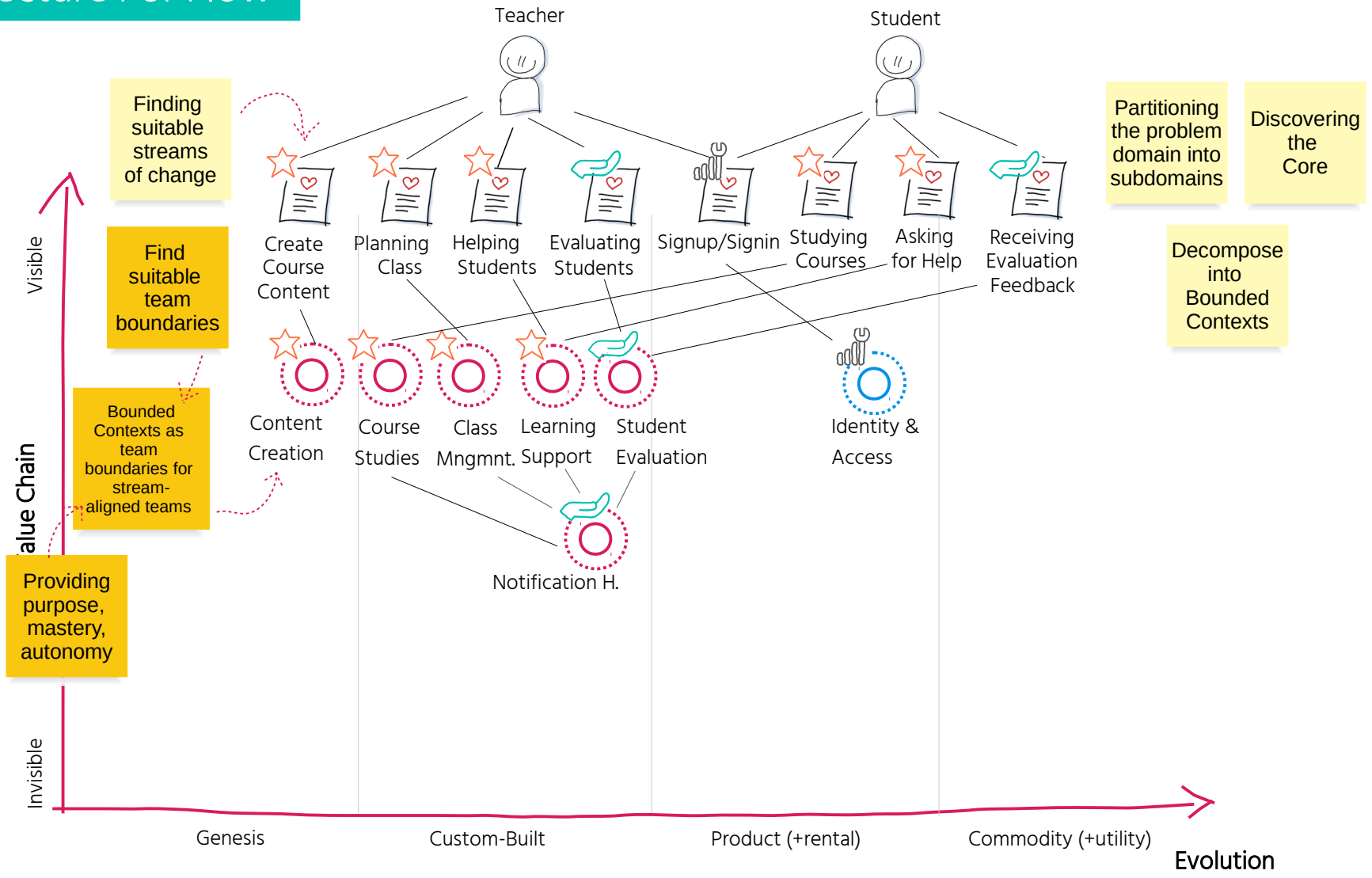
@suksr

Architecture For Flow



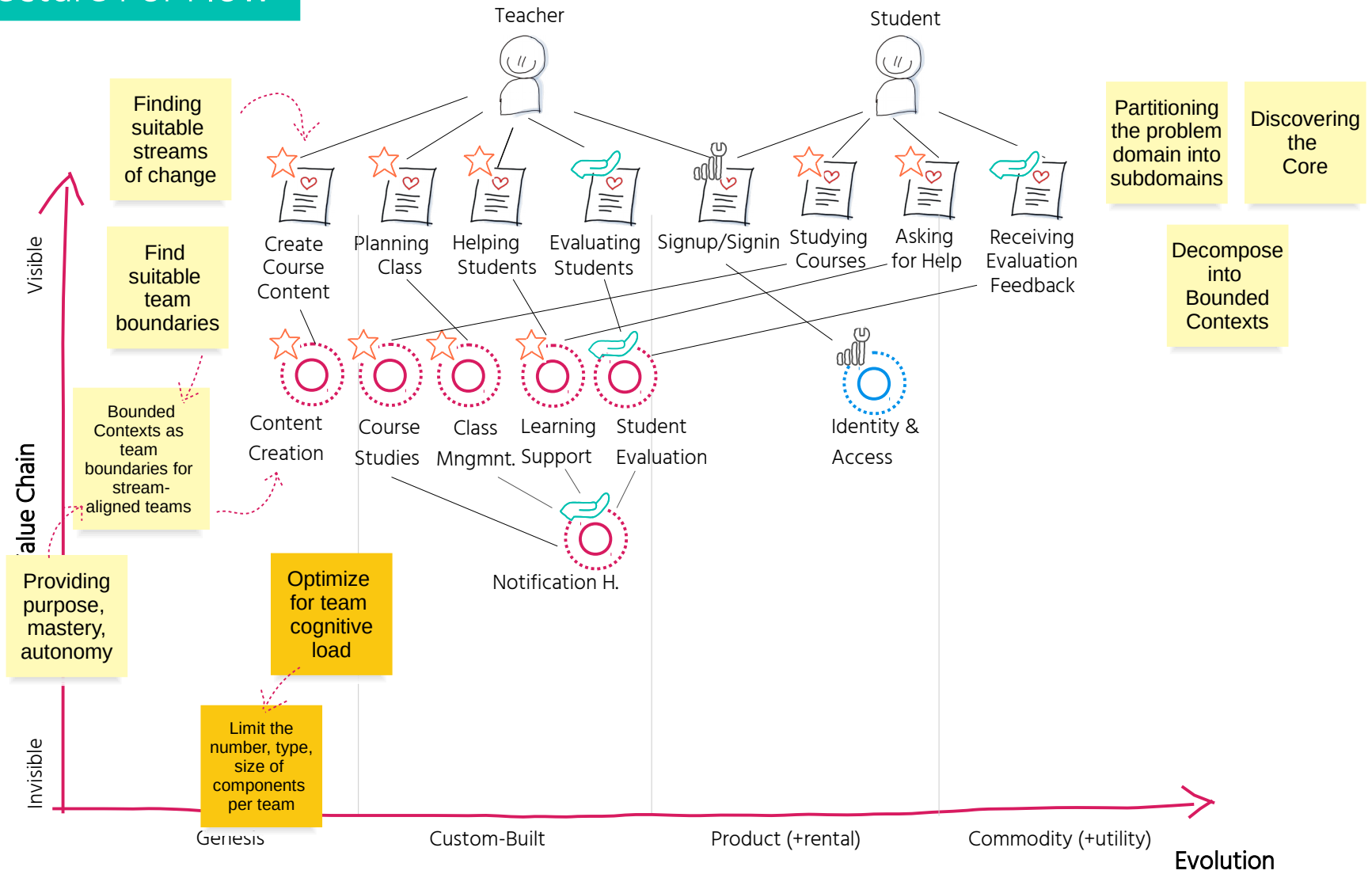
@suksr

Architecture For Flow



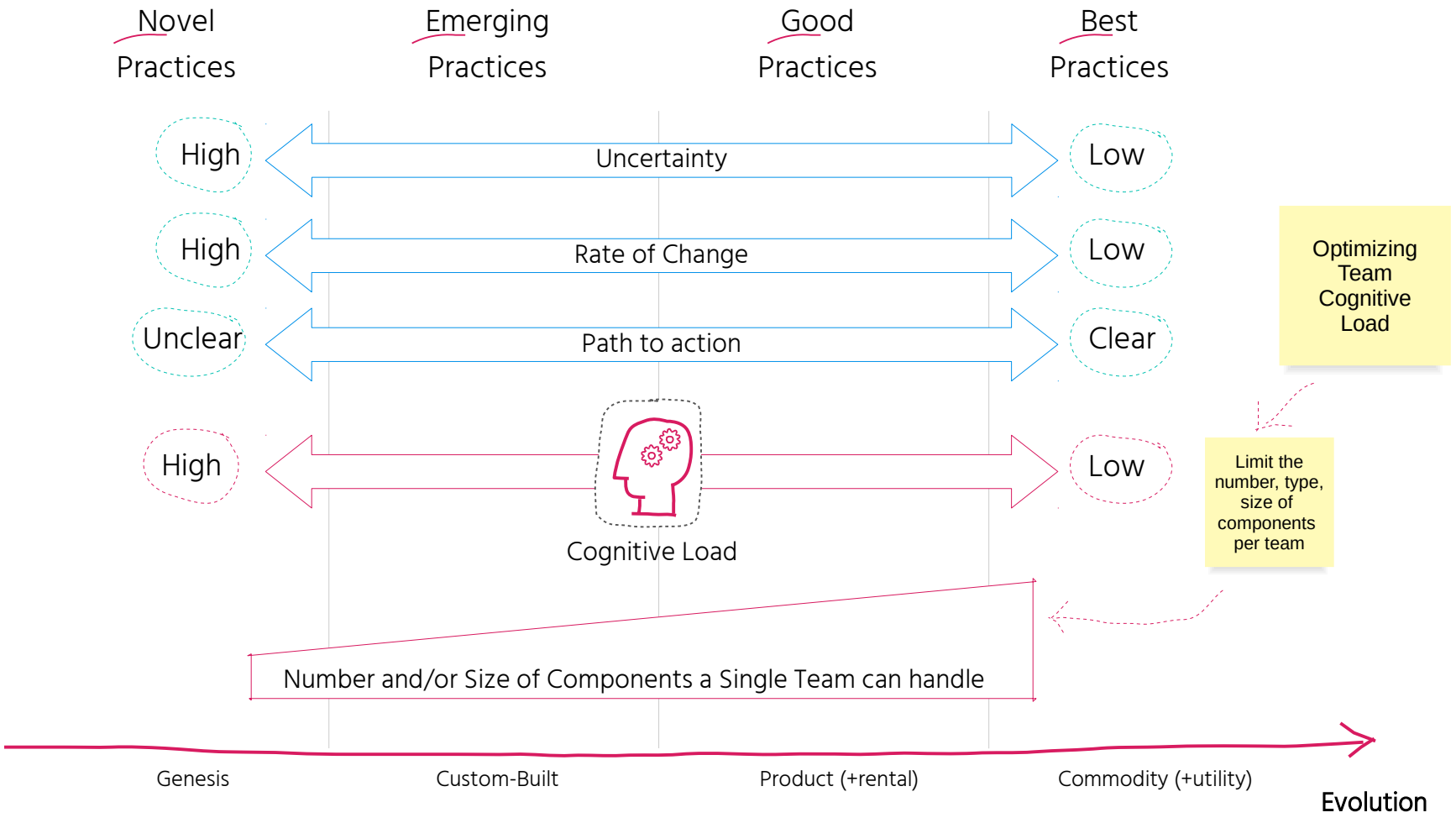
@suksr

Architecture For Flow

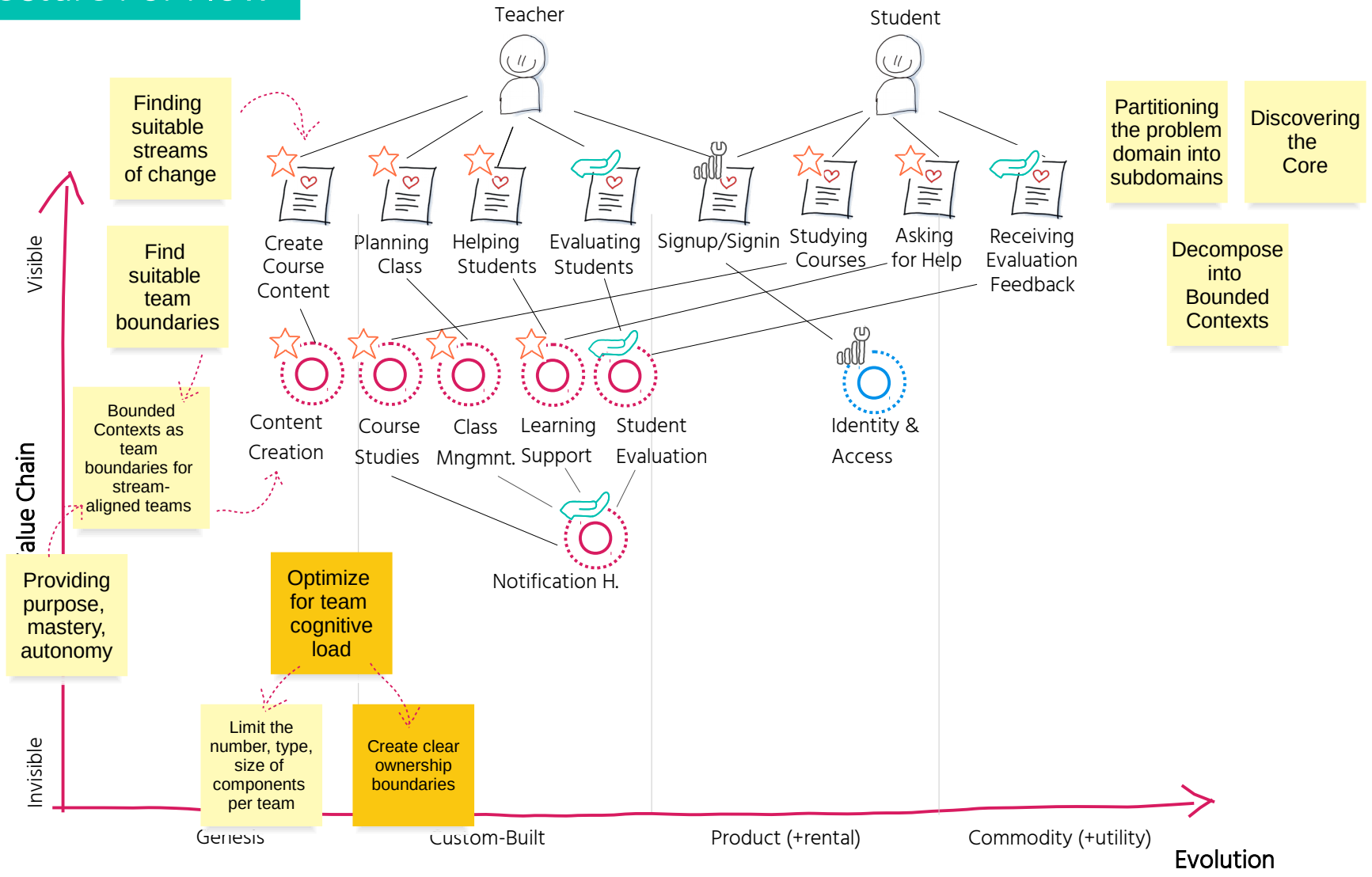


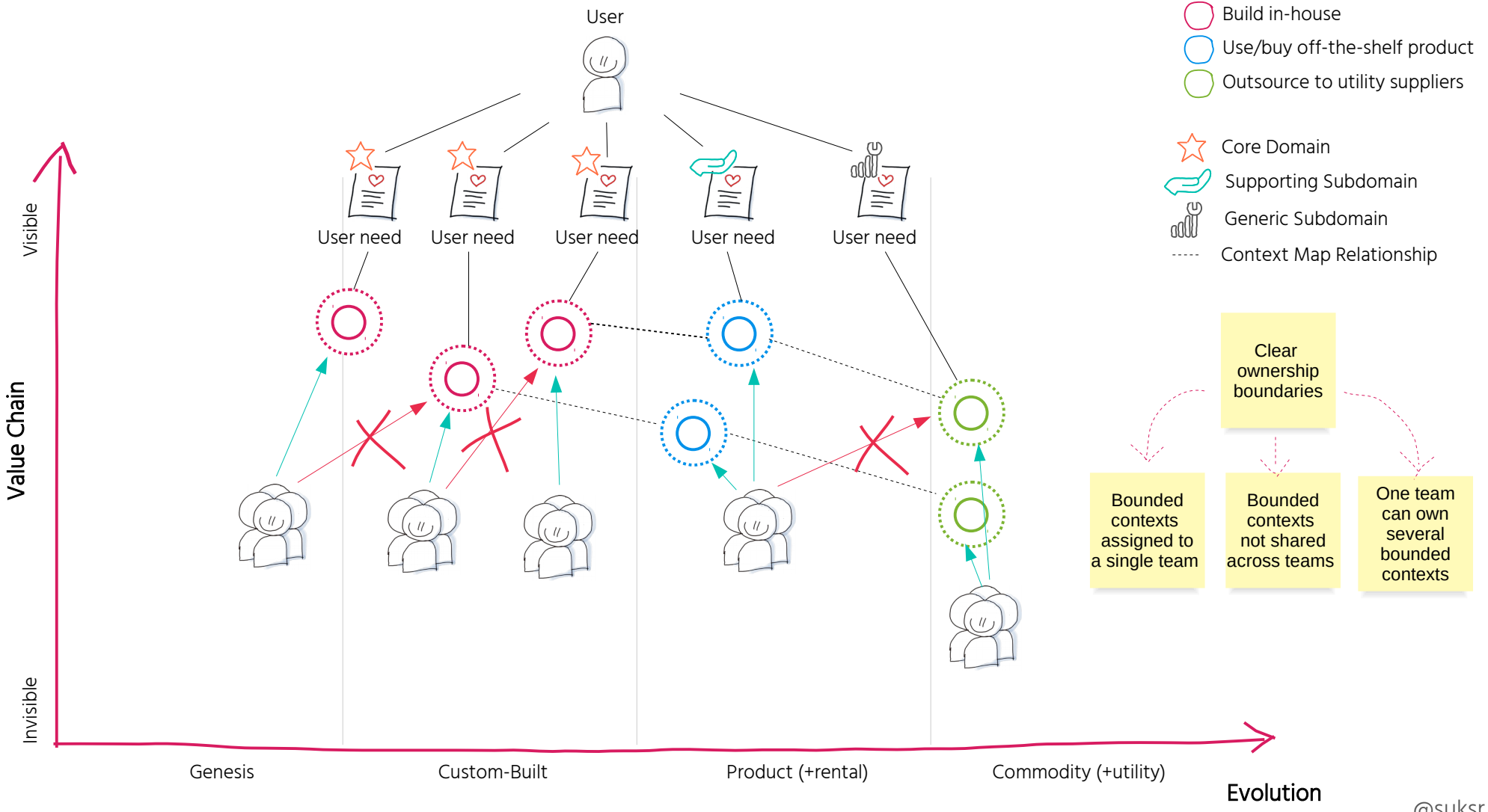
@suksr

Architecture For Flow



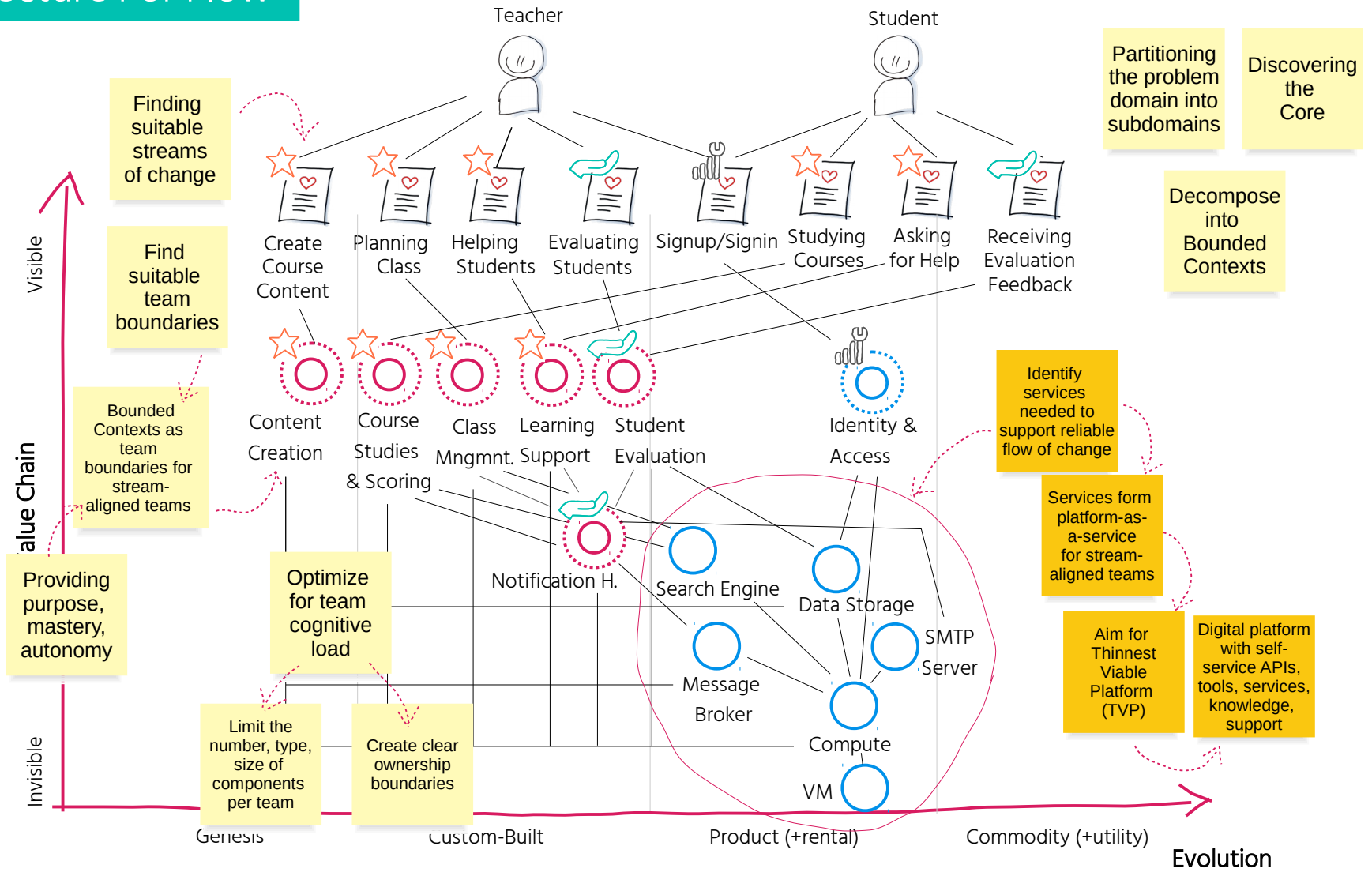
Architecture For Flow




User
User need
User need
User need
User need
User need
Build in-house
Use/buy off-the-shelf product
Outsource to utility suppliers
Core Domain
Supporting Subdomain
Generic Subdomain
Context Map Relationship
Clear ownership boundaries
Bounded contexts assigned to a single team
Bounded contexts not shared across teams
One team can own several bounded contexts
Visible
Value Chain
Invisible
Genesis
Custom-Built
Product (+rental)
Commodity (+utility)
Evolution

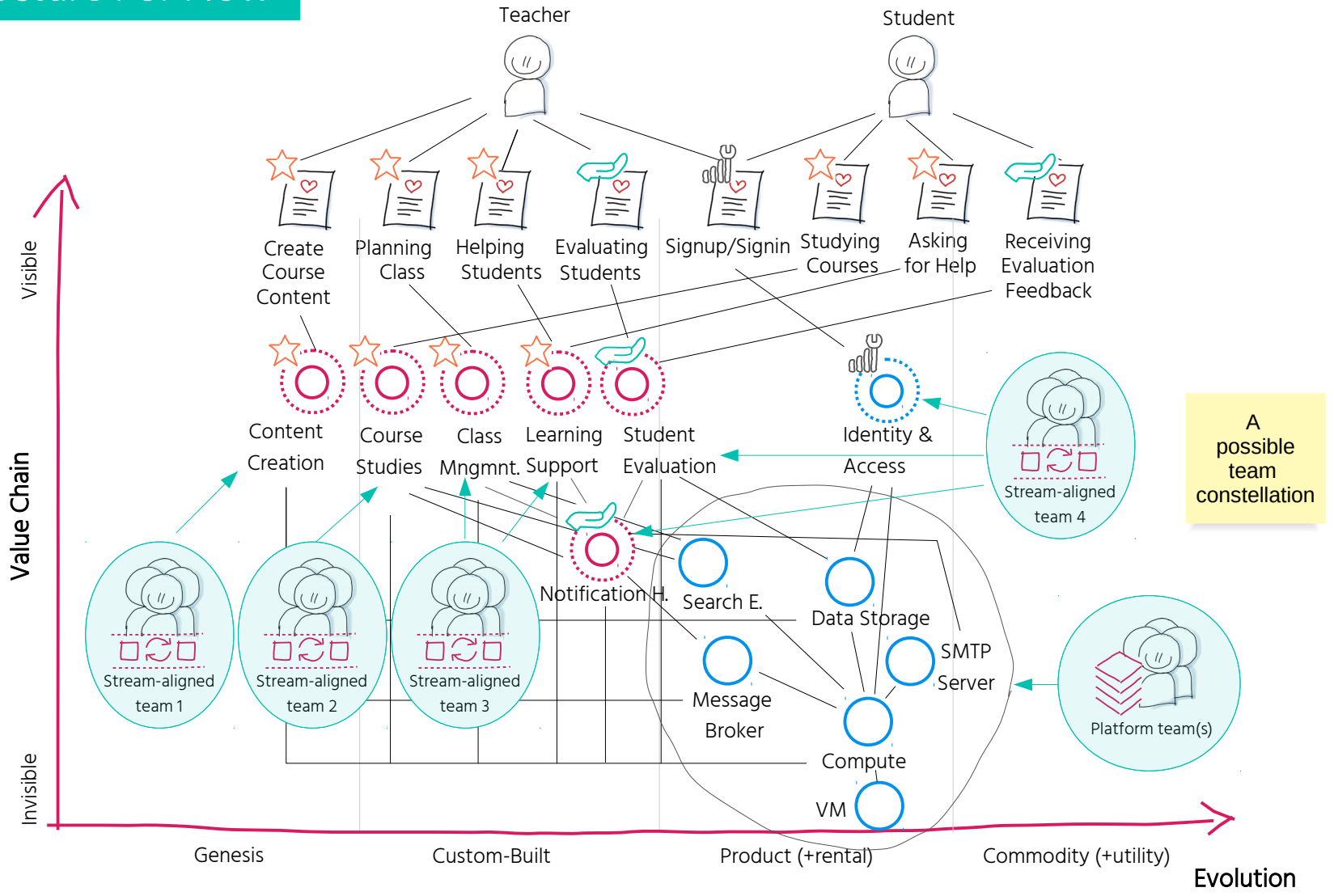
@suksr

Architecture For Flow

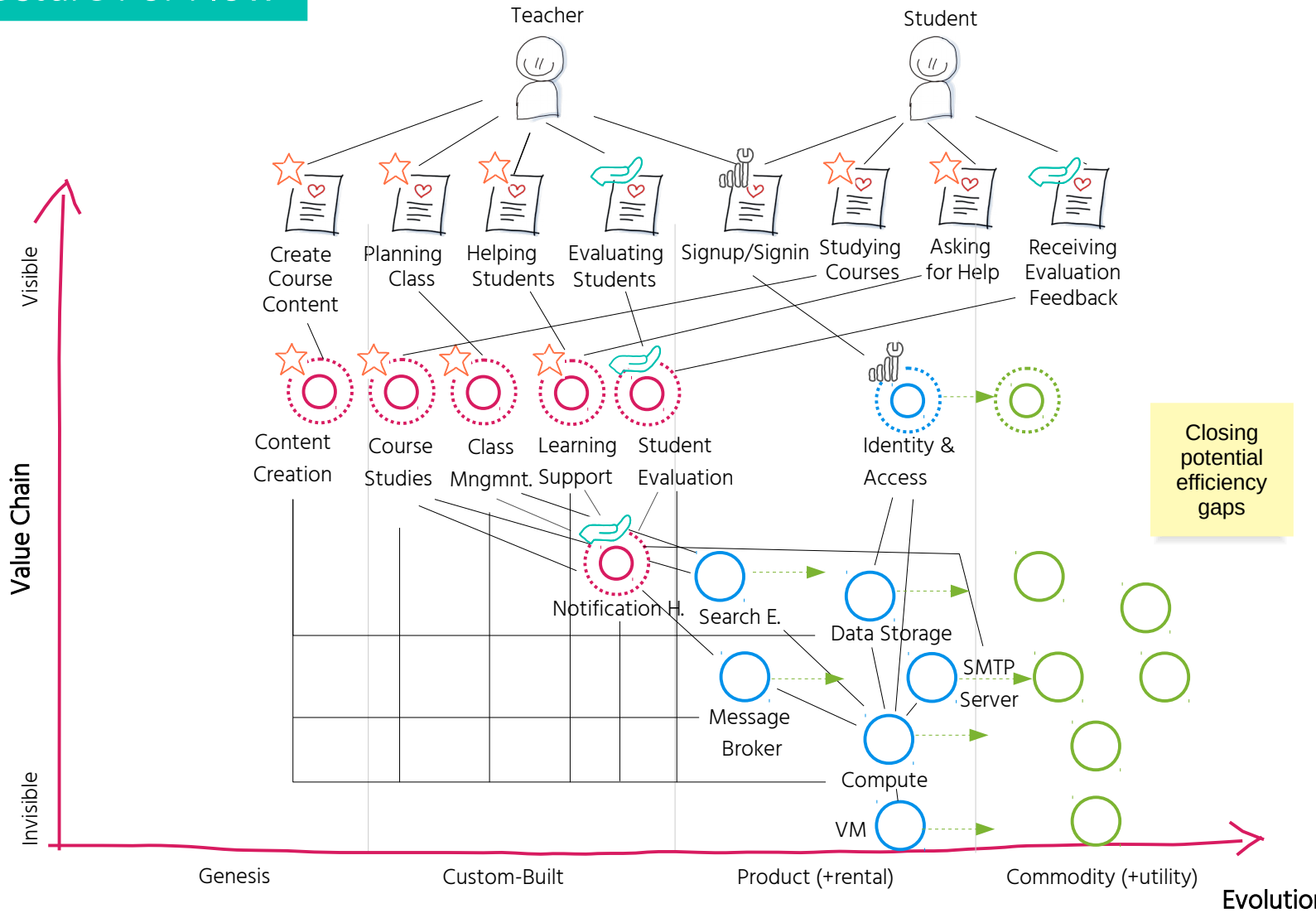


@suksr

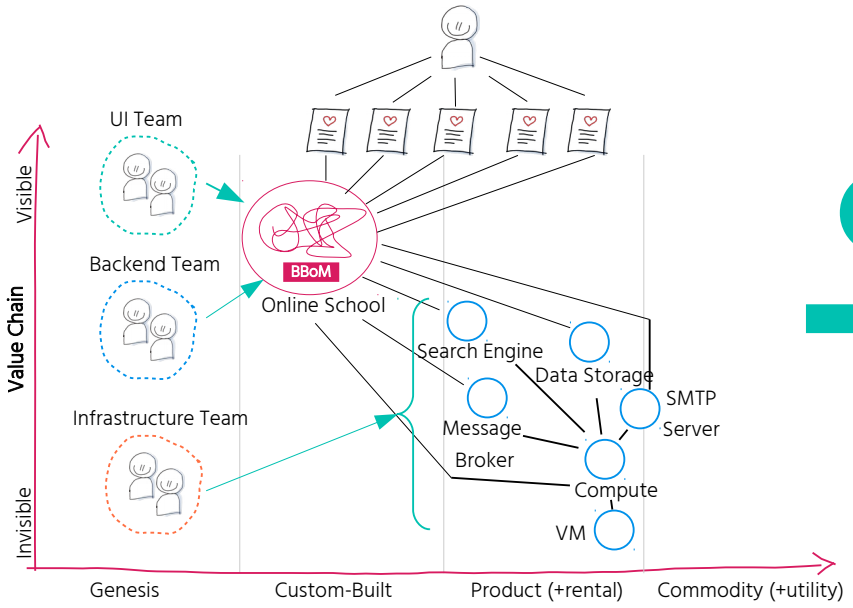
Architecture For Flow

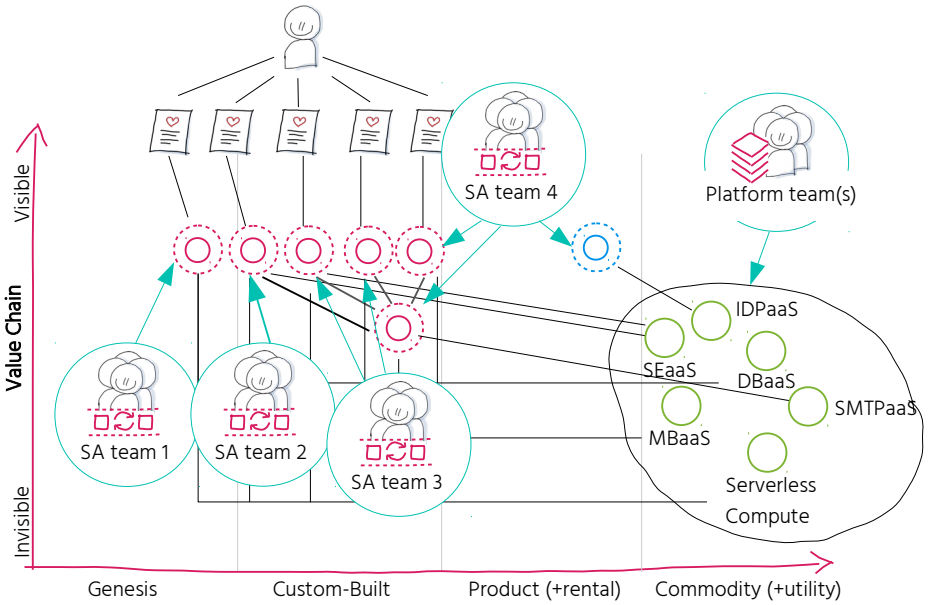


Architecture For Flow

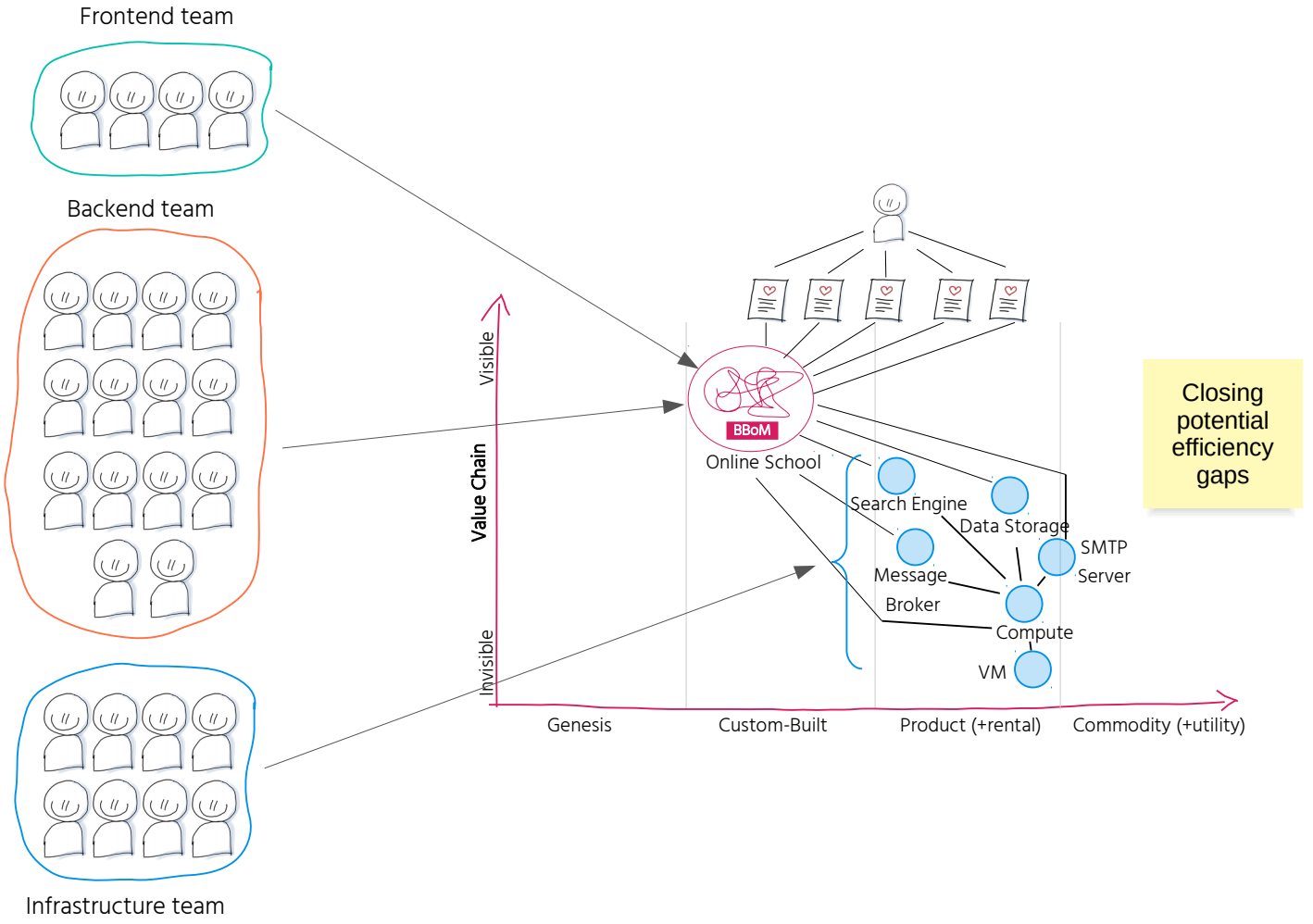


How to transition?

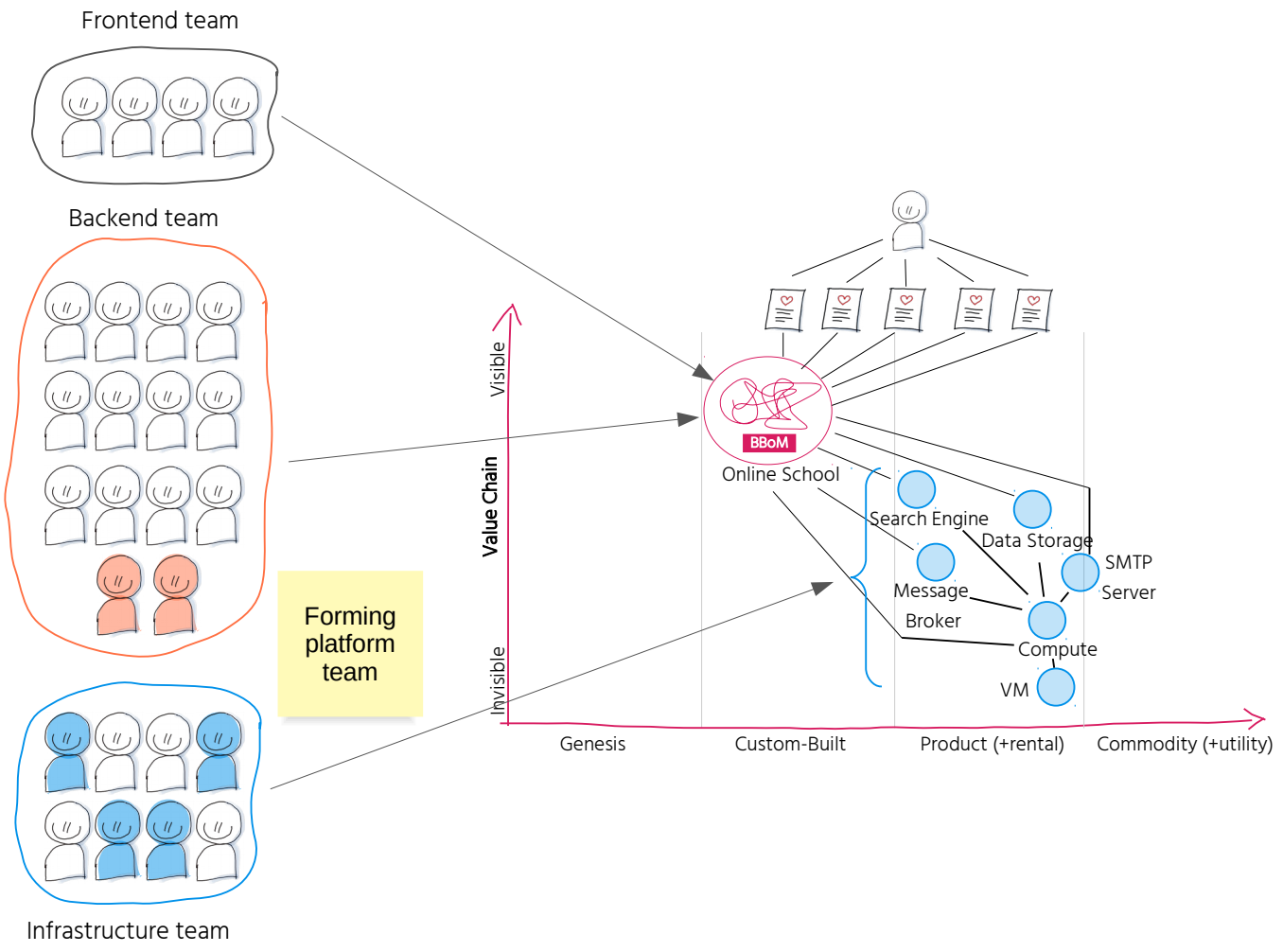




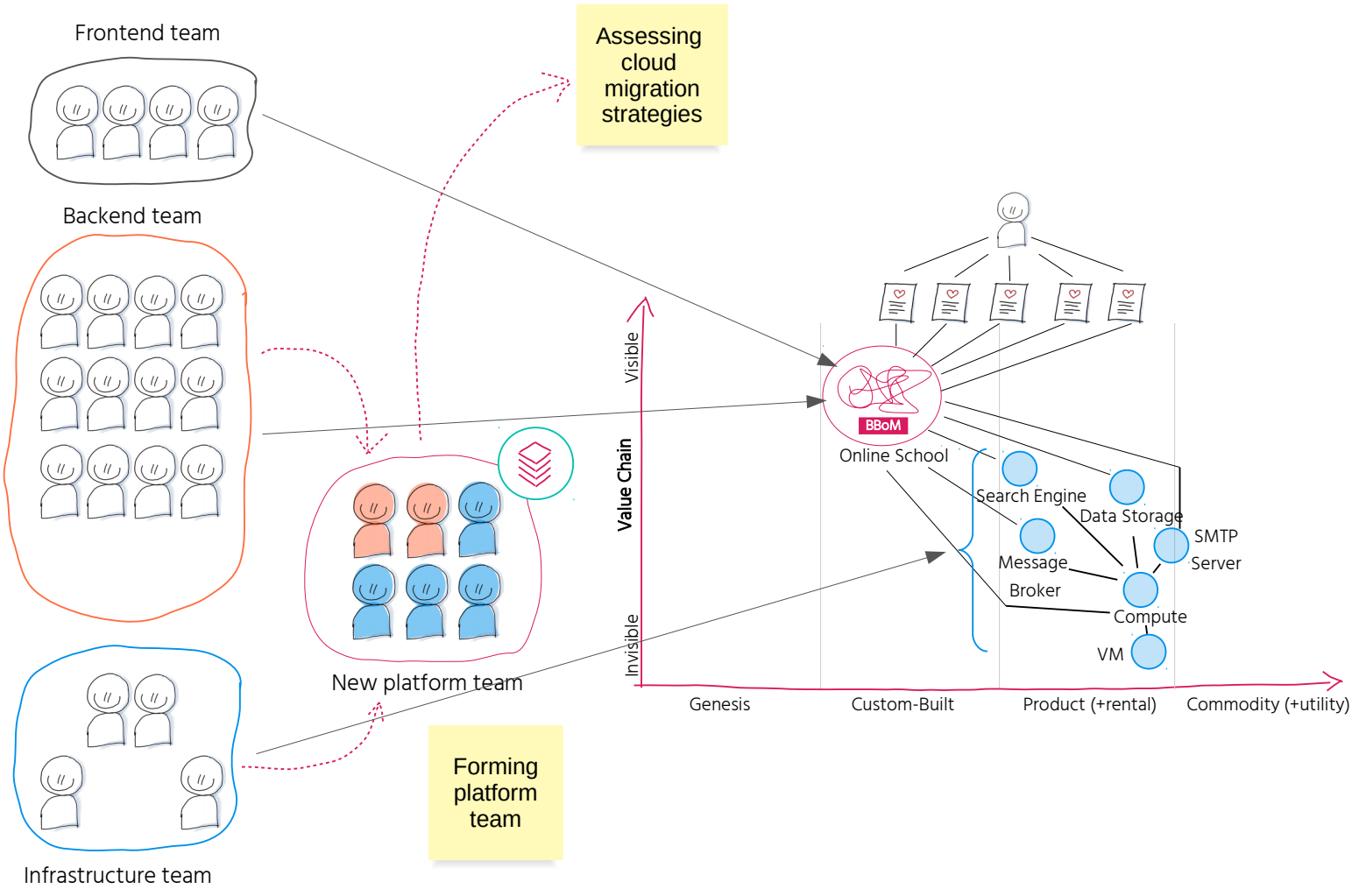
Implementing Flow Optimization



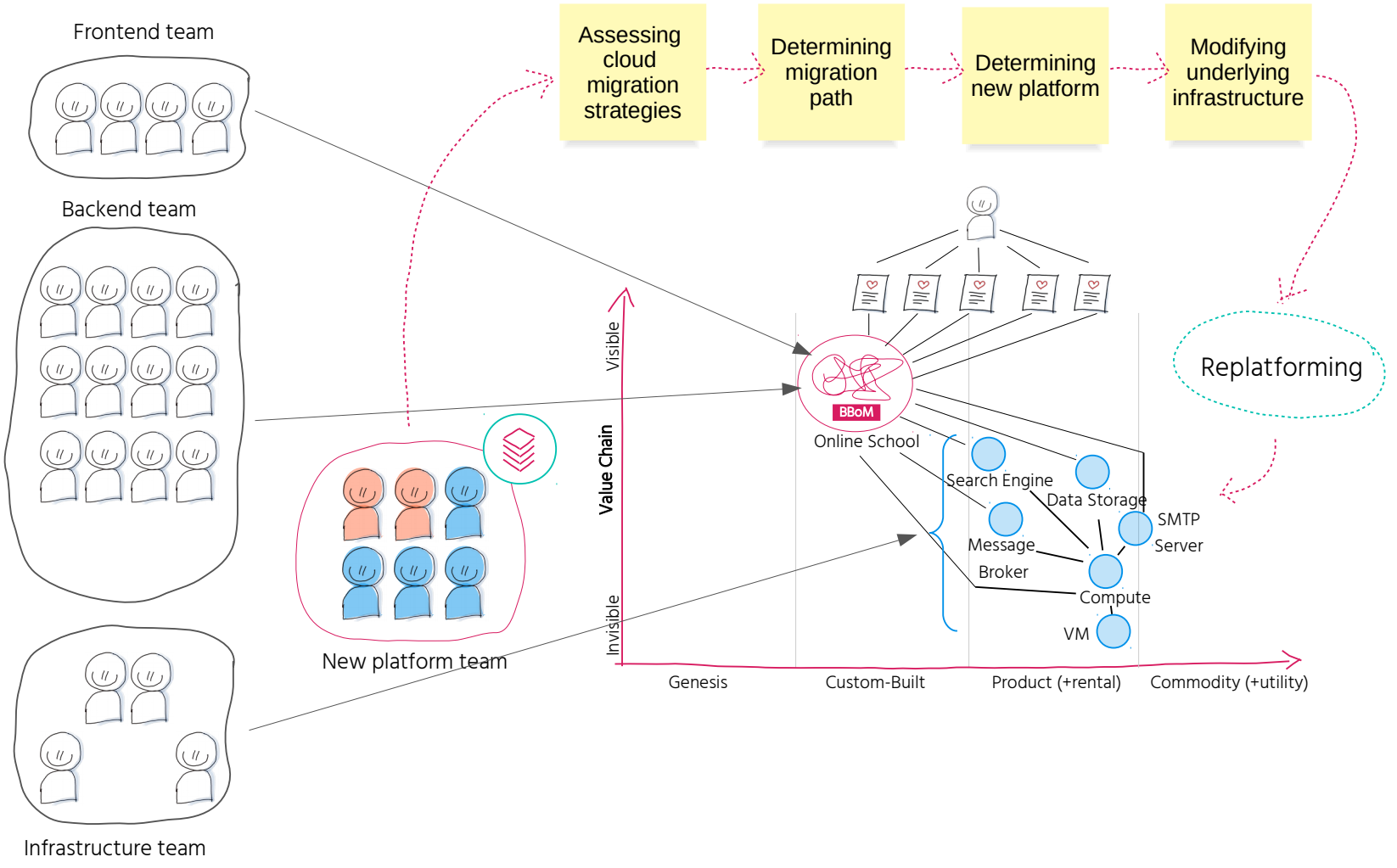
Implementing Flow Optimization



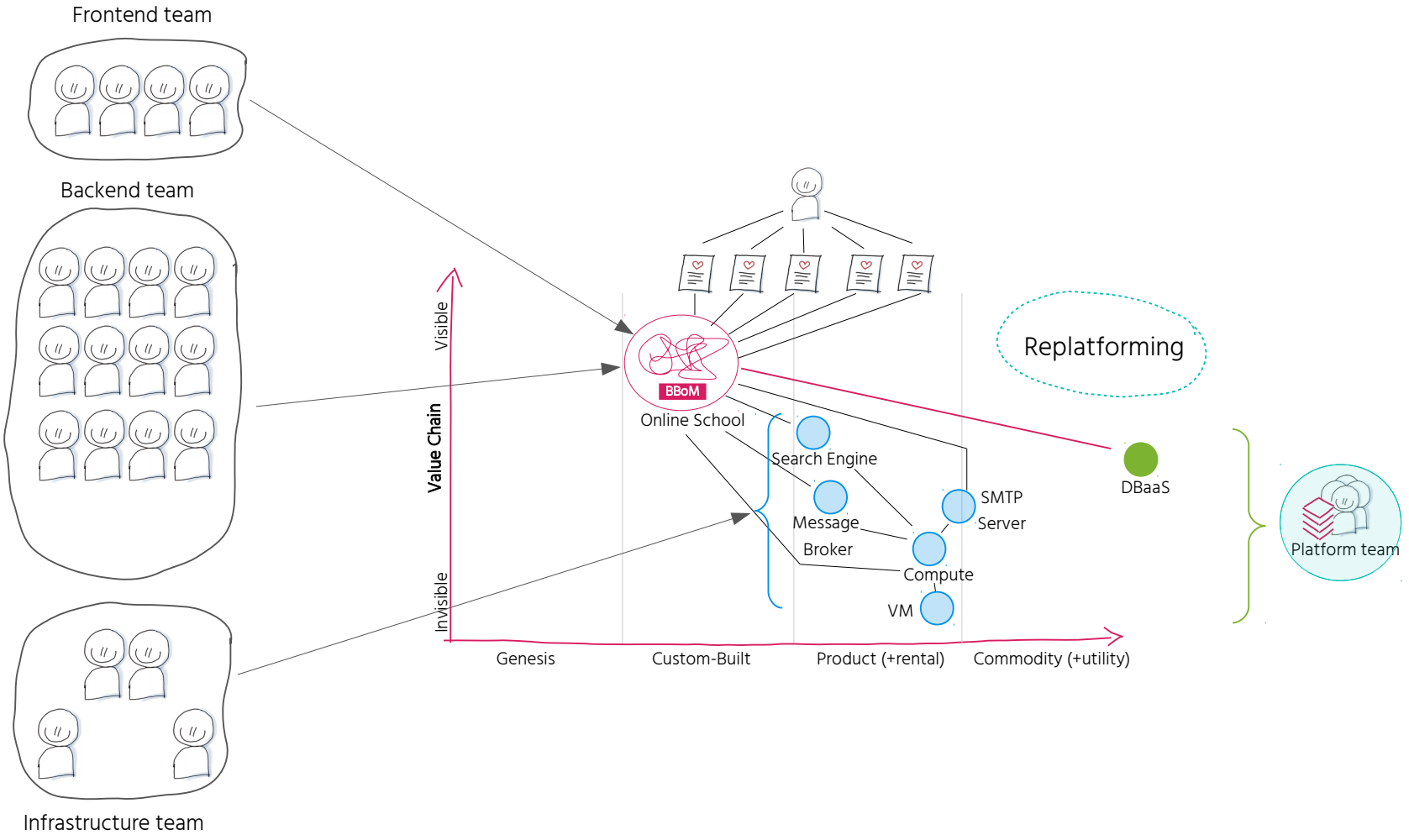
Implementing Flow Optimization



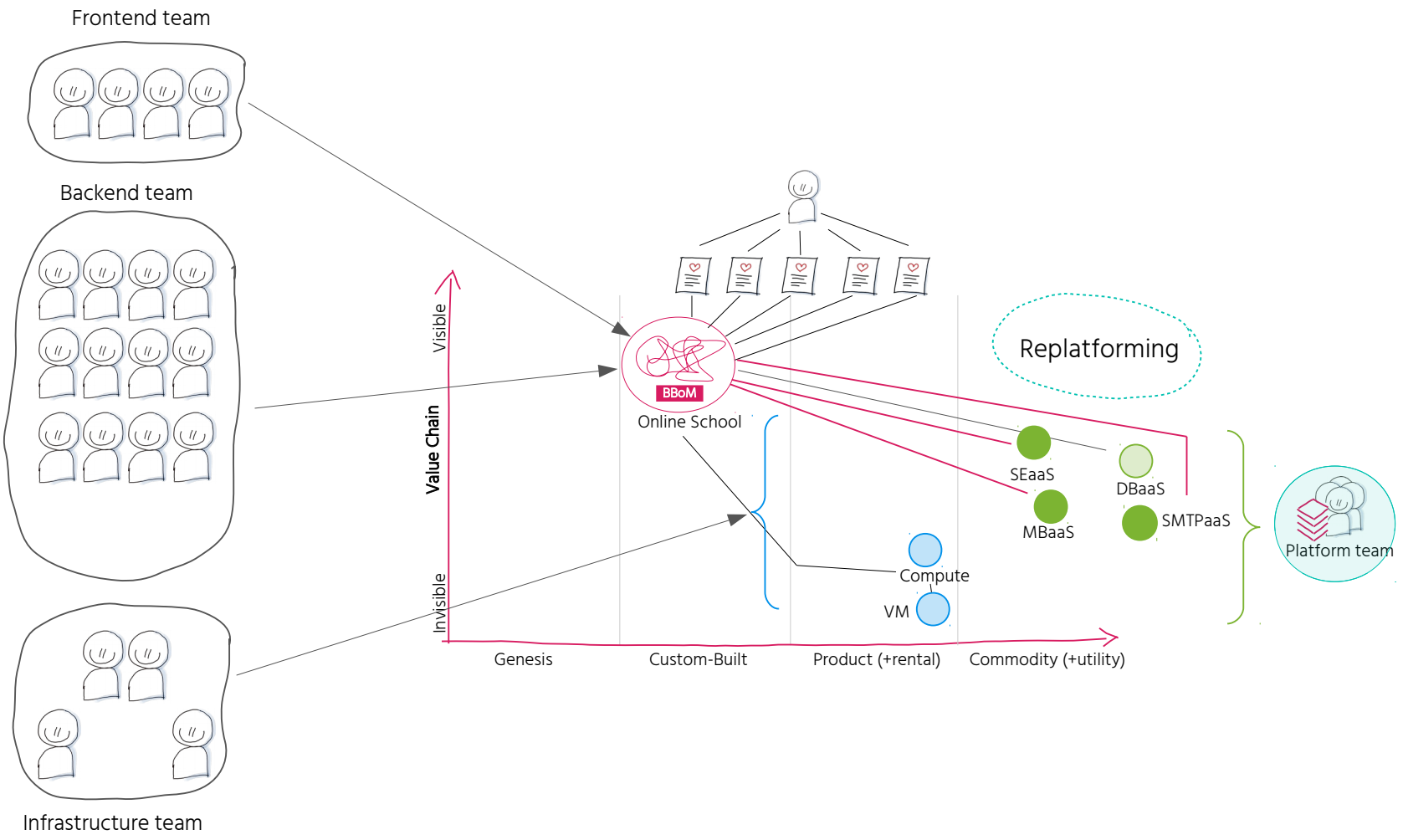
Implementing Flow Optimization



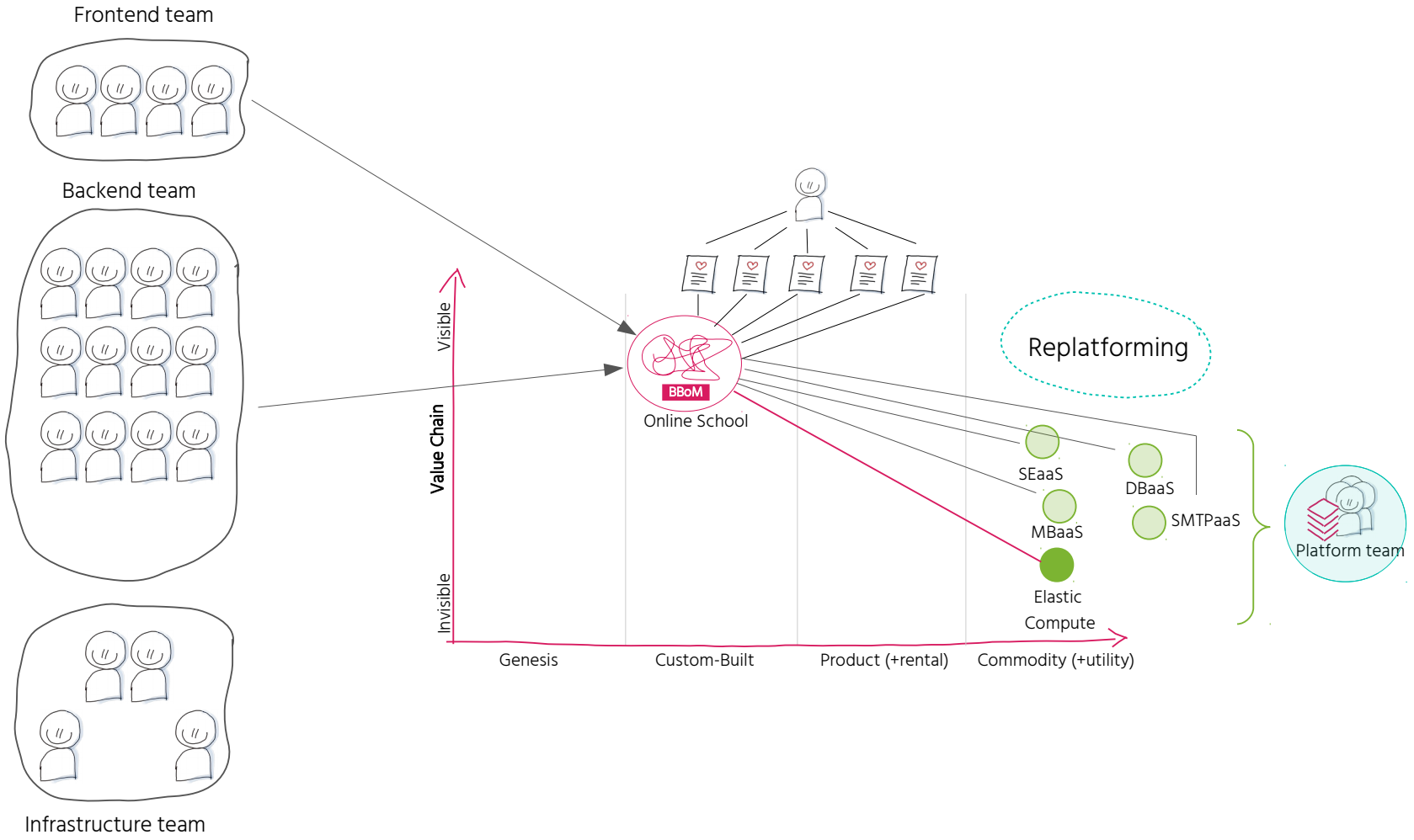
Implementing Flow Optimization



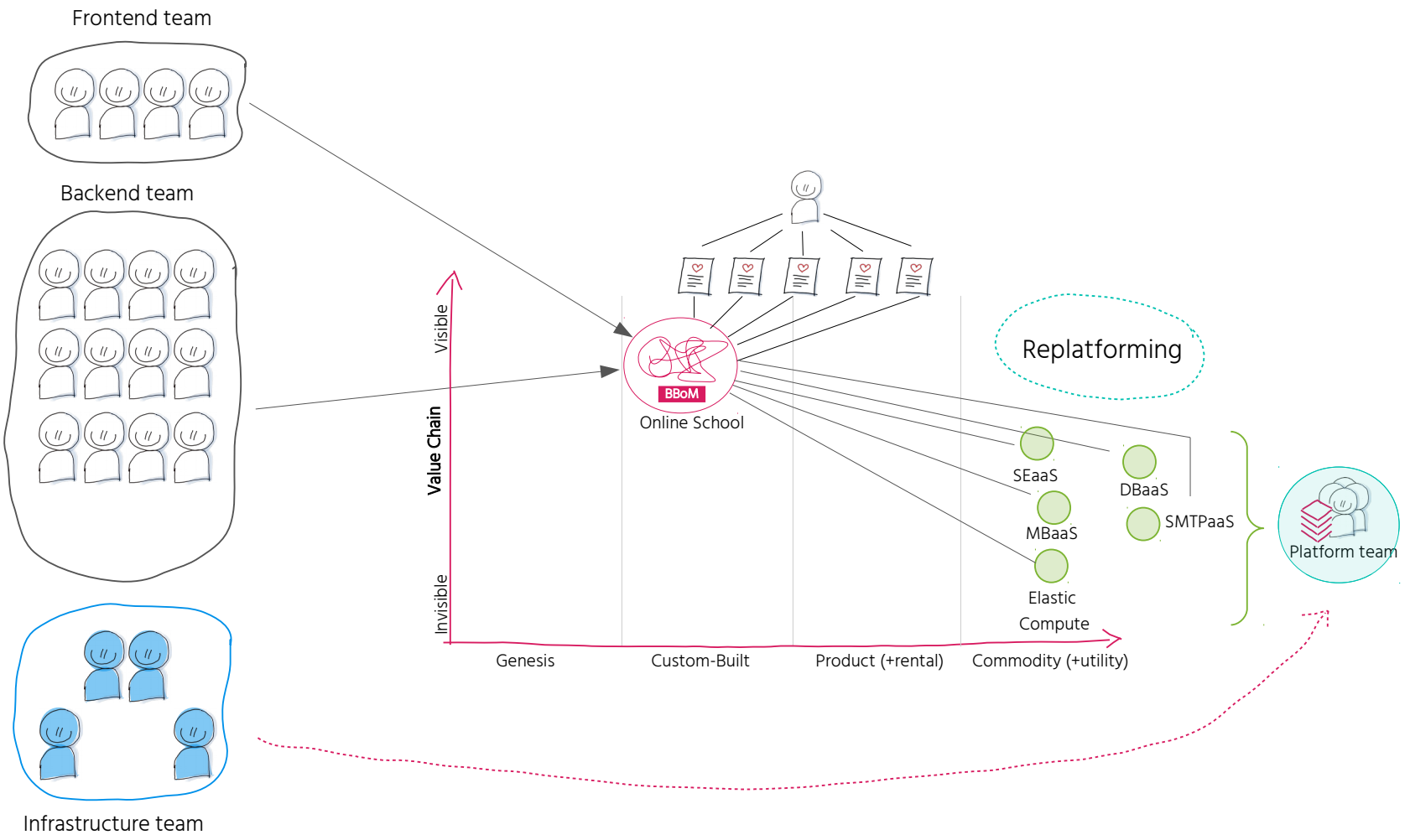
Implementing Flow Optimization



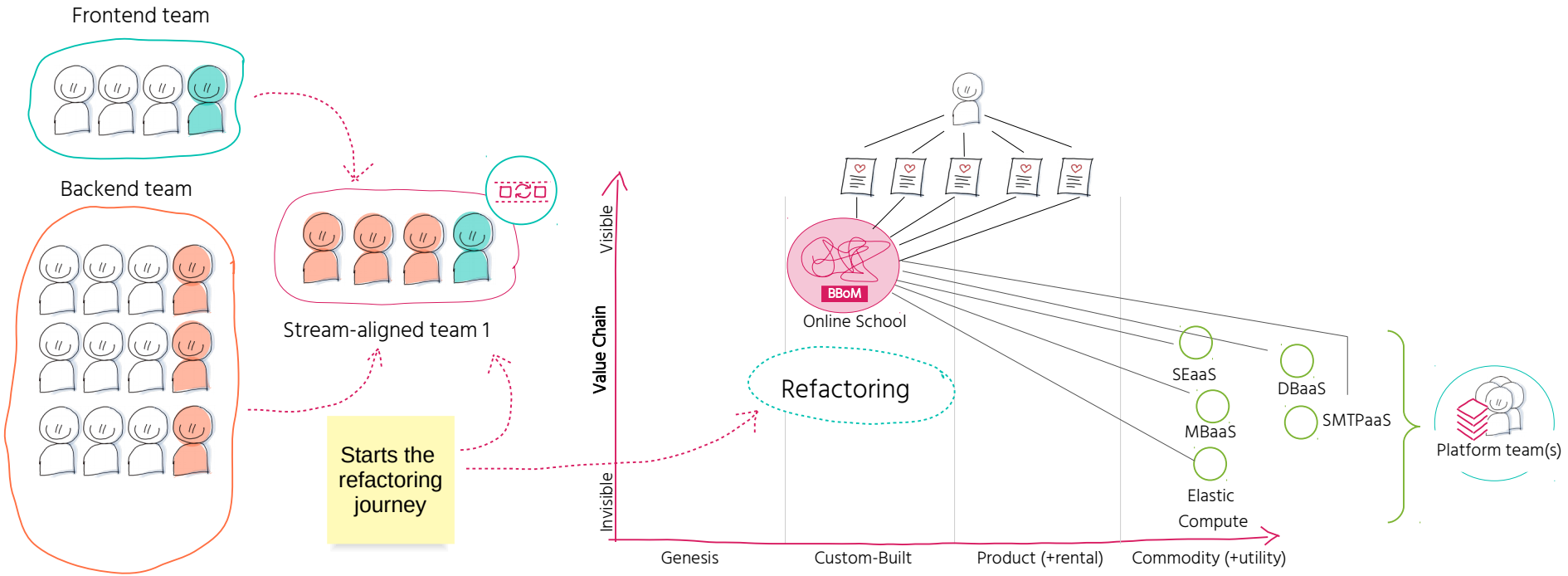
Implementing Flow Optimization



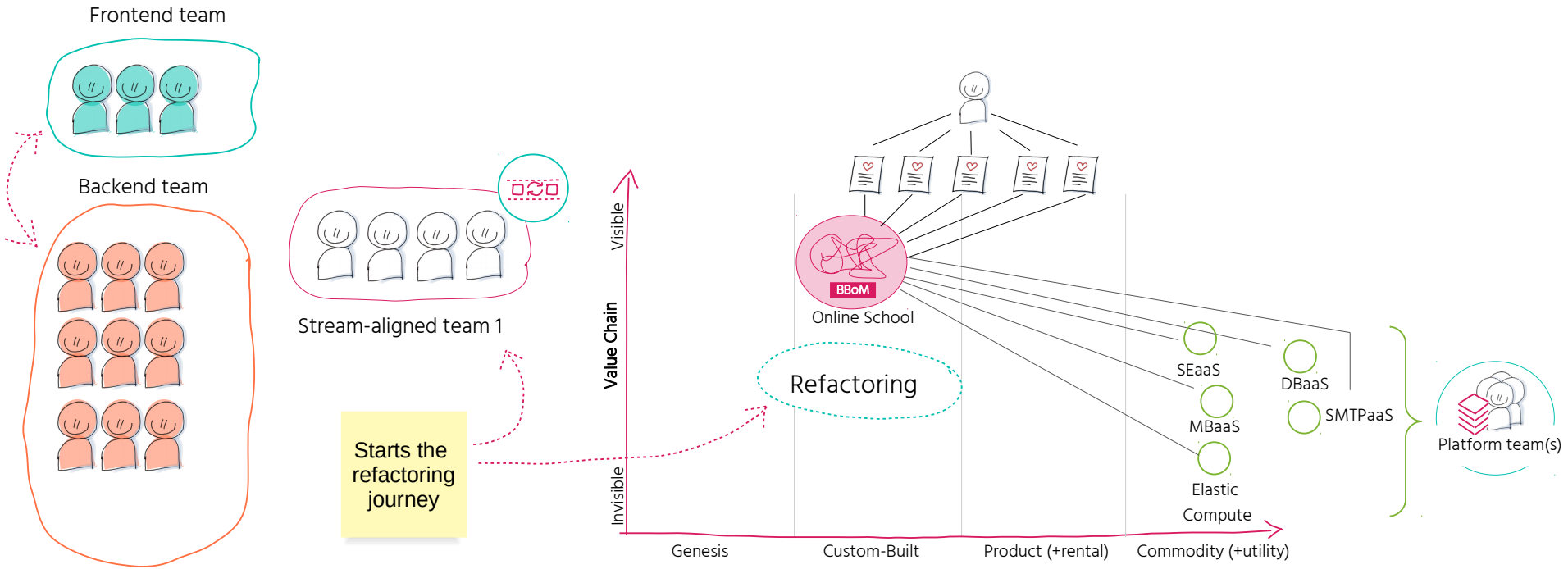
Implementing Flow Optimization



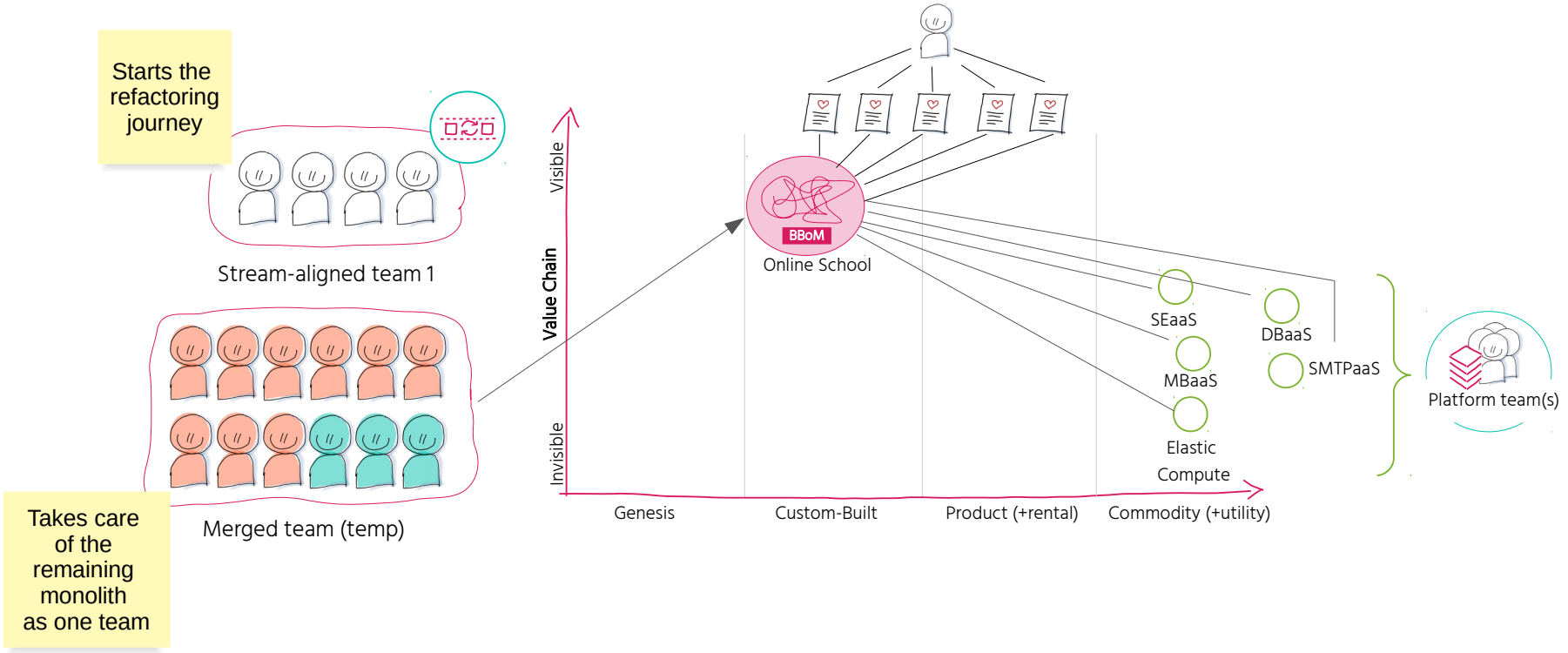
Implementing Flow Optimization



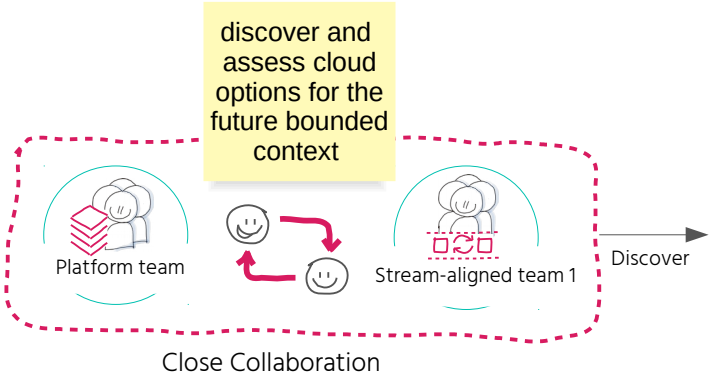
Implementing Flow Optimization

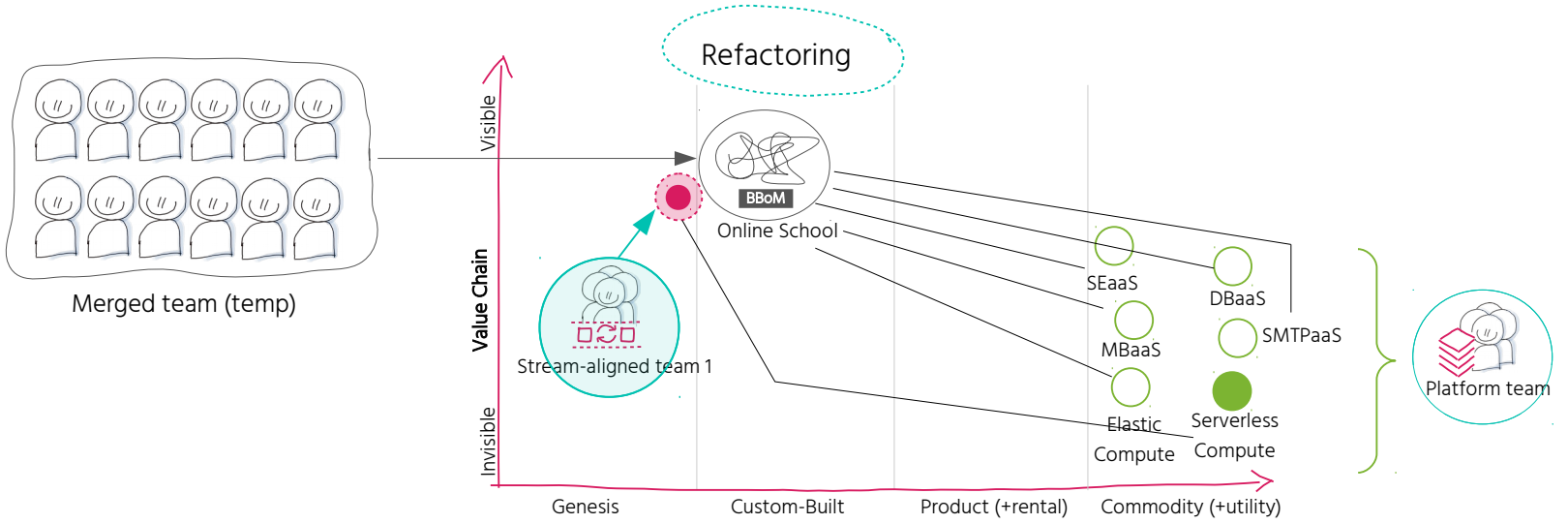


Implementing Flow Optimization

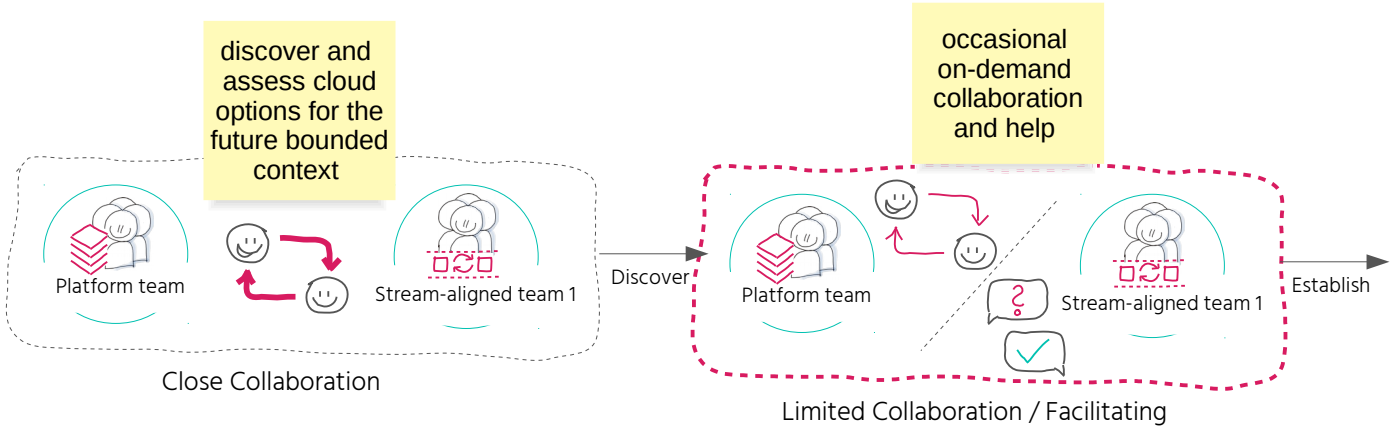


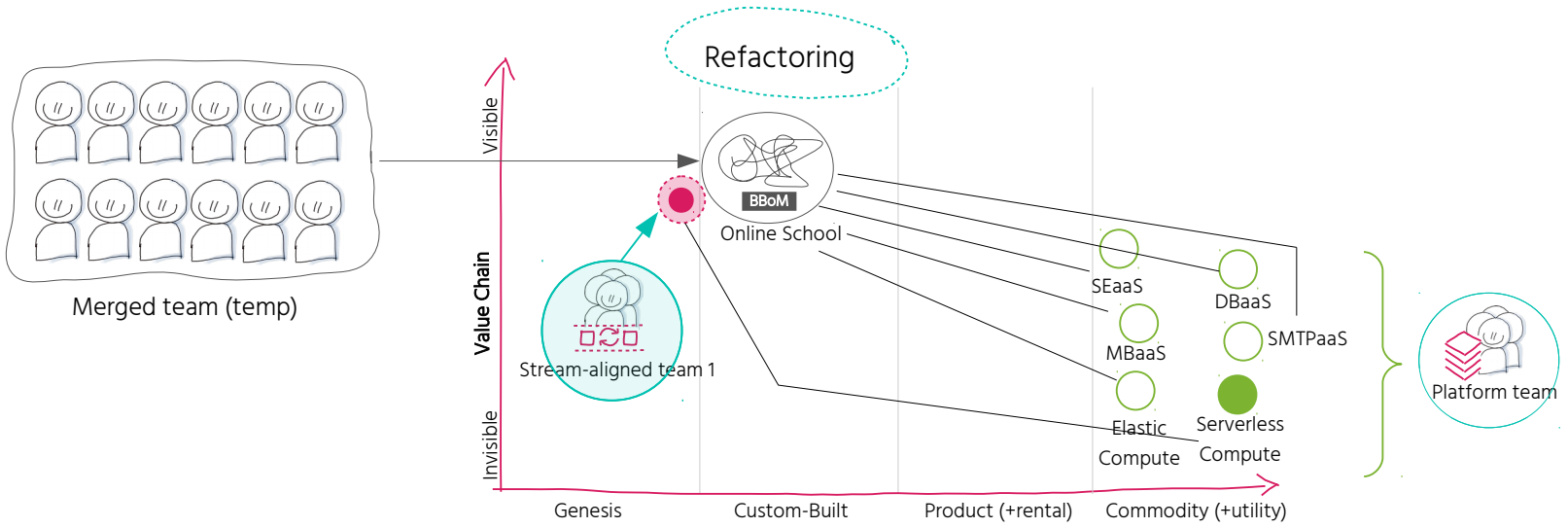
Implementing Flow Optimization



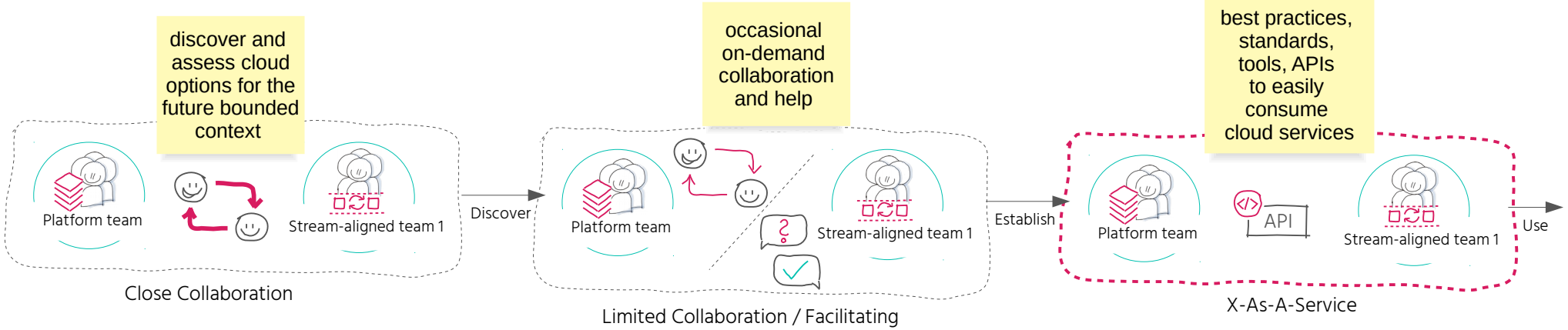


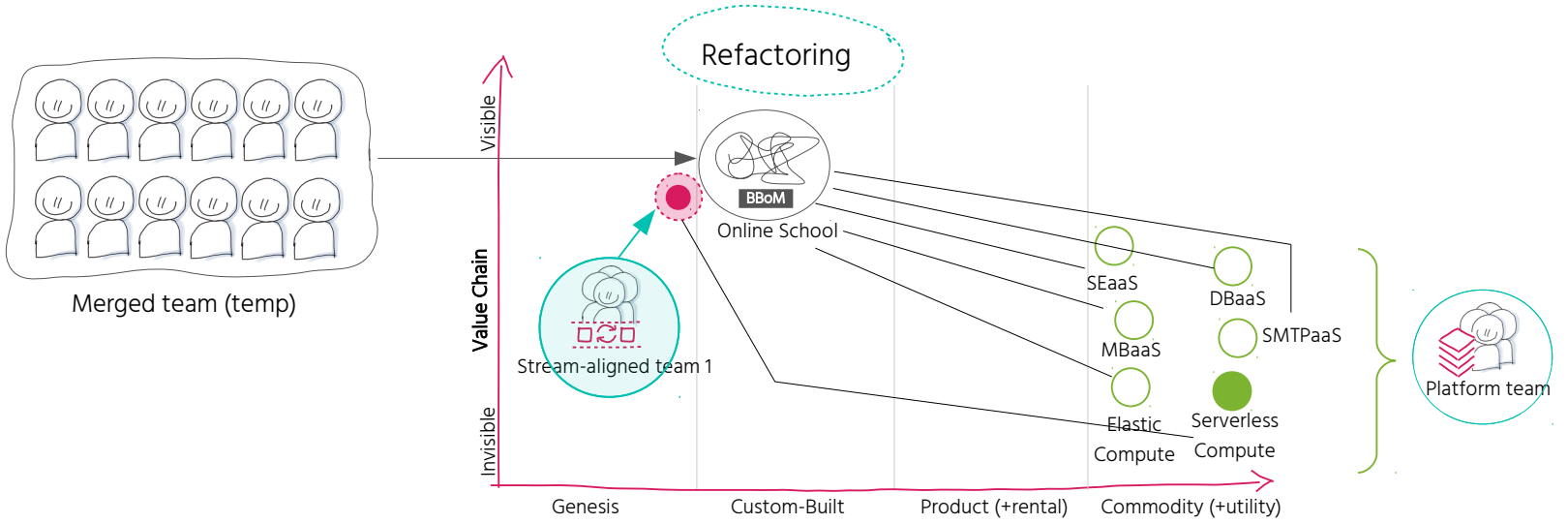
Implementing Flow Optimization



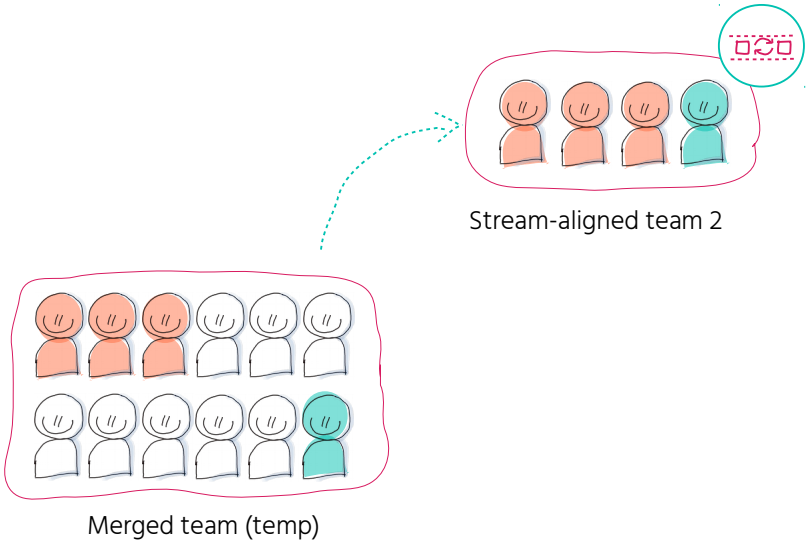


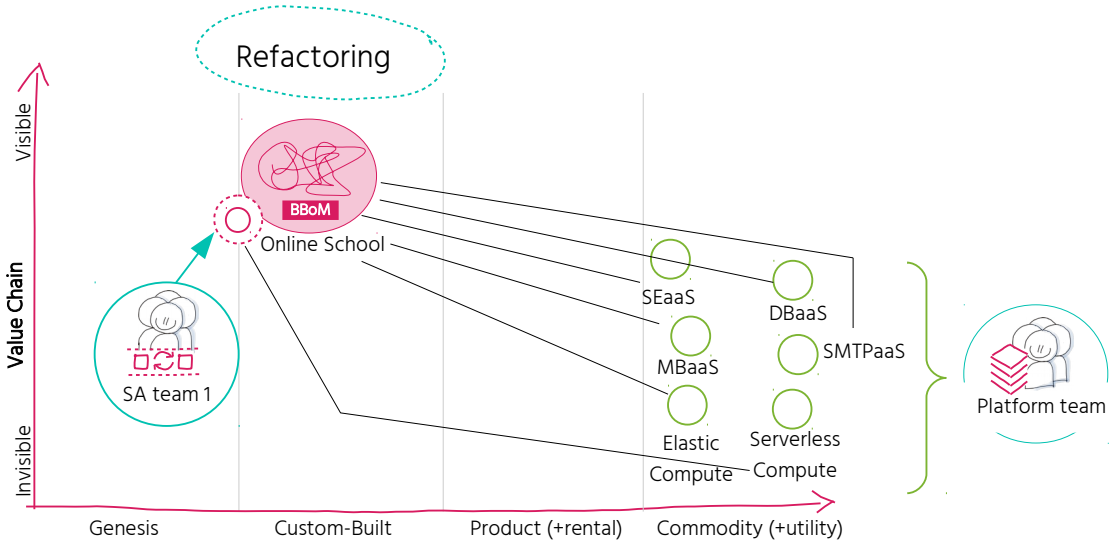
Implementing Flow Optimization



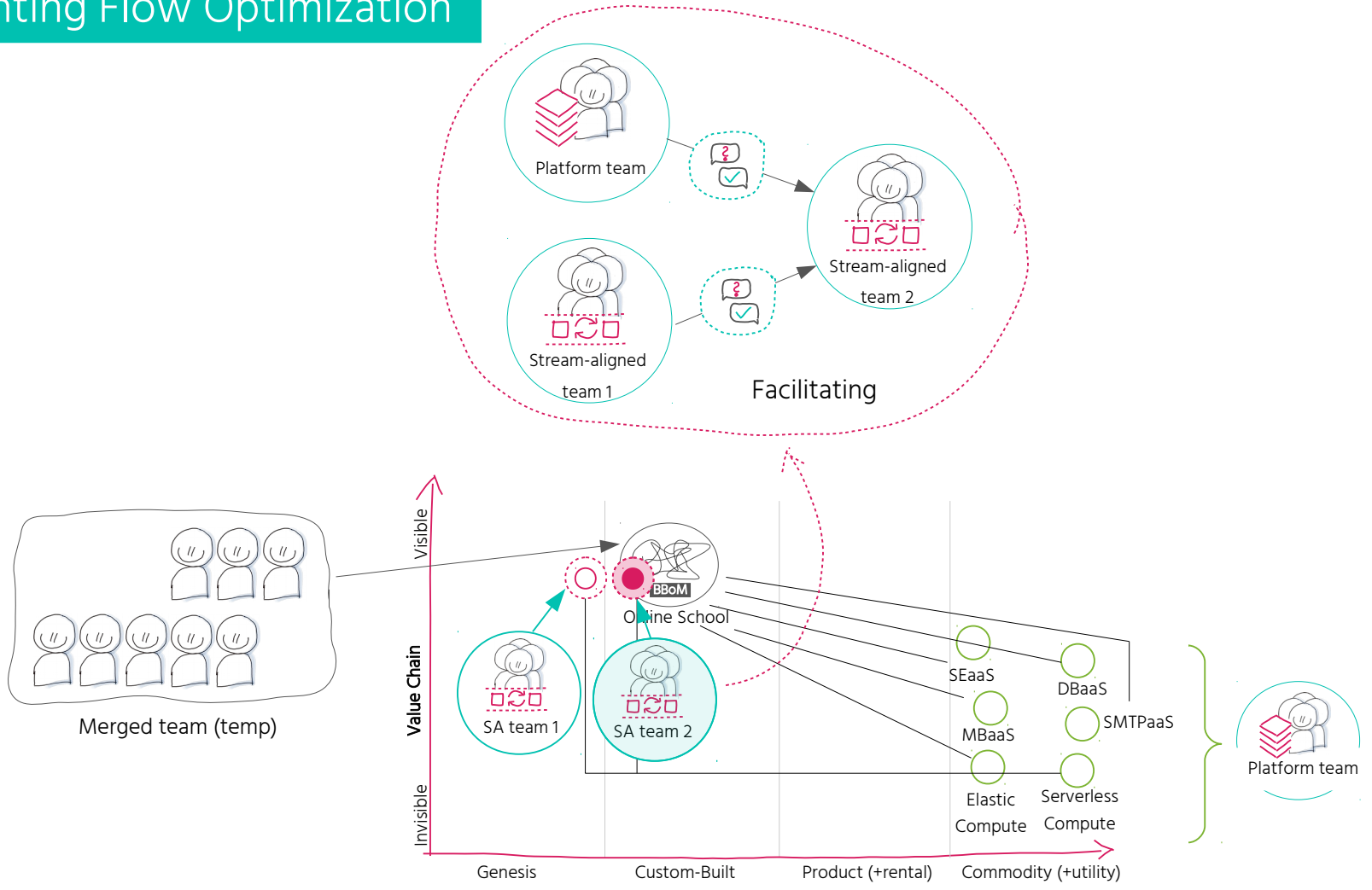


Implementing Flow Optimization

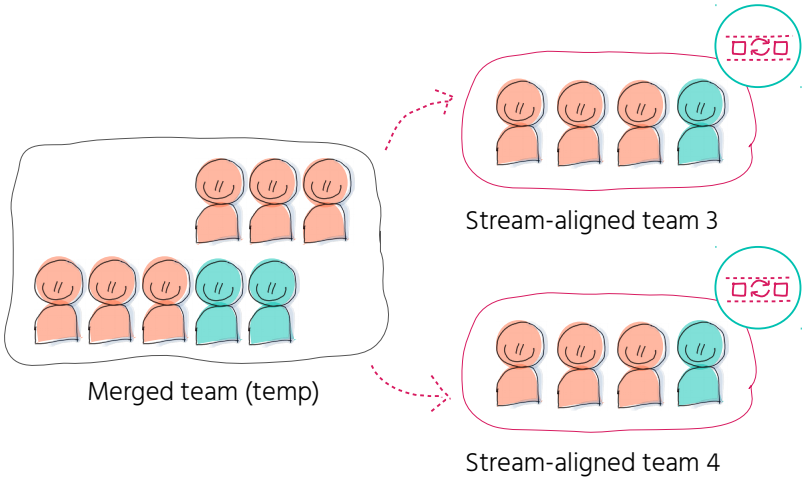


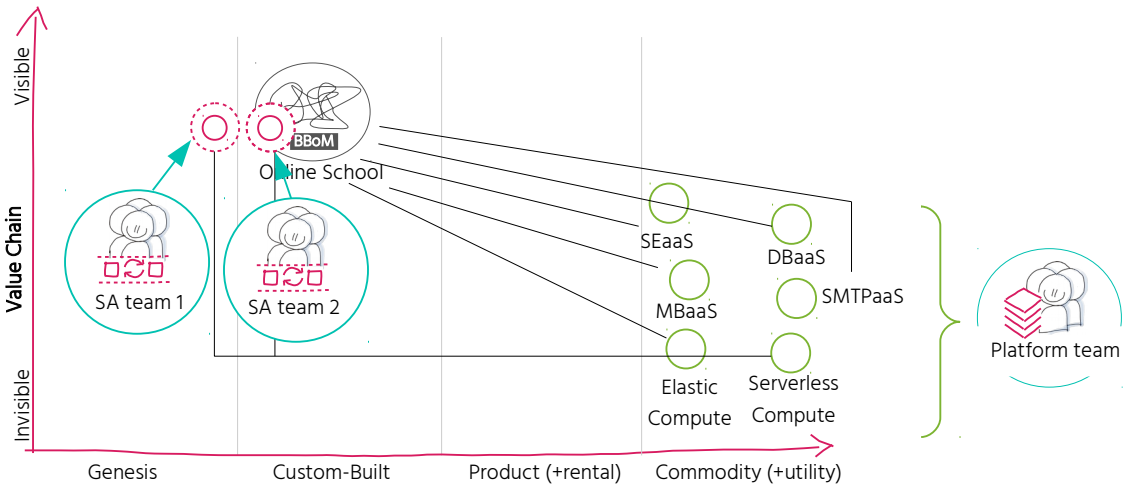


Implementing Flow Optimization

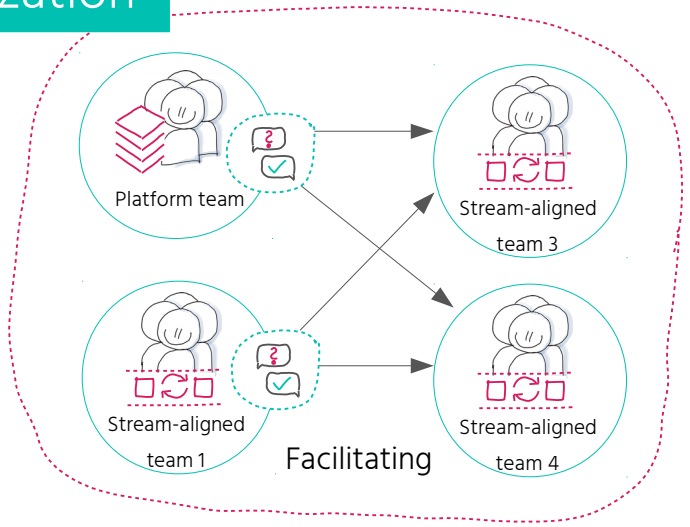


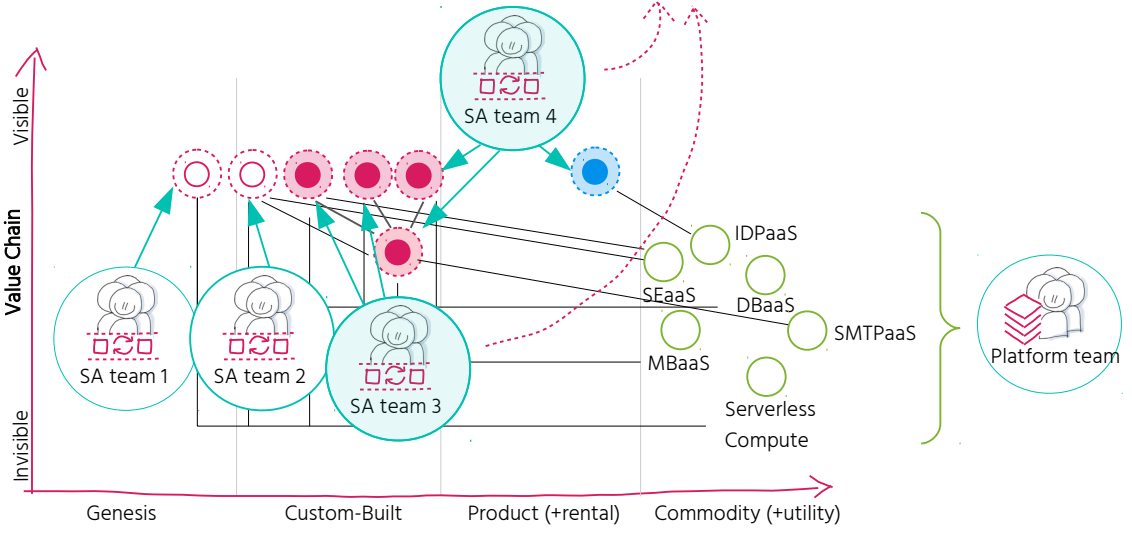
Implementing Flow Optimization



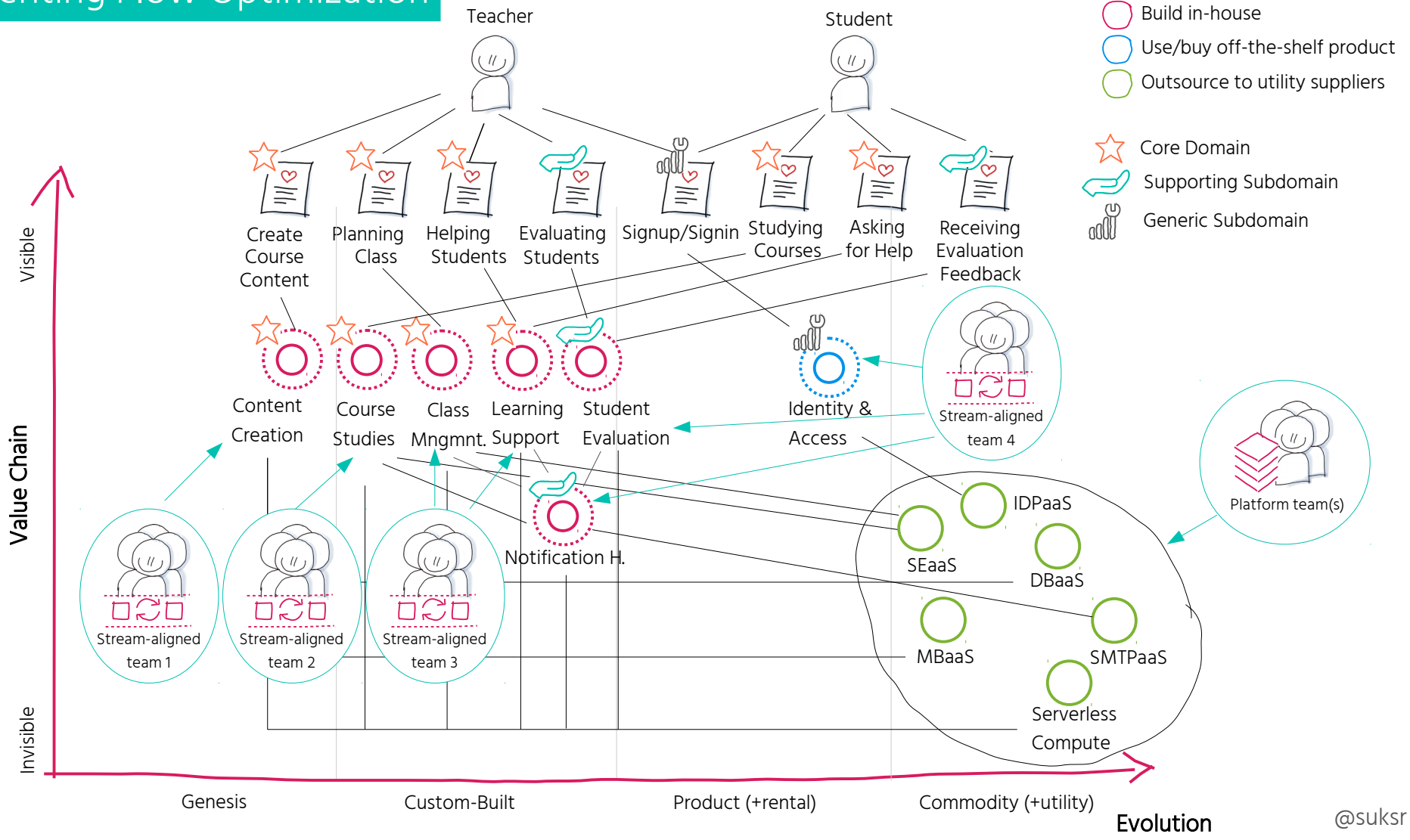


Implementing Flow Optimization



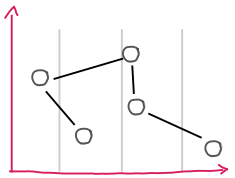


Implementing Flow Optimization



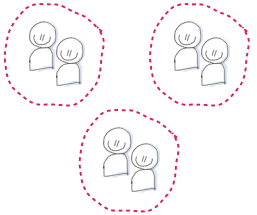
@suksr

Key Takeaways



Wardley Mapping

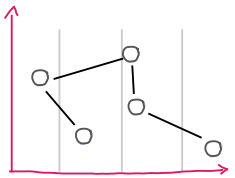
Domain-Driven Design



Team Topologies

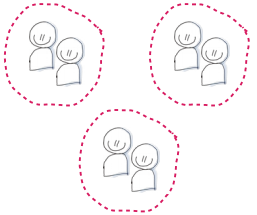
- Understanding the environment an organization is operating & competing in
- Gaining domain knowledge & discovering the core
- Knowing what components to build, buy/use, or outsource
- Decomposing the problem domain into modular boundext contexts
- Aligning teams and evolving their interactions to the system we build & the strategy we plan

Key Takeaways



Wardley Mapping

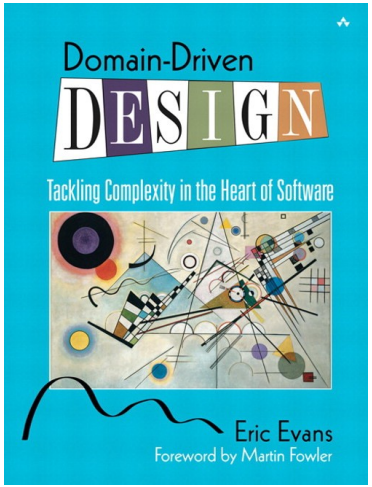
Domain-Driven Design

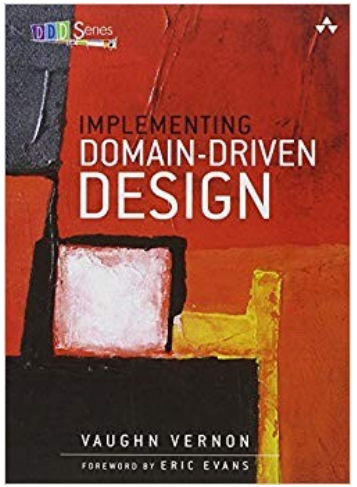


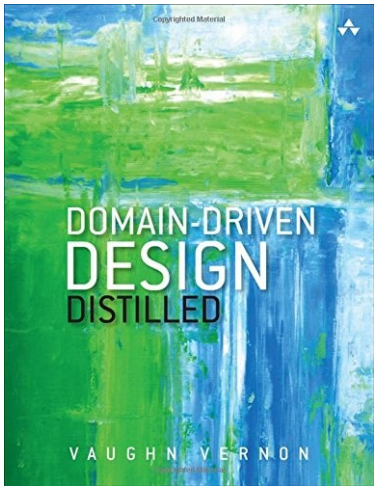
Team Topologies

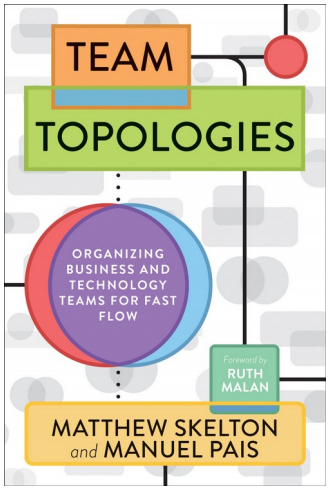
- Understanding the environment an organization is operating & competing in
- Gaining domain knowledge & discovering the core
- Knowing what components to build, buy/use, or outsource
- Decomposing the problem domain into modular bounded contexts
- Aligning teams and evolving their interactions to the system we build & the strategy we plan
- Identifying potential efficiency gaps
- Eliminating bottlenecks & increasing software delivery performance
- Being able to respond to changes quickly
- Optimizing for a a fast flow of change with the focus on improving the performance of a system as a whole

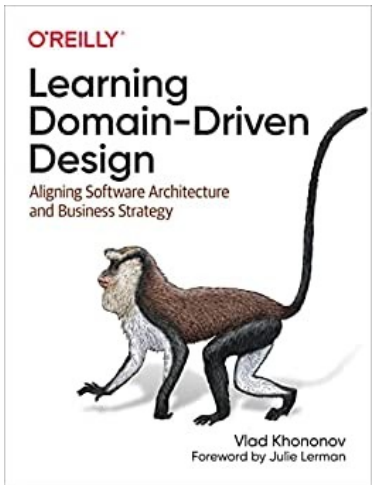
Some References

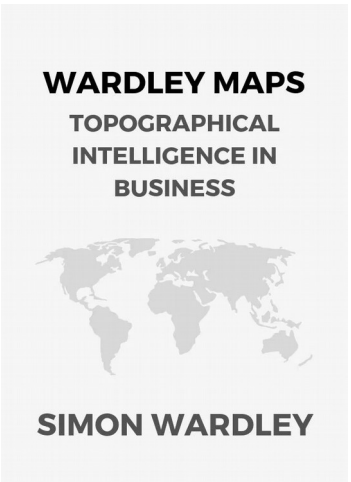


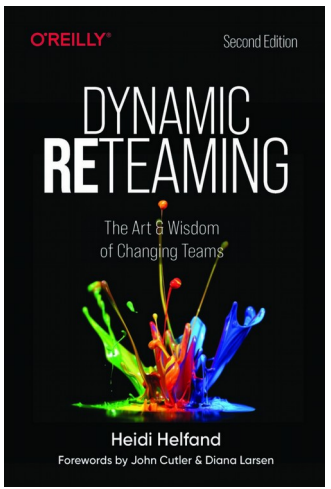






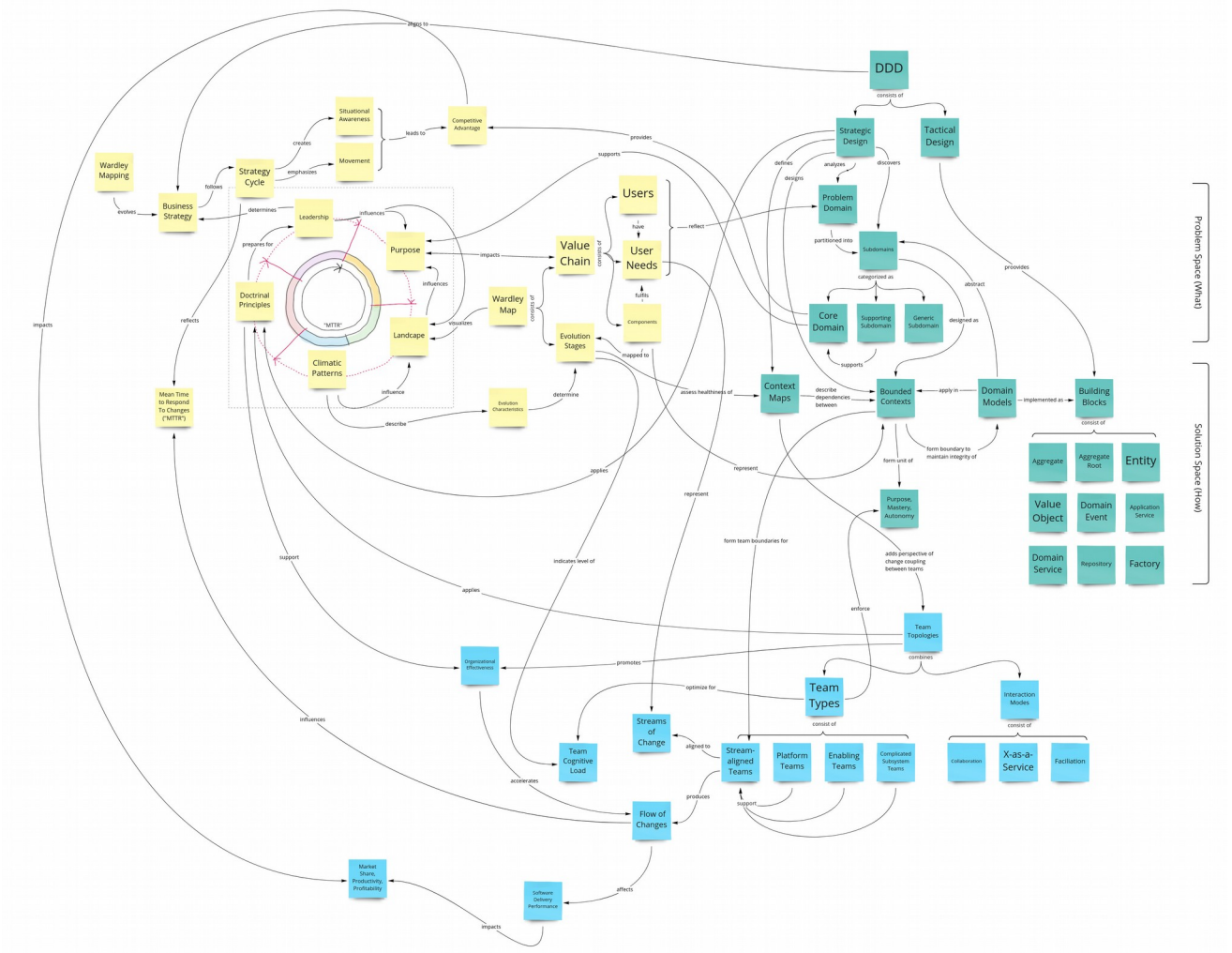


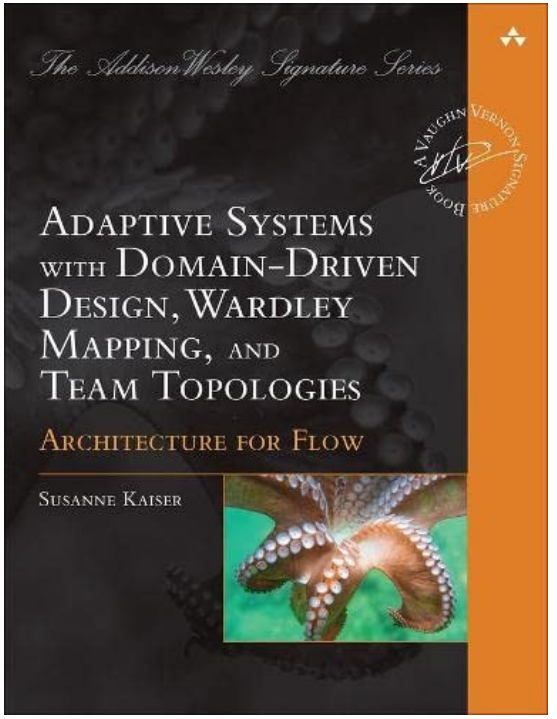




https://medium.com/wardleymaps
https://learnwardleymapping.com/
https://github.com/wardley-maps-community/awesome-wardley-maps
https://github.com/ddd-crew
https://www.dddheuristics.com

If you are interested in more details ...





@suksr

THANK YOU

Your questions
"Kesselhaus"

https://app.sli.do/event/tGEgNGuZACjQKvWufmDD9w/live/questions?w=AQpQV