

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

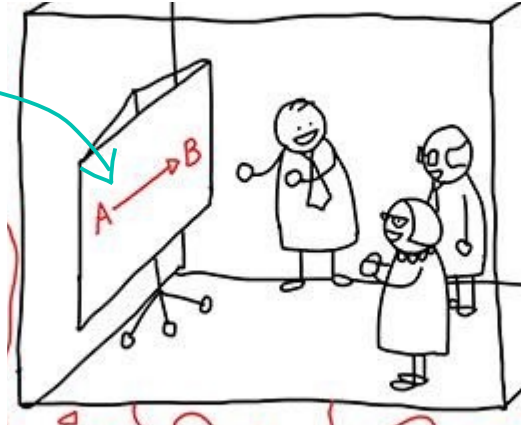
Susanne Kaiser

Independent Tech Consultant

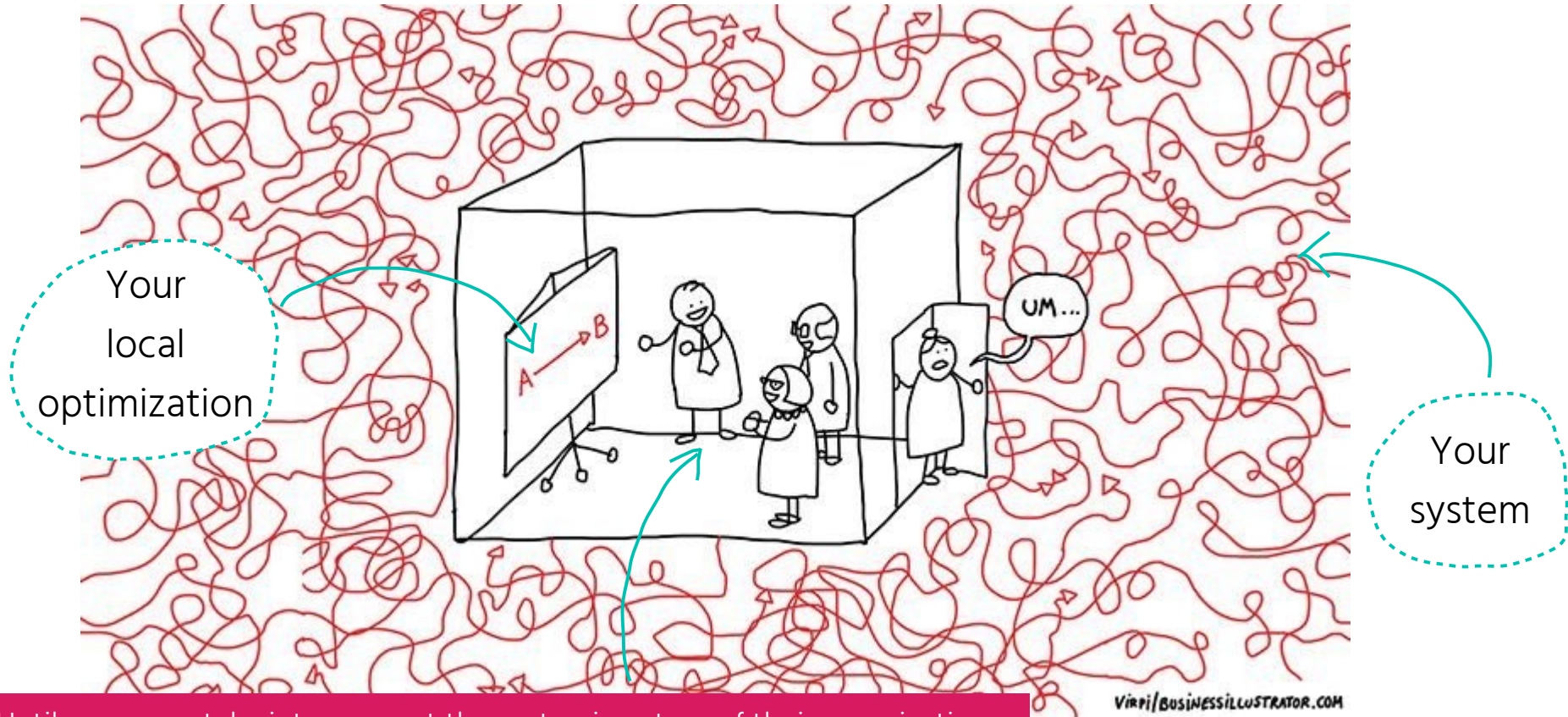
@suksr

# Problem with Local Optimization

Your  
local  
optimization



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



VIRPI/BUSINESSILLUSTRATOR.COM

“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

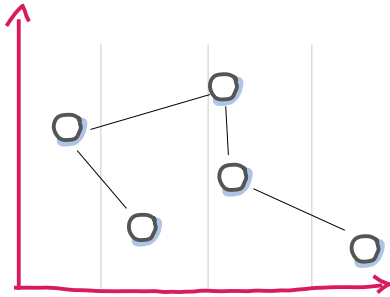
“Doing the wrong thing right is not nearly as good as doing the right thing wrong”

Dr. Russell Ackoff

# Three Perspectives to Build Adaptive Systems

1.

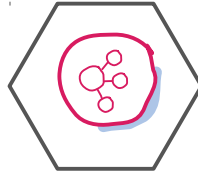
Business-Strategy



w/ Wardley Mapping

2.

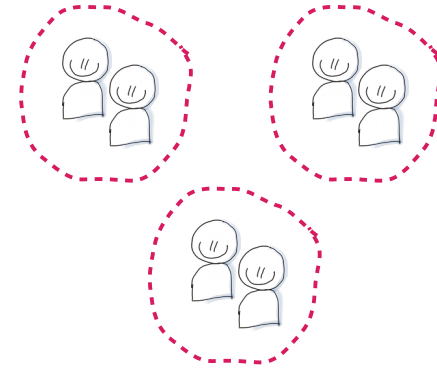
Software-Design/  
-Architecture



w/ Domain-Driven Design

3.

Team-Organization



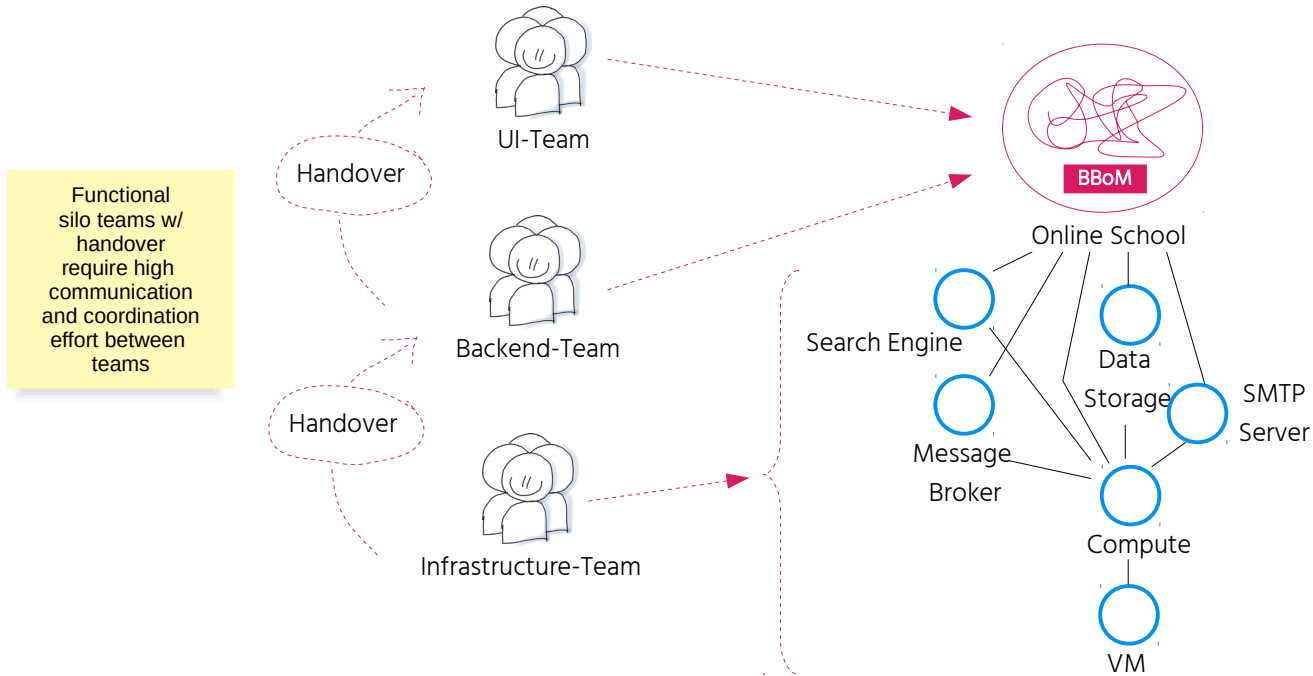
w/ Team Topologies

# Online School

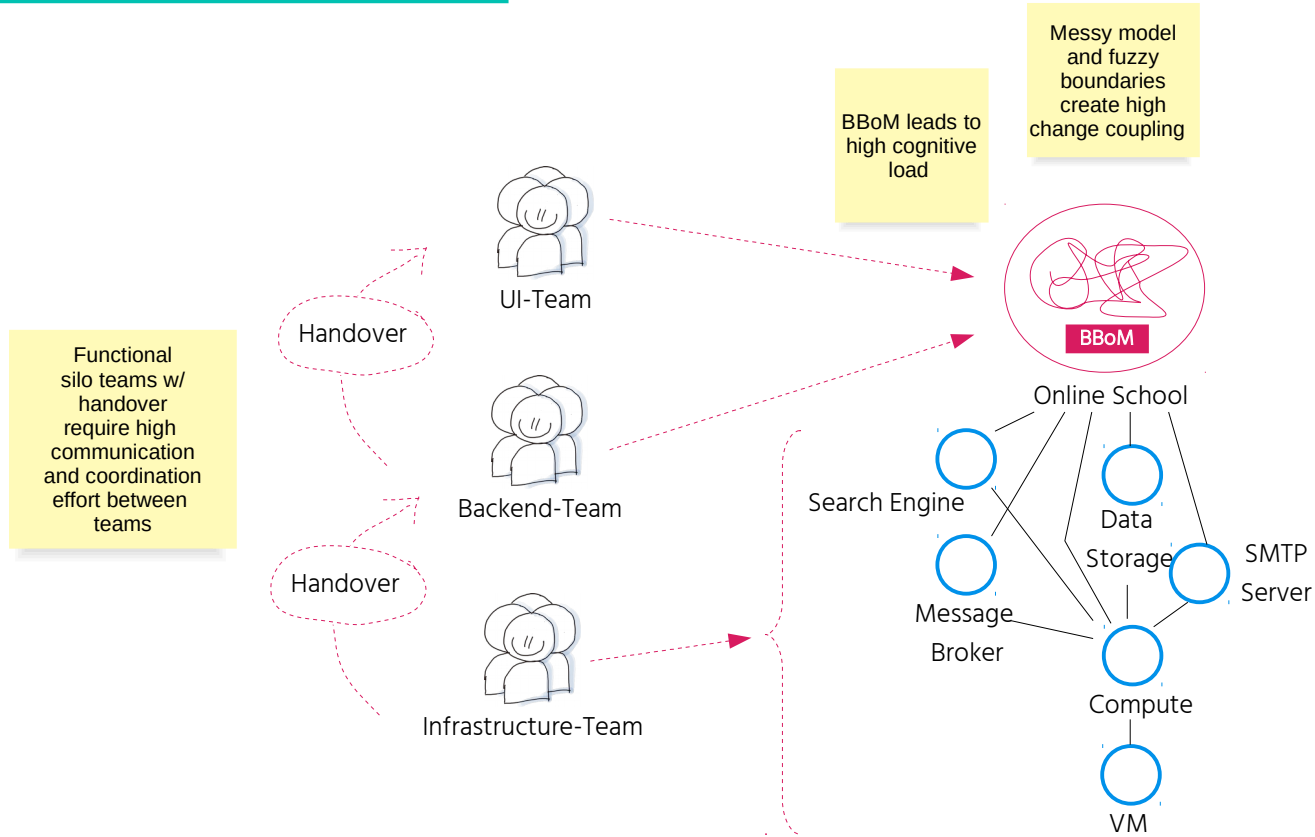


Source: <https://www.food-management.com>

# Their Current Challenges

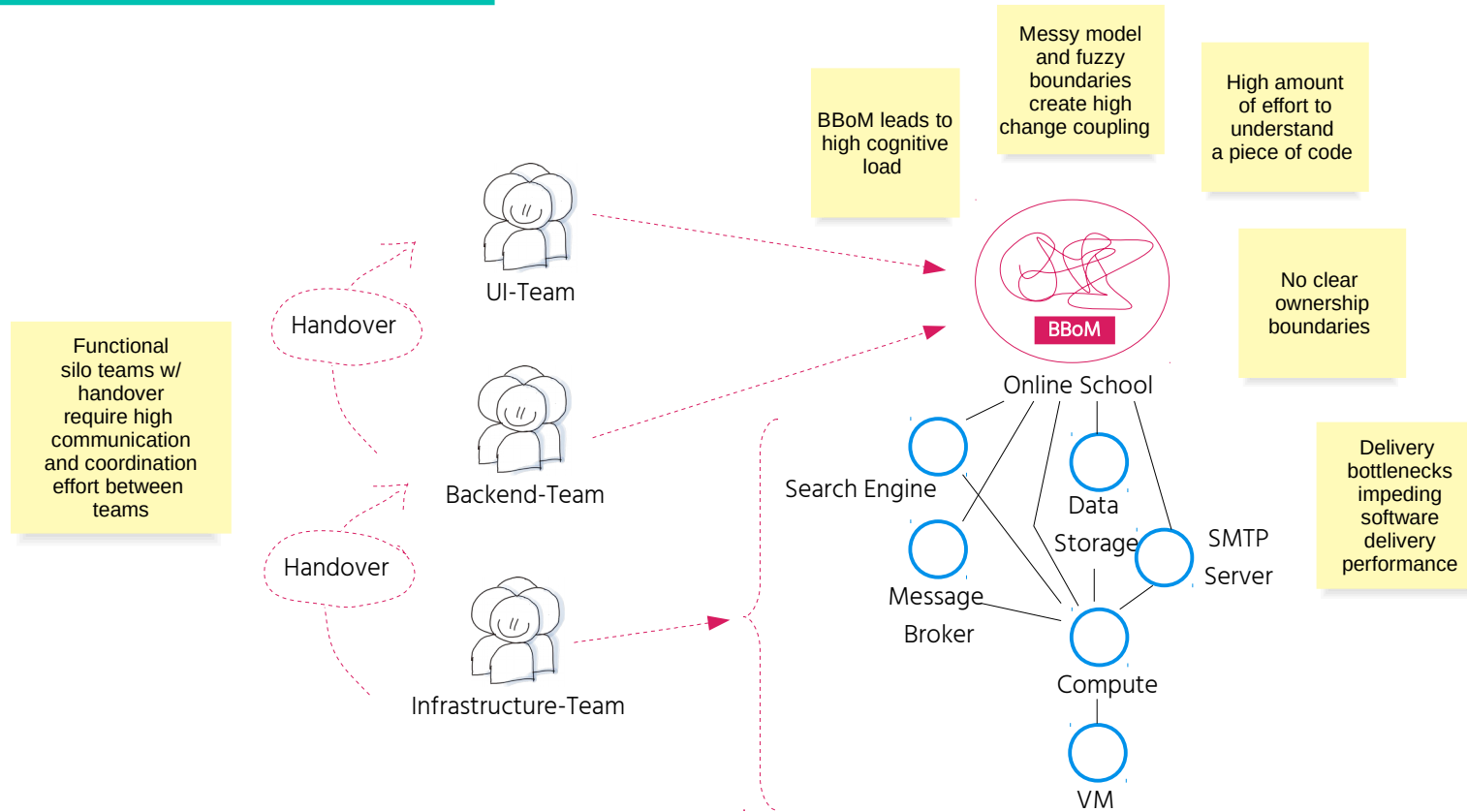


# Their Current Challenges

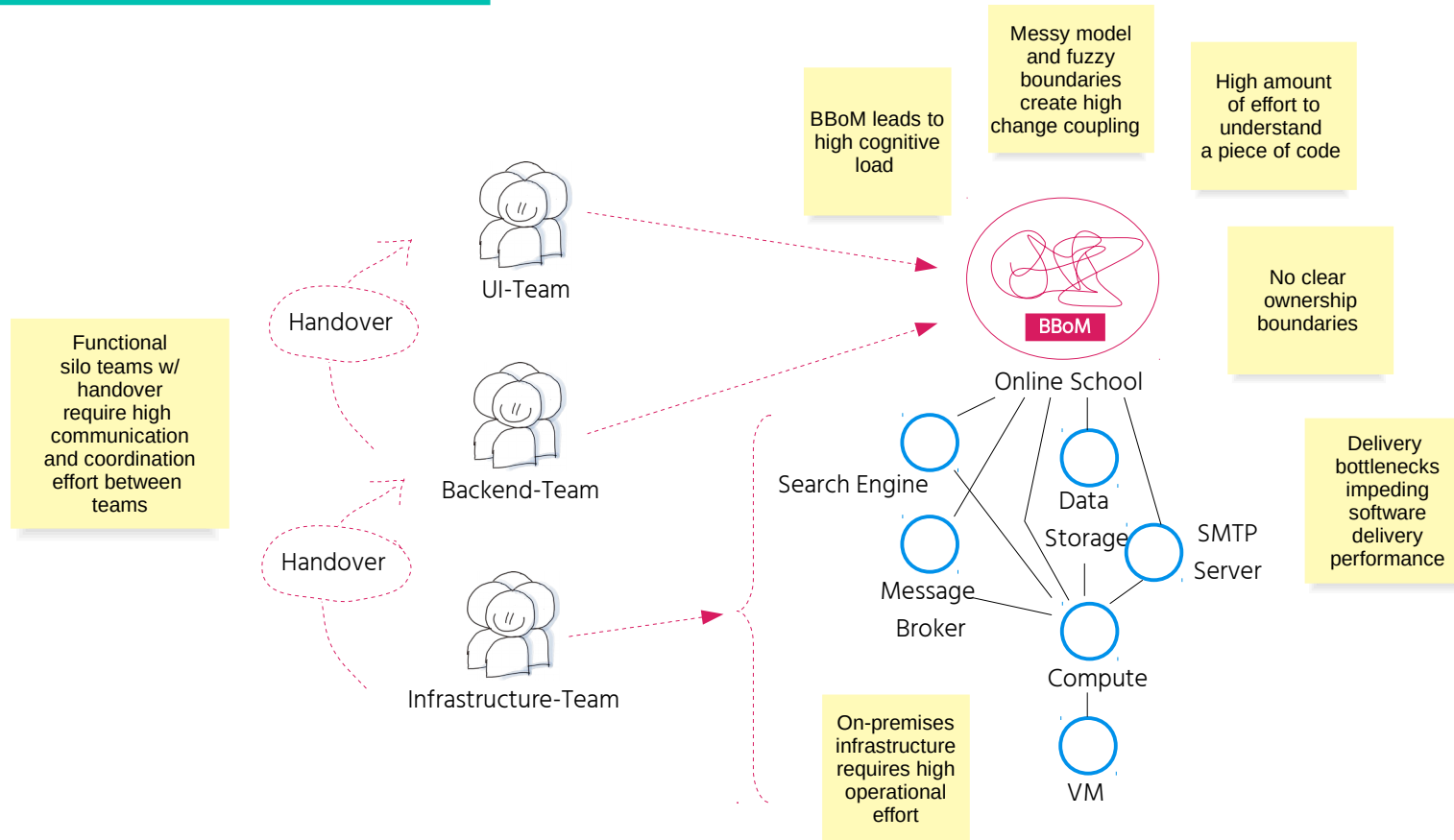




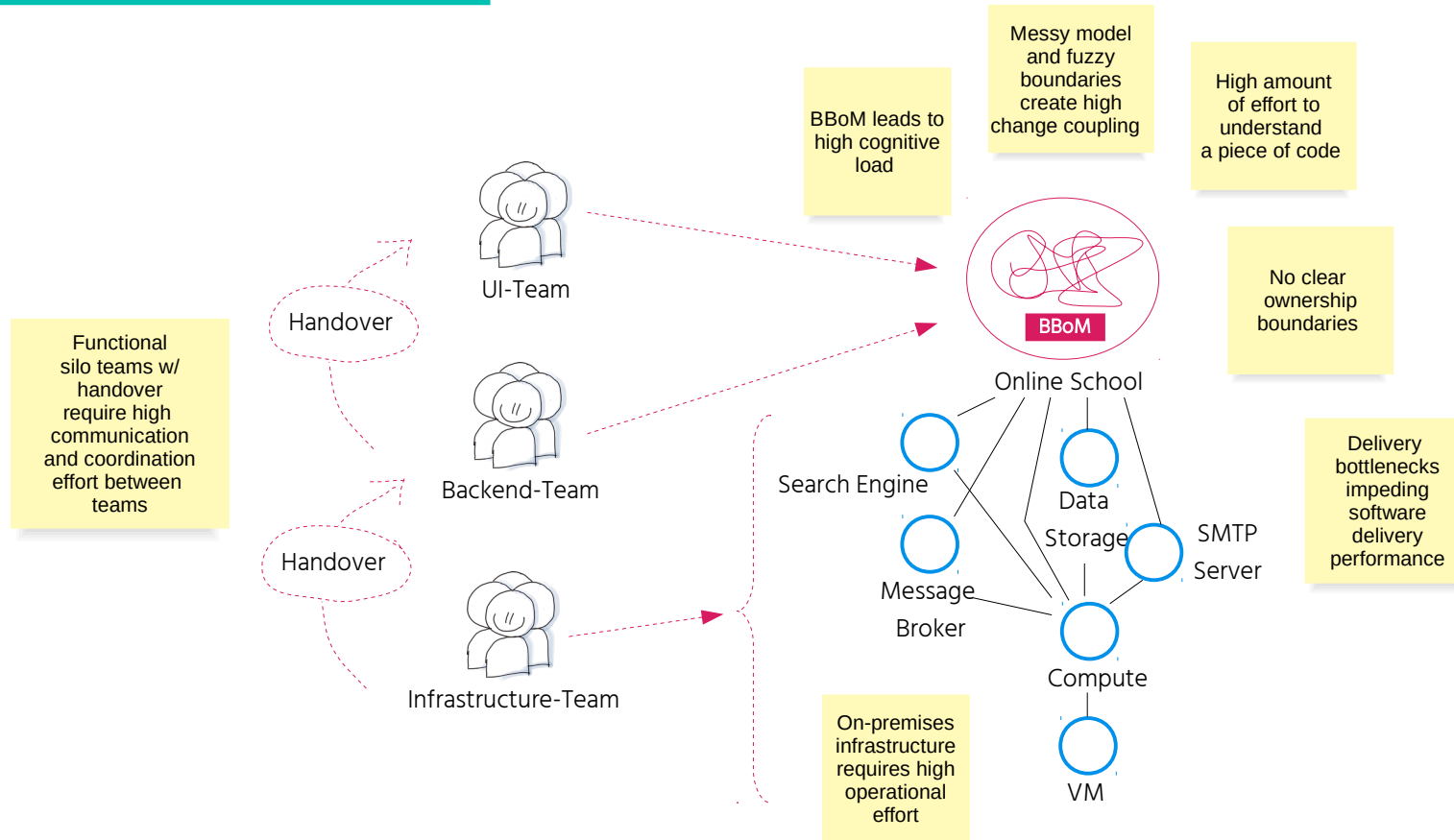
# Their Current Challenges



# Their Current Challenges



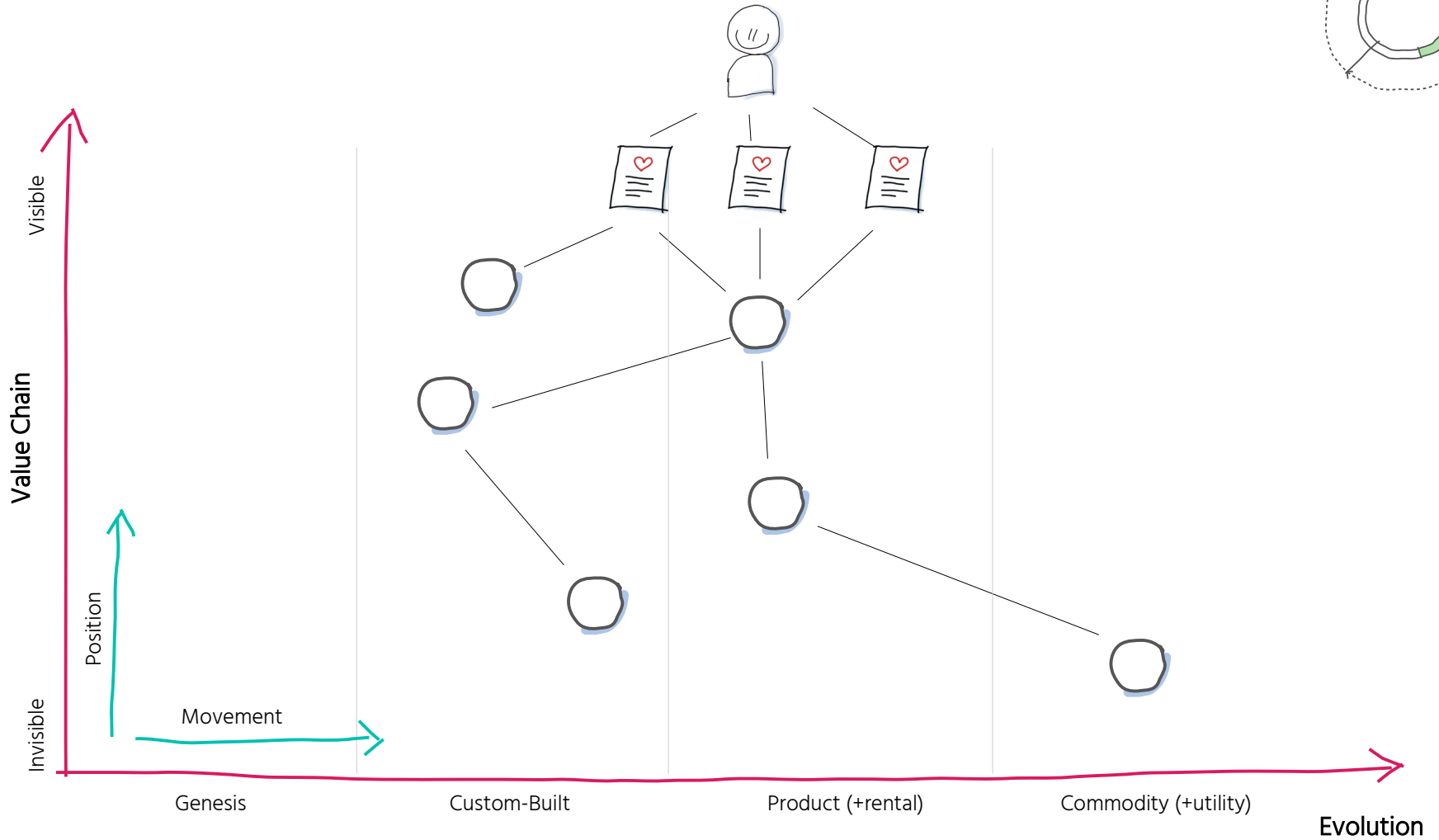
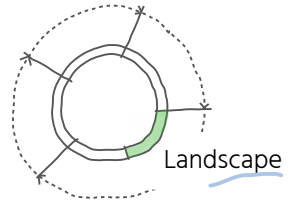
# Their Current Challenges



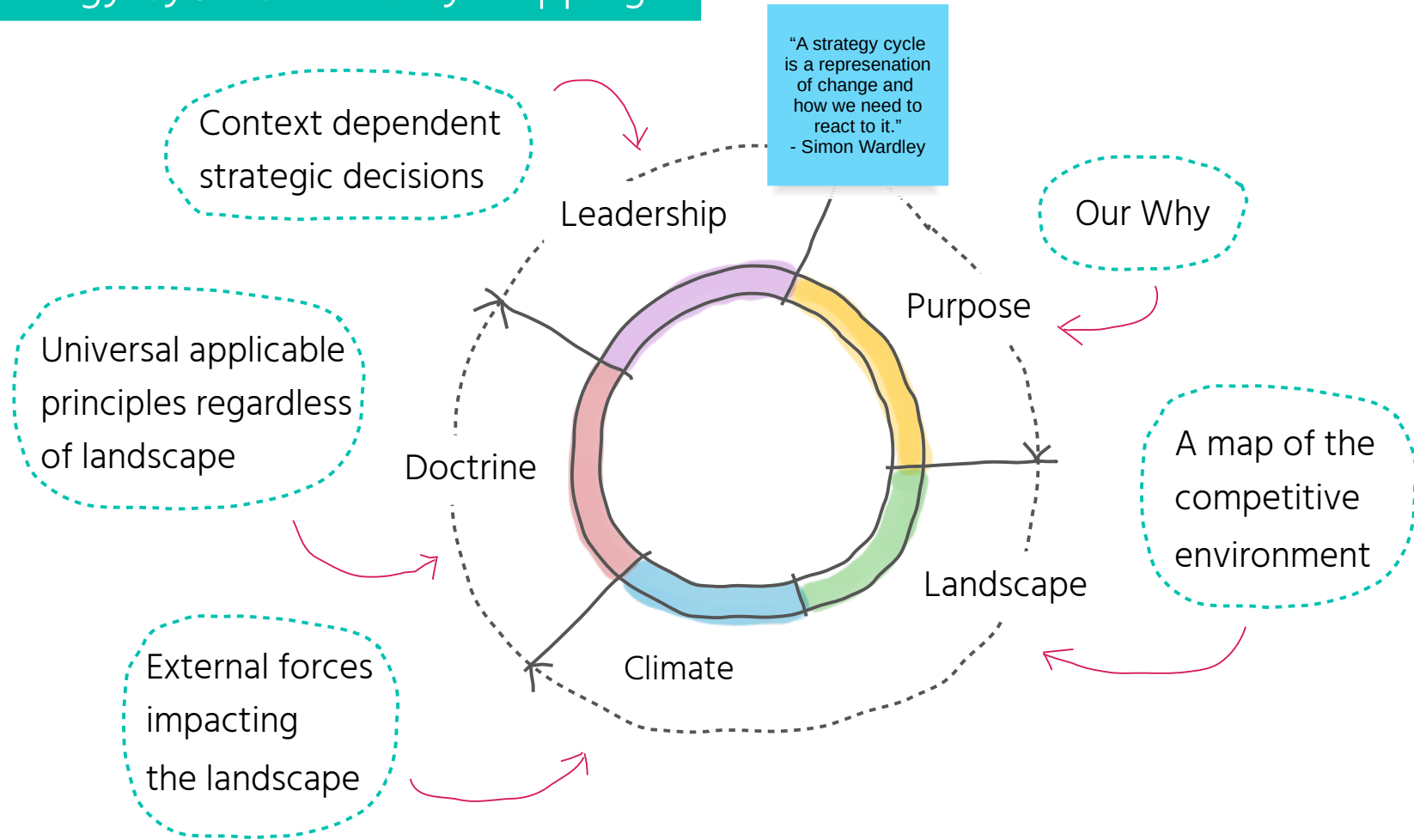
“Local optimization does not improve the performance of the whole.”

Dr. Russell Ackoff

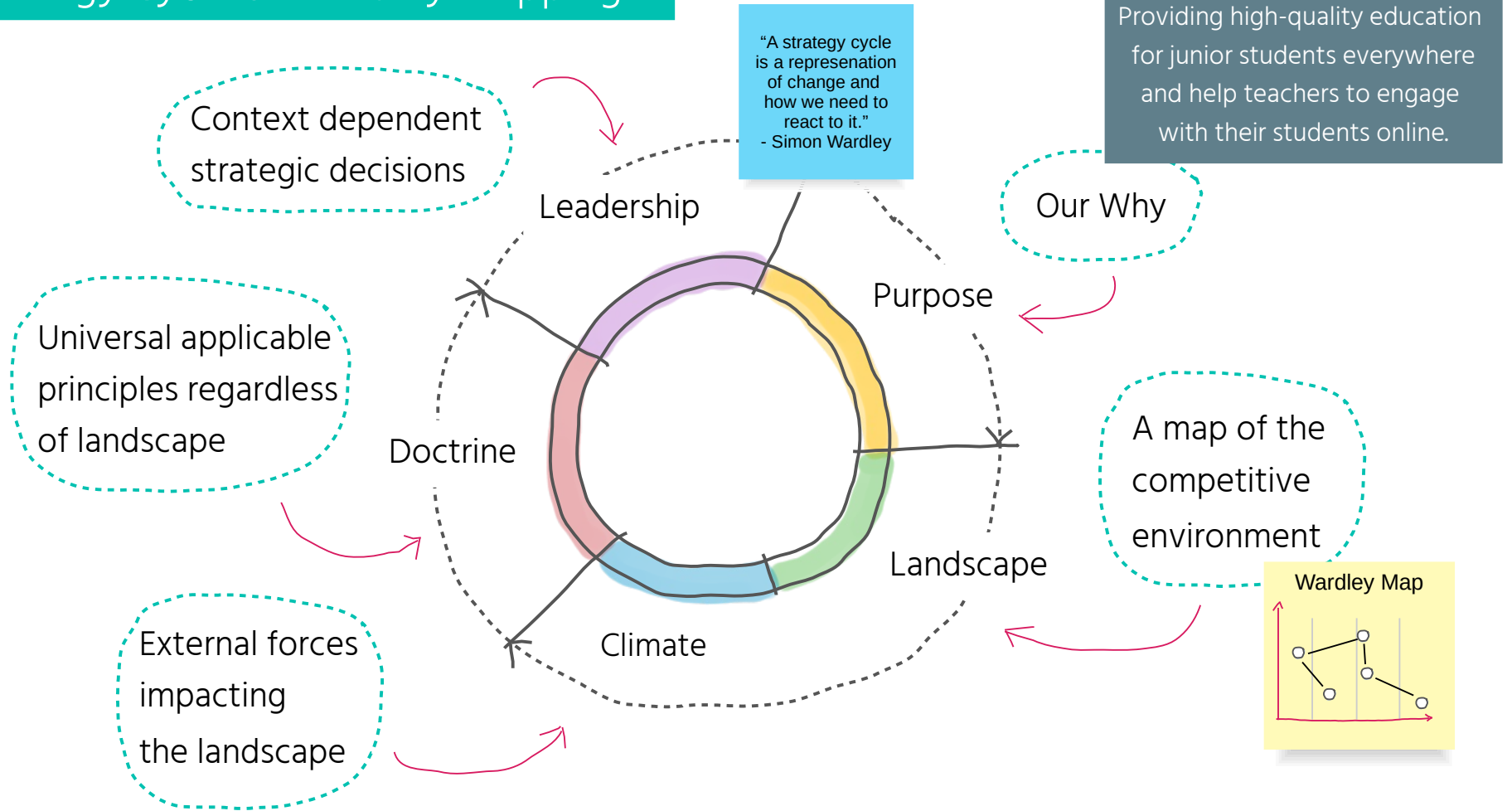
# Visualizing the Landscape w/ a Wardley Map



# The Strategy Cycle of Wardley Mapping

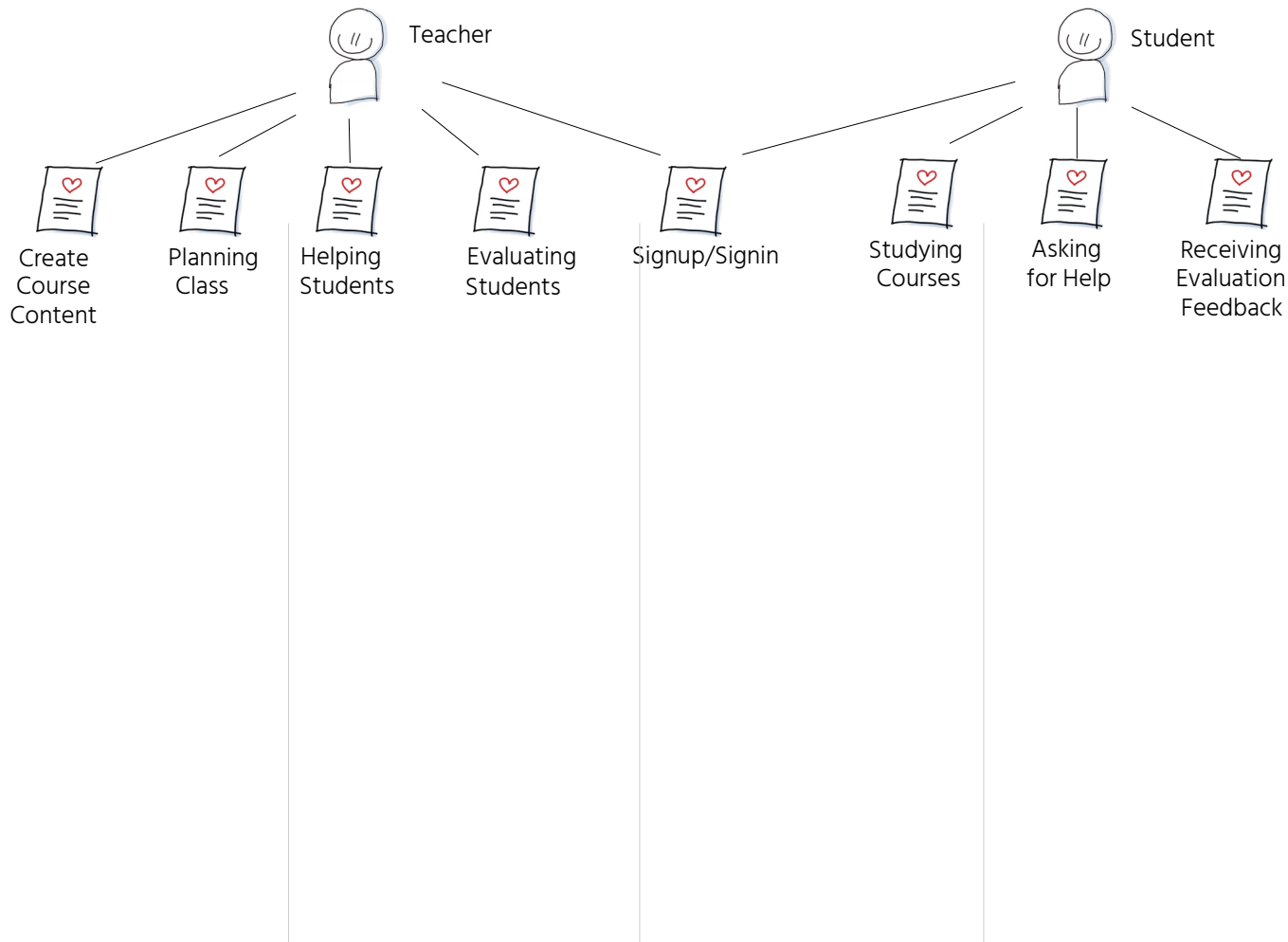


# The Strategy Cycle of Wardley Mapping

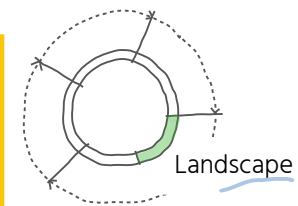


# The Landscape of the Current State

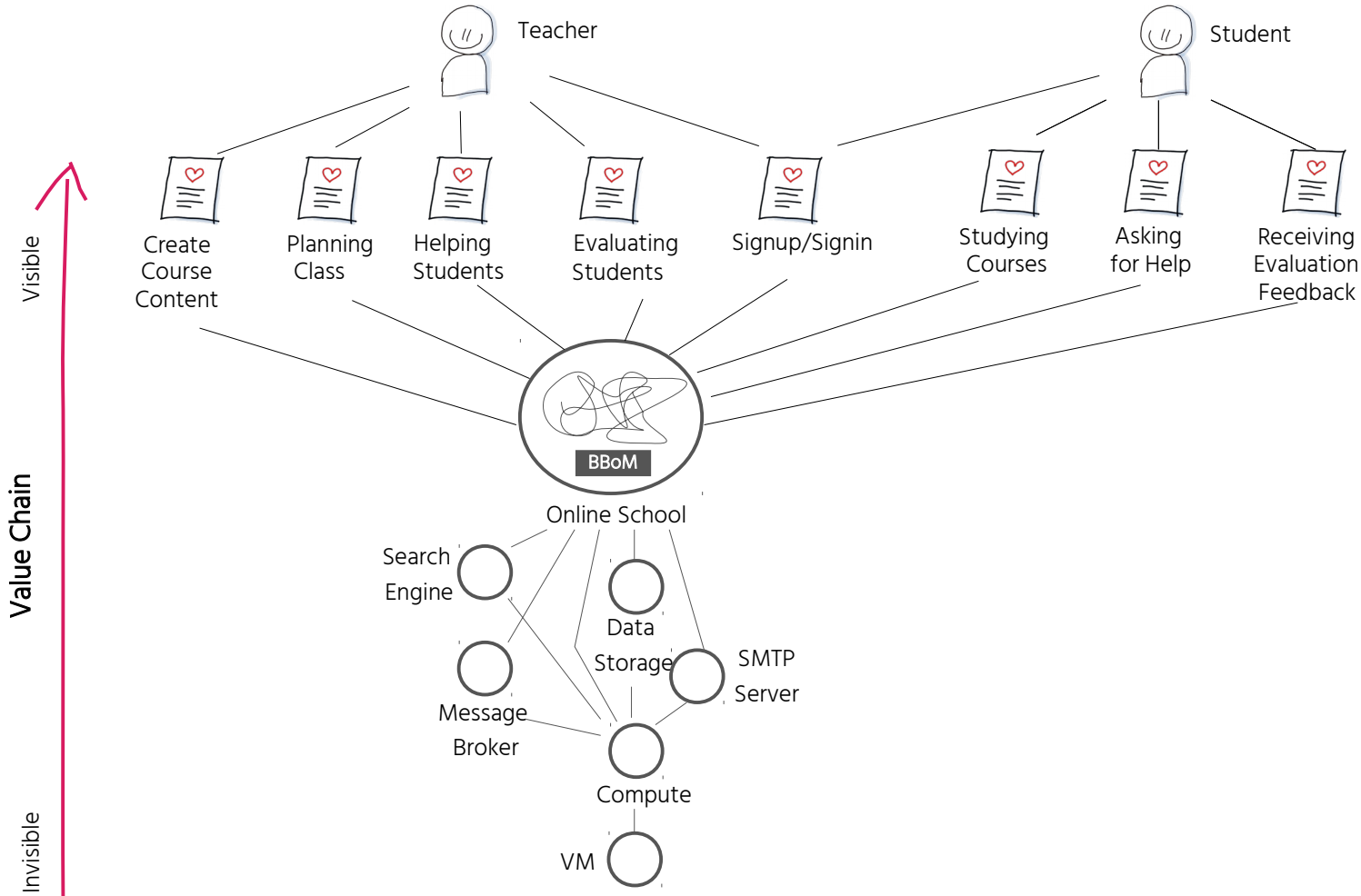
Visible  
Value Chain  
Invisible



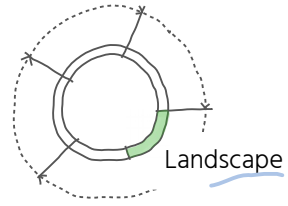
- 1 Identify users
- 2 Identify user needs



# The Landscape of the Current State

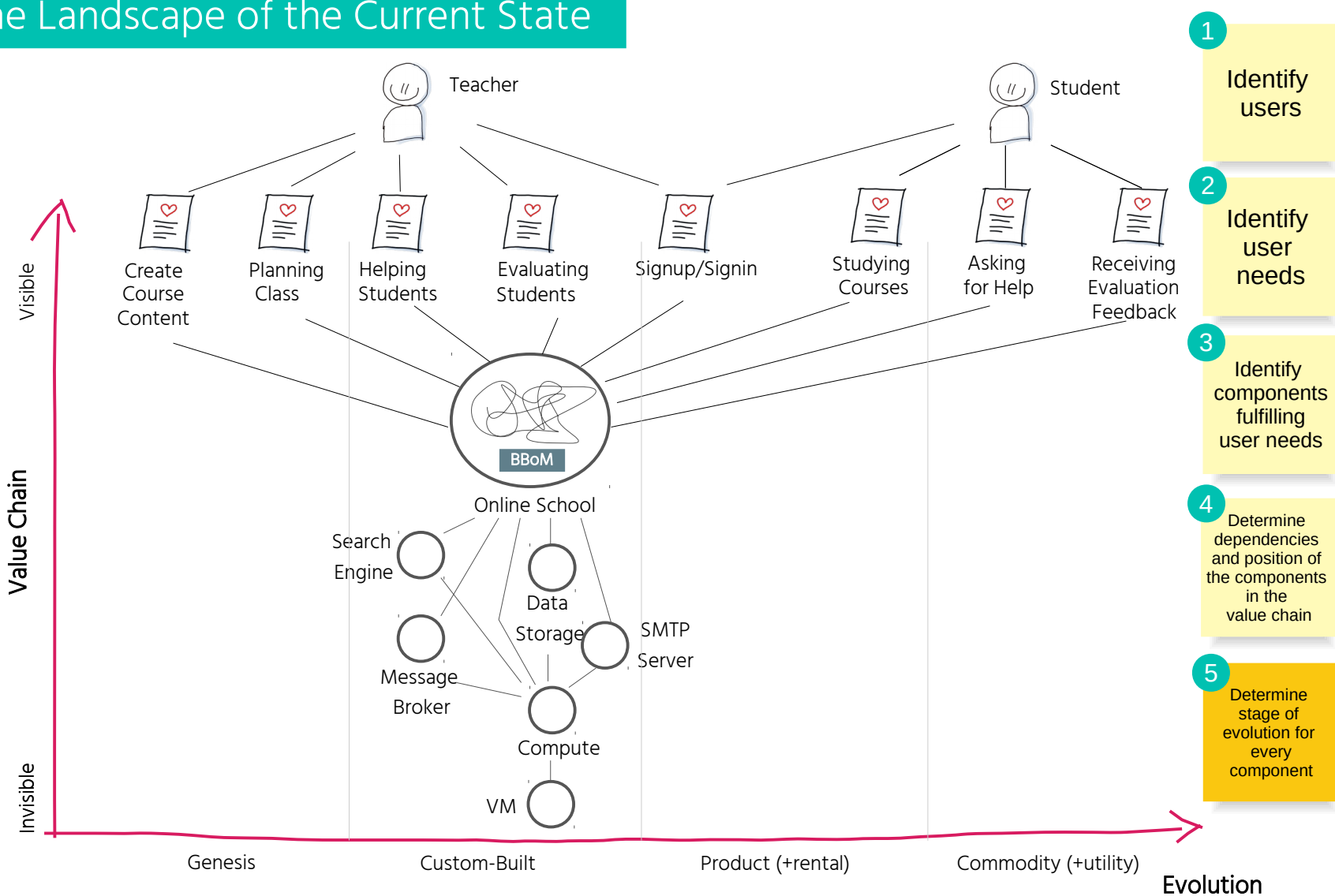


- 1 Identify users
- 2 Identify user needs
- 3 Identify components fulfilling user needs
- 4 Determine dependencies and position of the components in the value chain

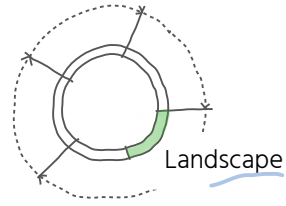




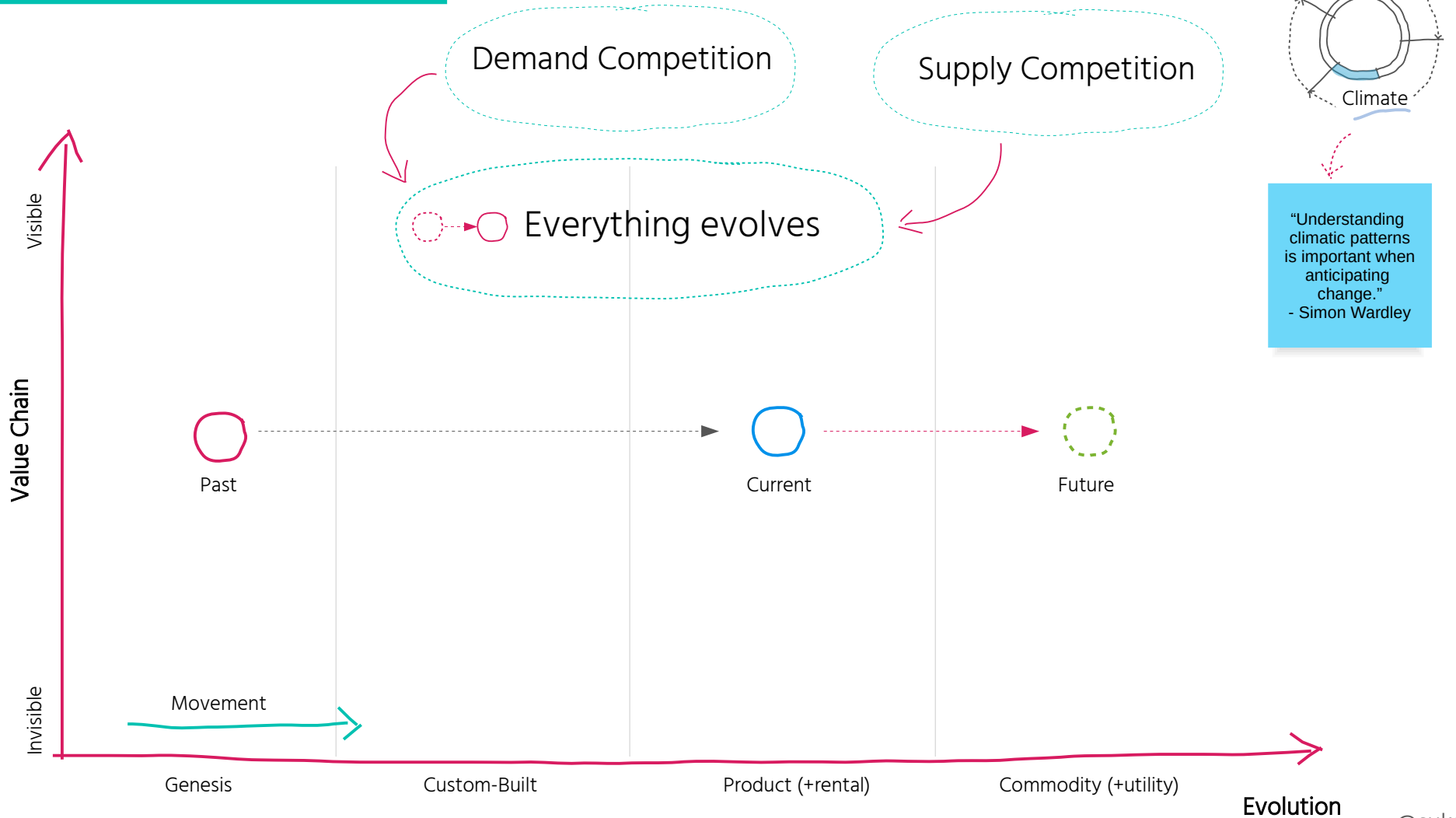
# The Landscape of the Current State



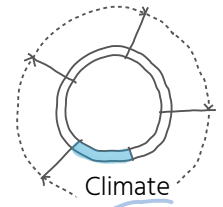
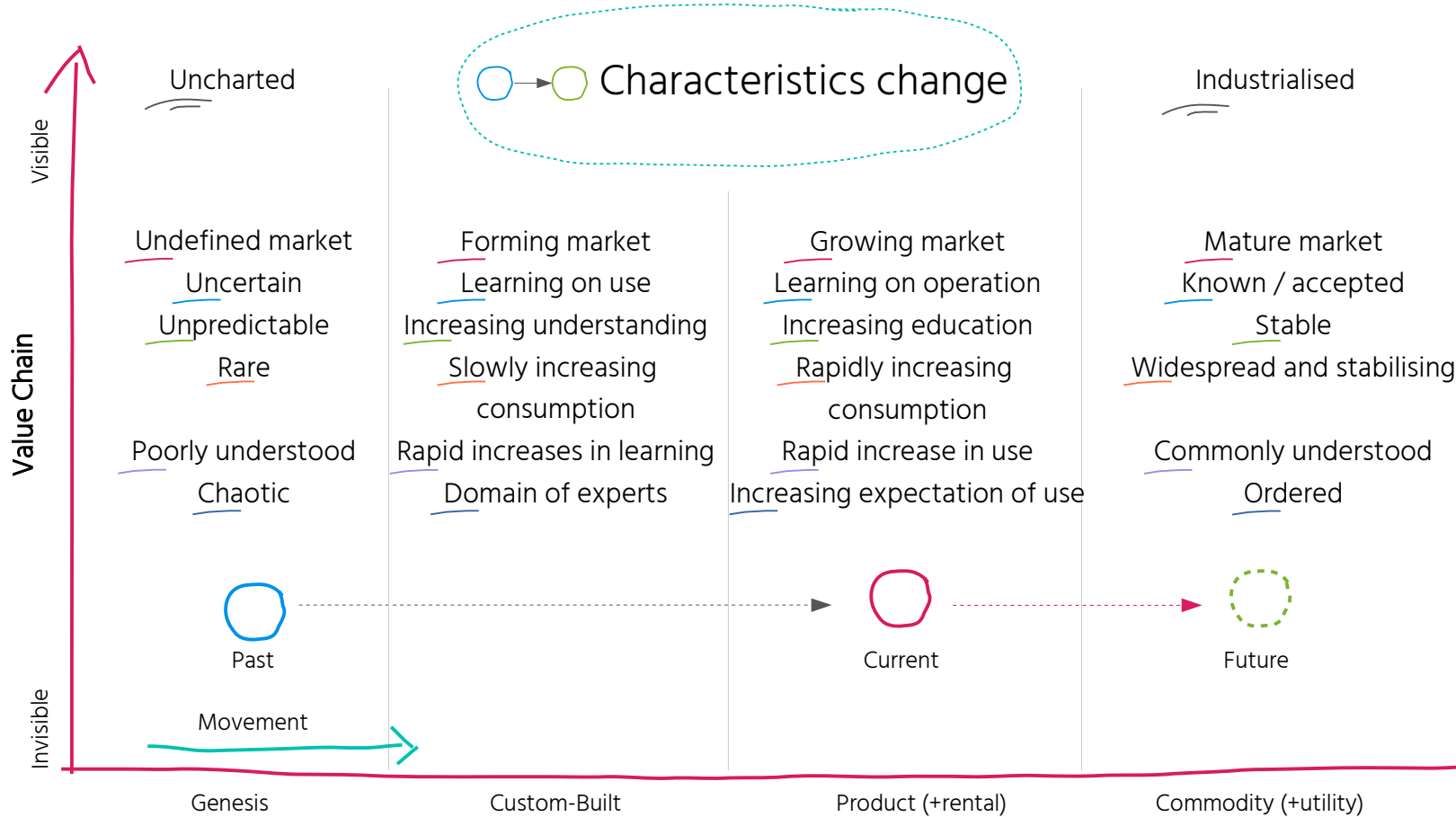
- 1 Identify users
- 2 Identify user needs
- 3 Identify components fulfilling user needs
- 4 Determine dependencies and position of the components in the value chain
- 5 Determine stage of evolution for every component



# The Climatic Patterns (extract)

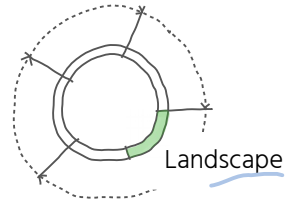
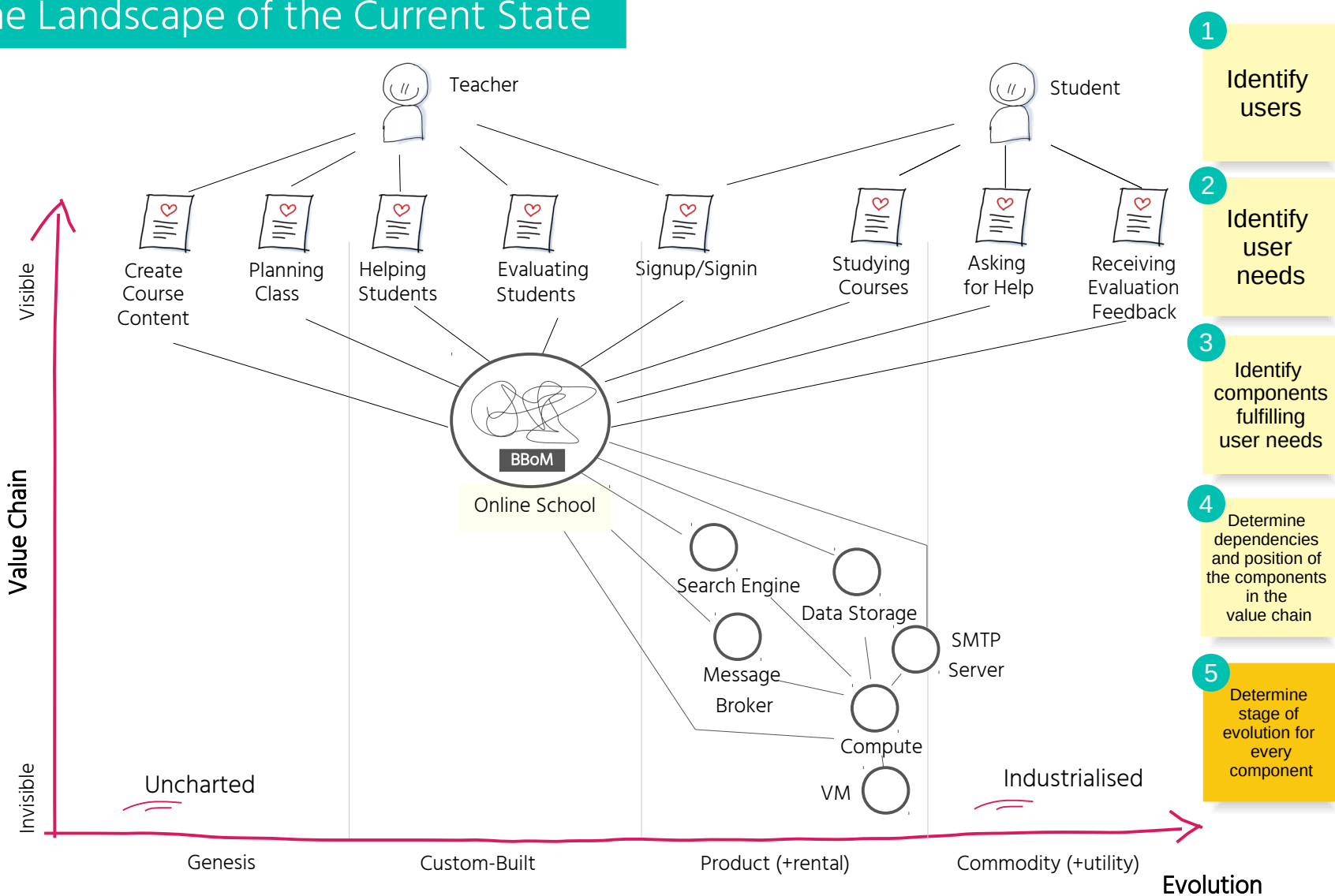


# The Climatic Patterns (extract)



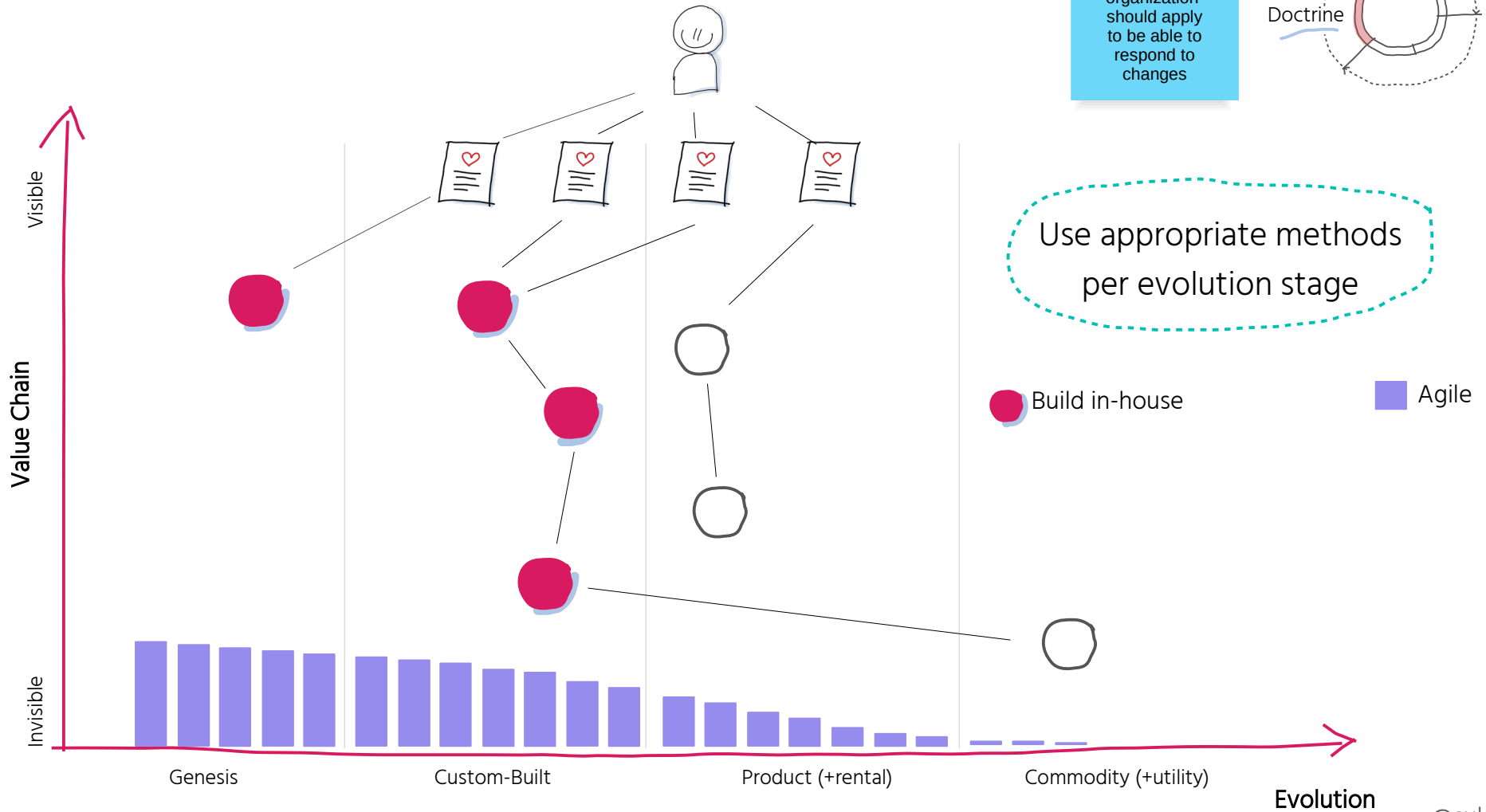
“Understanding climatic patterns is important when anticipating change.”  
- Simon Wardley

# The Landscape of the Current State



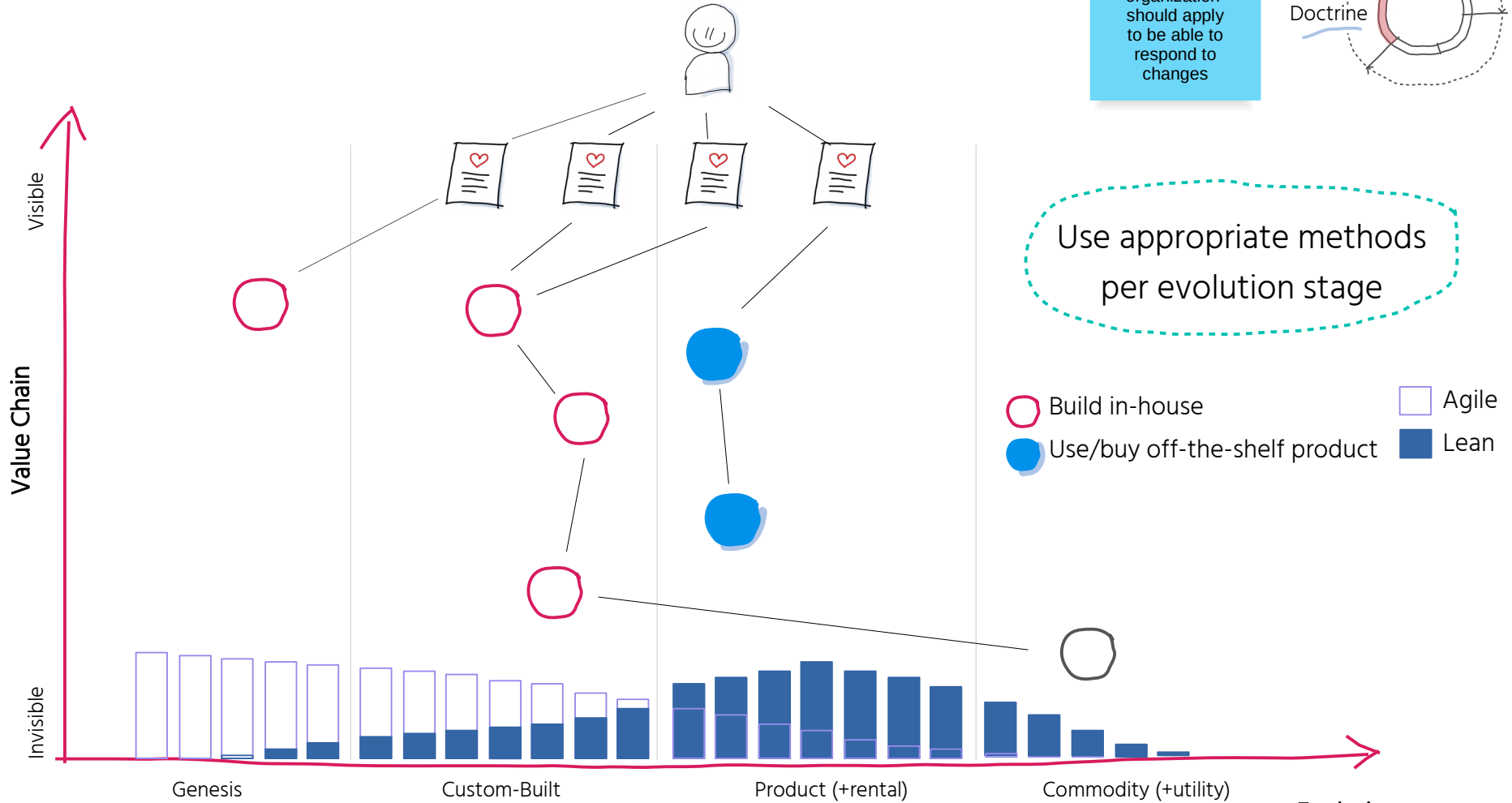
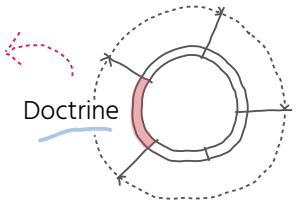
- 1 Identify users
- 2 Identify user needs
- 3 Identify components fulfilling user needs
- 4 Determine dependencies and position of the components in the value chain
- 5 Determine stage of evolution for every component

# Applying Doctrinal Principles



# Applying Doctrinal Principles

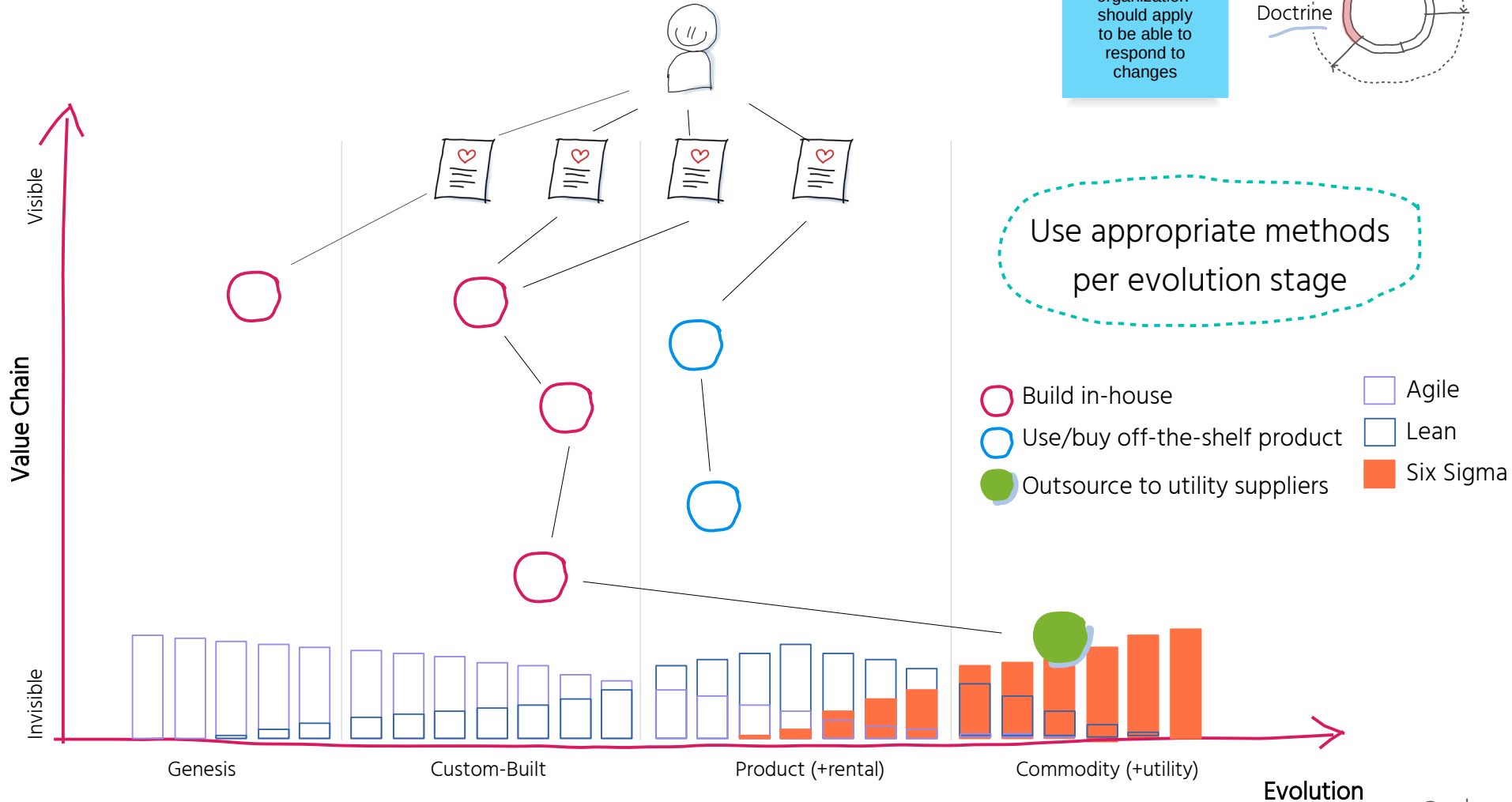
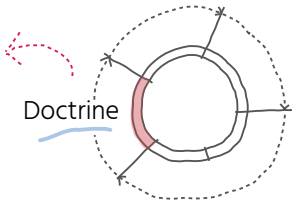
Universal principles an organization should apply to be able to respond to changes



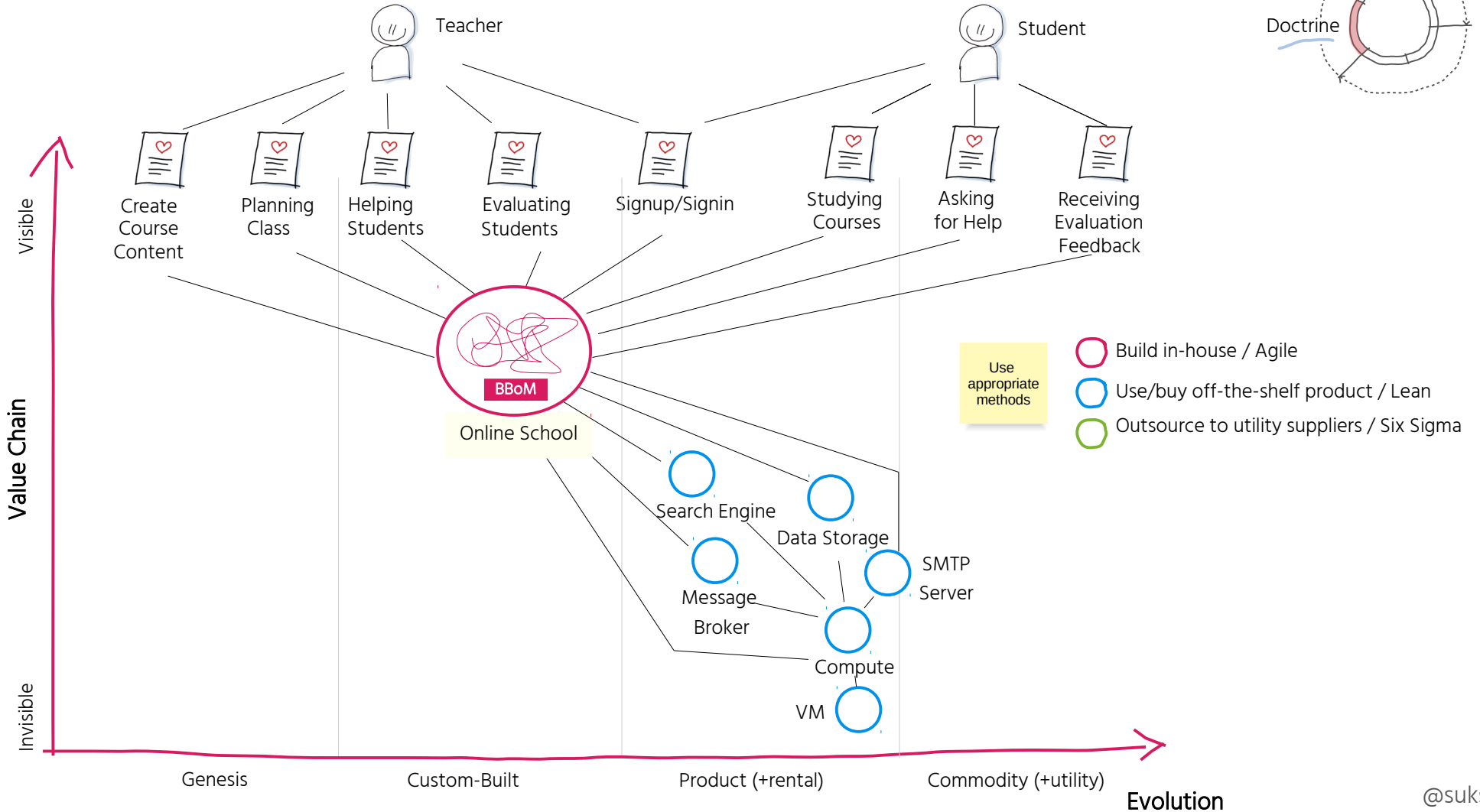
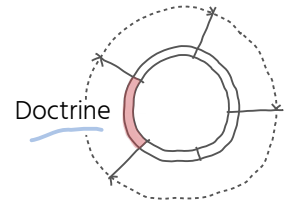
Evolution

# Applying Doctrinal Principles

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

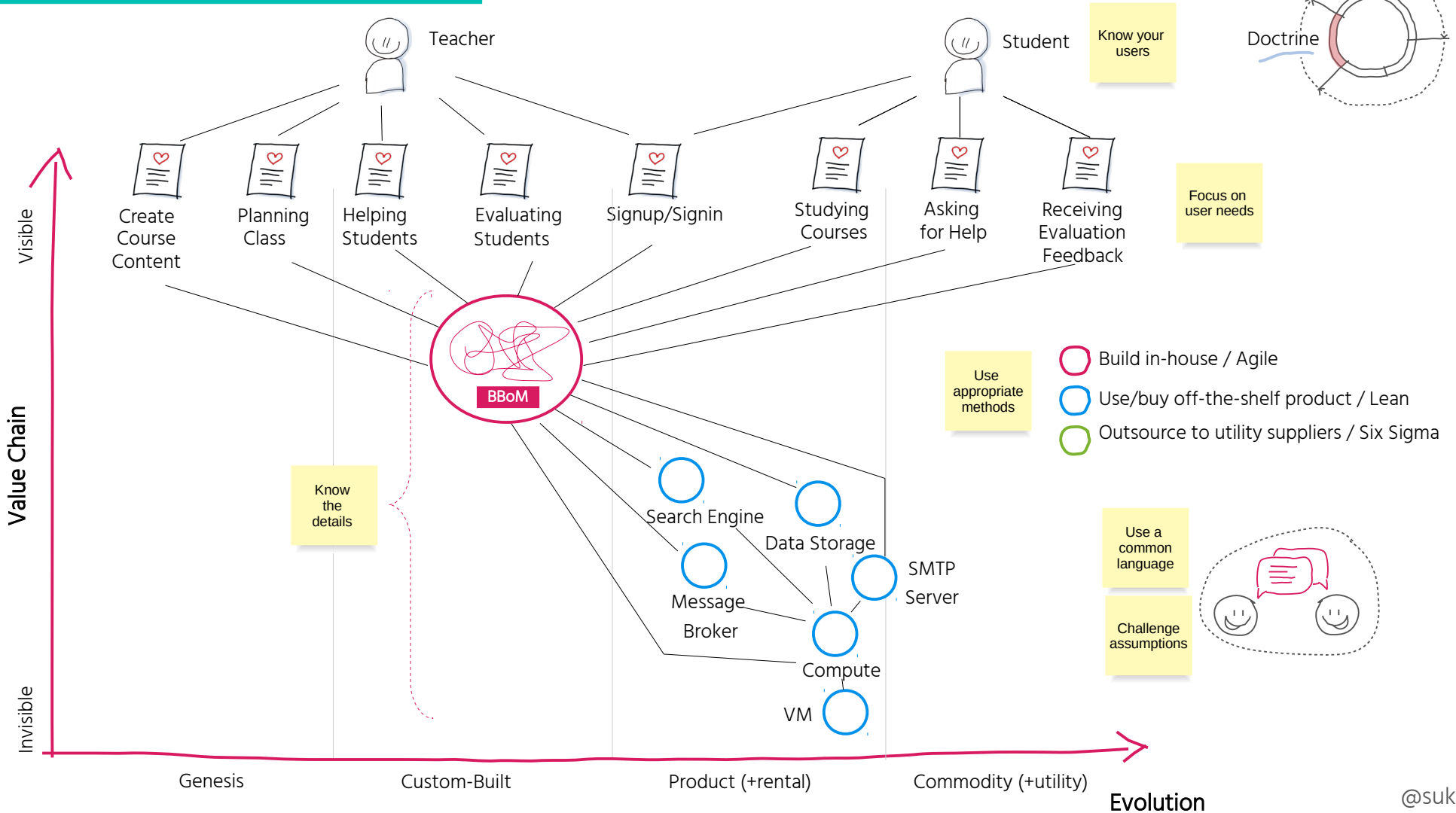


# Applying Doctrinal Principles

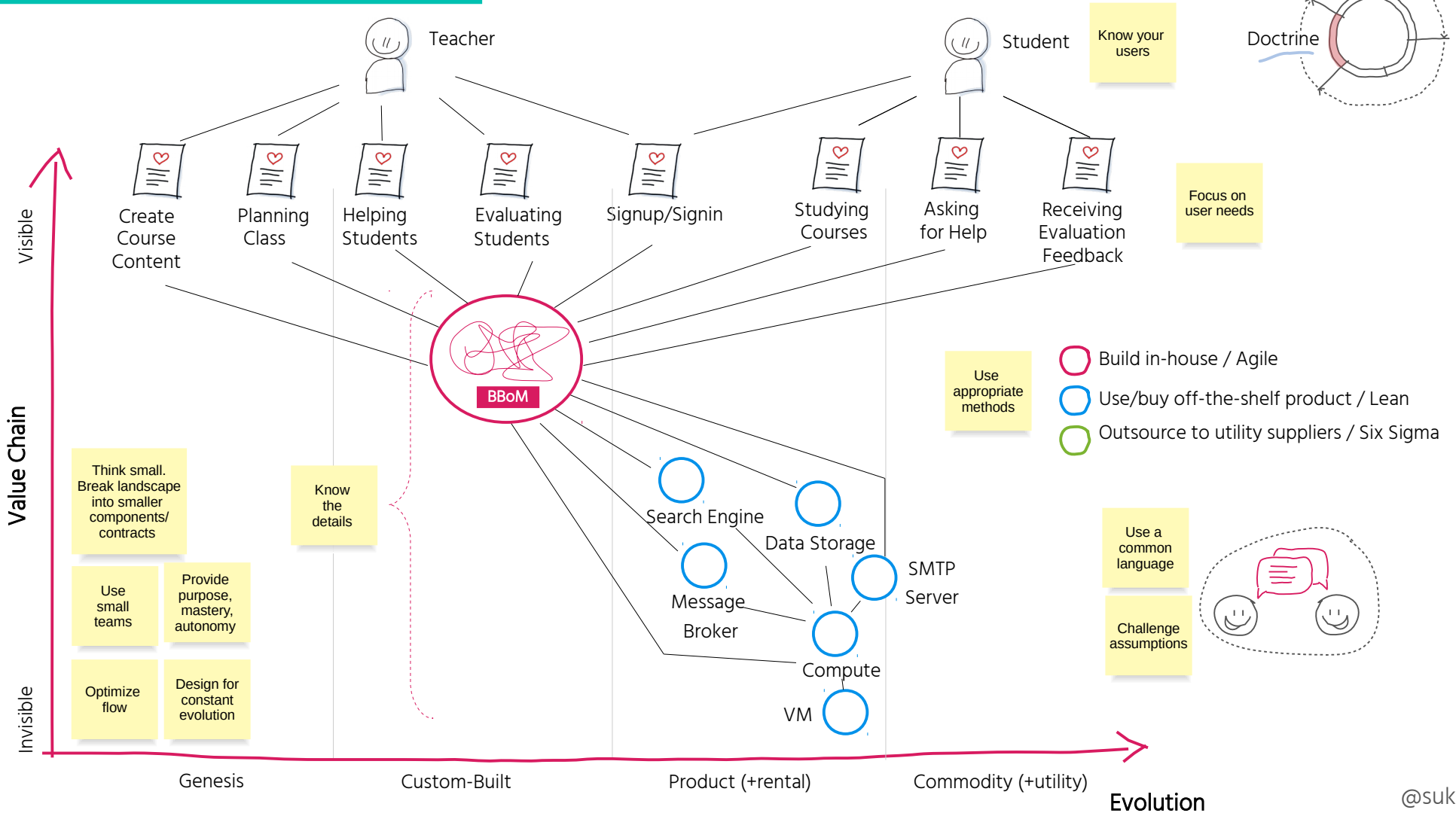




# The Doctrinal Principles (extract)



# The Doctrinal Principles (extract)

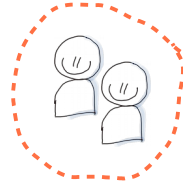
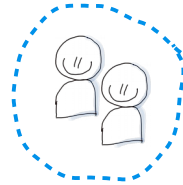
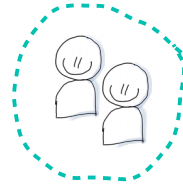


# To optimize for flow of change from a team perspective requires ...

cross-functional,  
autonomous teams

no handover between teams

restricting high, on-going  
communication bandwidth  
between teams

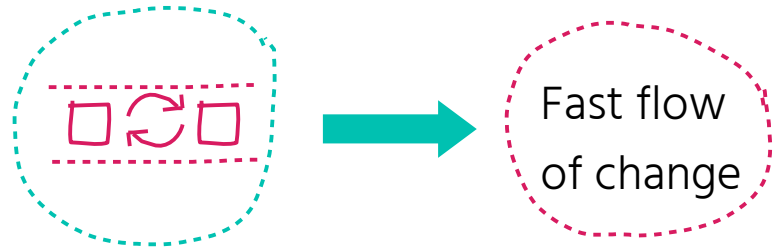


small, long-lived teams

minimizing cognitive load

team ownership

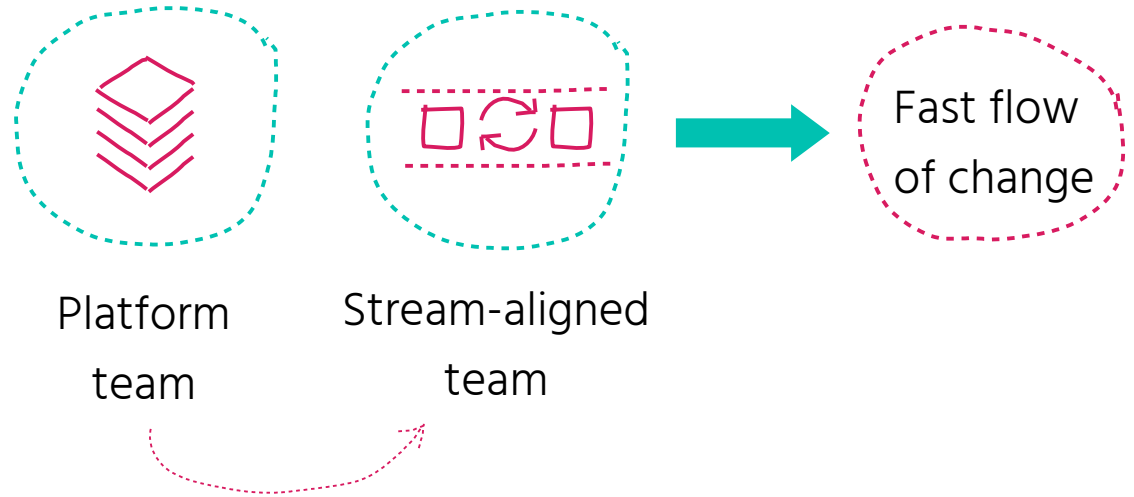
# Four Team Types of Team Topologies



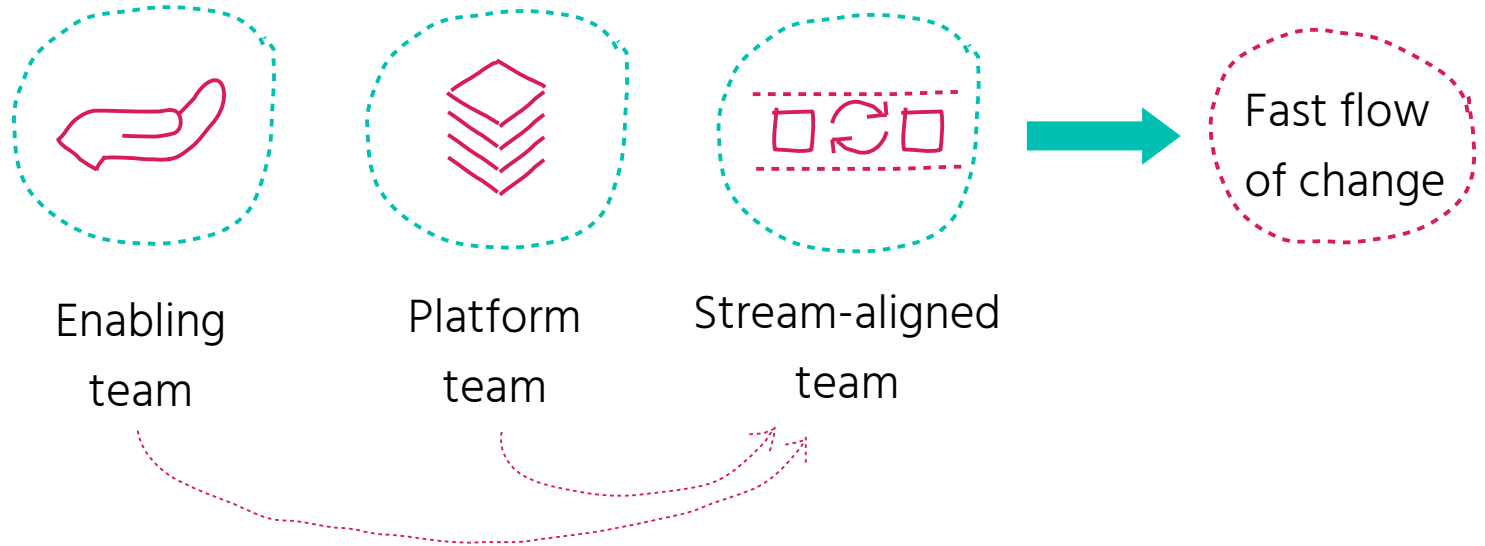
Stream-aligned  
team

Fast flow  
of change

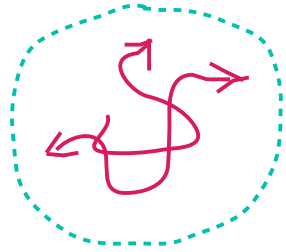
# Four Team Types of Team Topologies



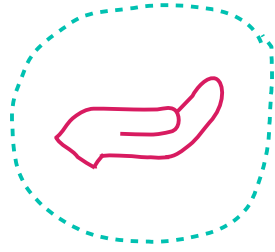
# Four Team Types of Team Topologies



# Four Team Types of Team Topologies



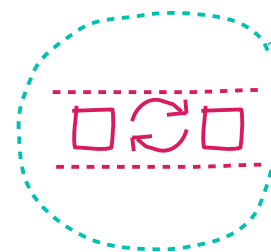
Complicated  
subsystem team



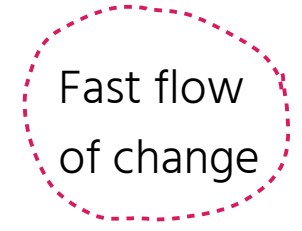
Enabling  
team



Platform  
team



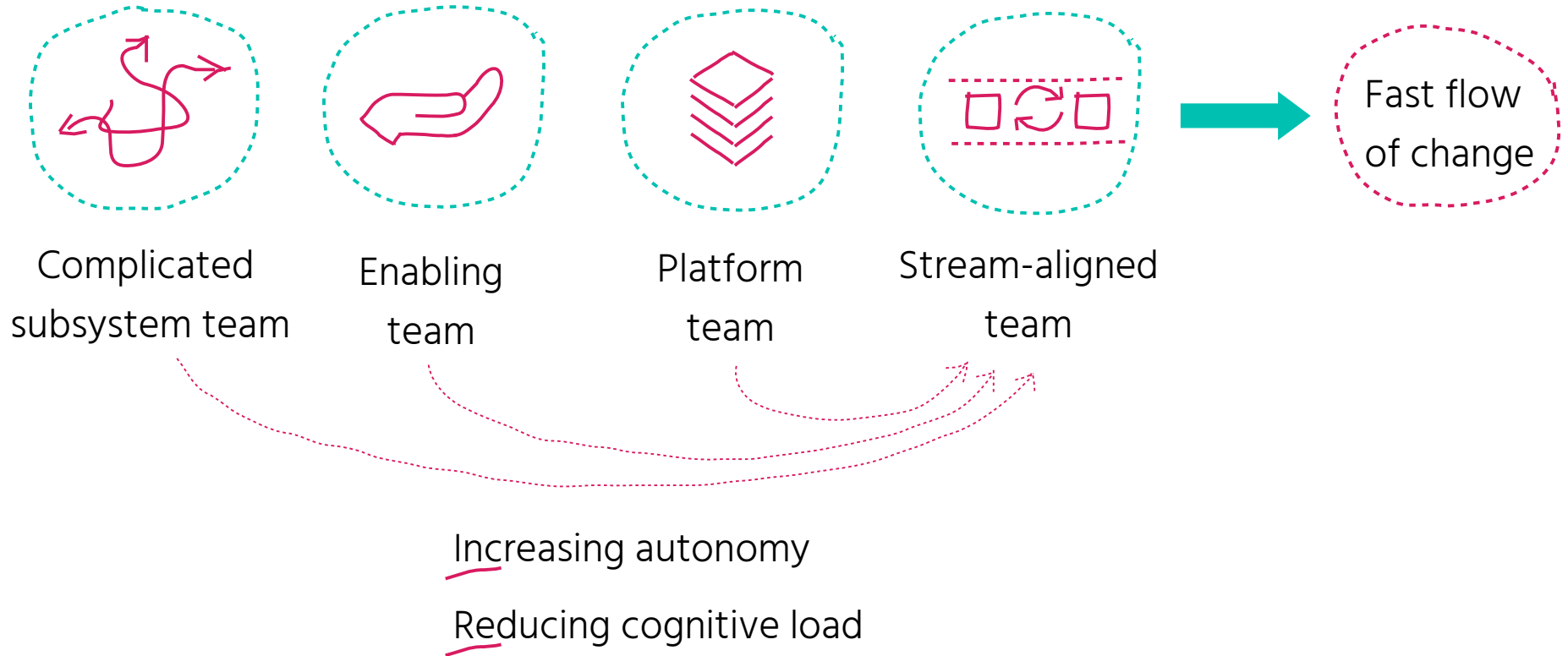
Stream-aligned  
team



Fast flow  
of change

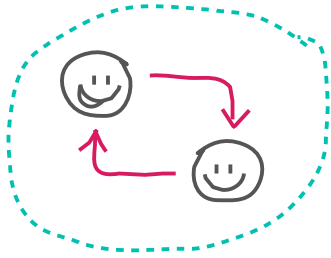


# Four Team Types of Team Topologies





# Three Interaction Modes

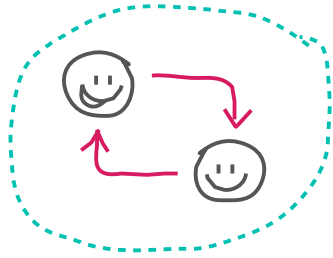


Collaboration



Rapid discovery

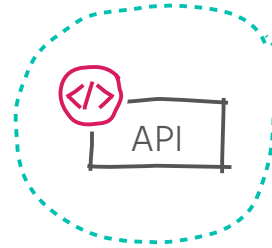
# Three Interaction Modes



Collaboration



Rapid discovery

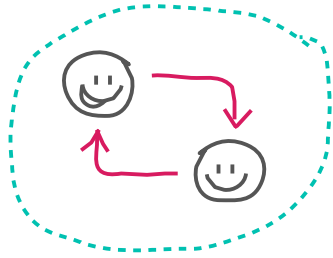


X-as-a-Service



Predictable  
delivery

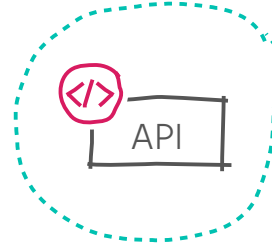
# Three Interaction Modes



Collaboration



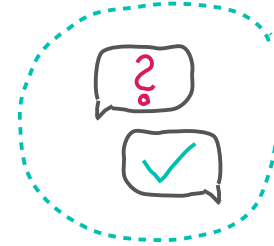
Rapid discovery



X-as-a-Service



Predictable  
delivery



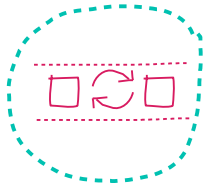
Facilitating



Active help

# Team Topologies

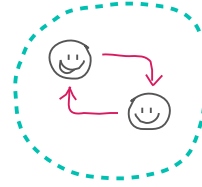
Stream-aligned team



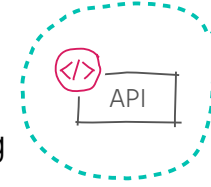
Platform team



Collaboration



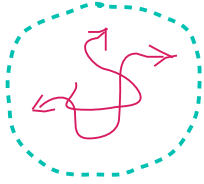
X-as-a-Service



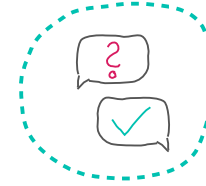
Enabling team



Complicated subsystem team

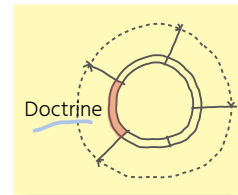


Facilitating

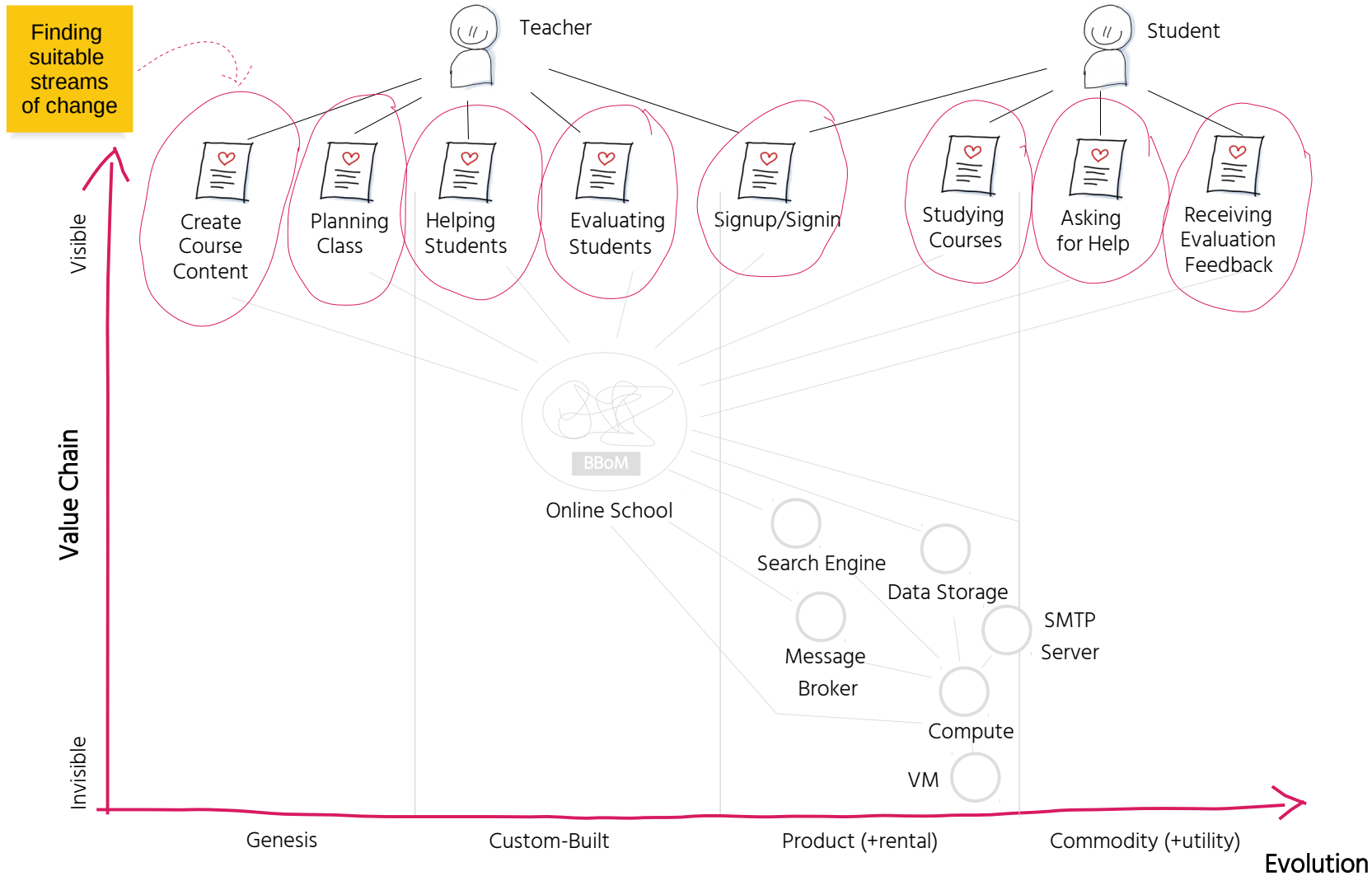


Promotes organizational effectiveness

Helps to apply Wardley's Doctrinal Principles



# Architecture For Flow



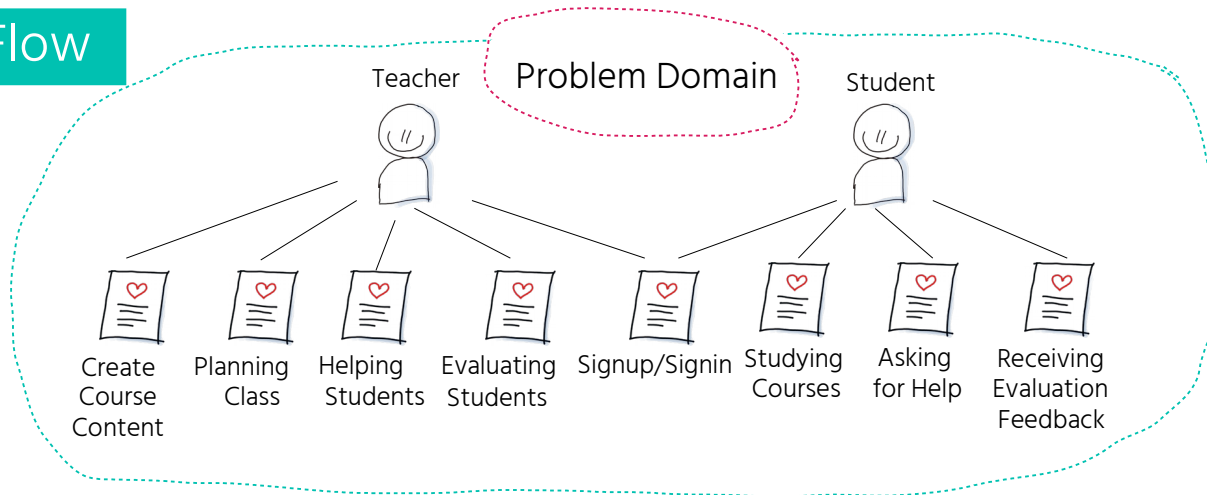
# Architecture For Flow

Finding suitable streams of change

Visible

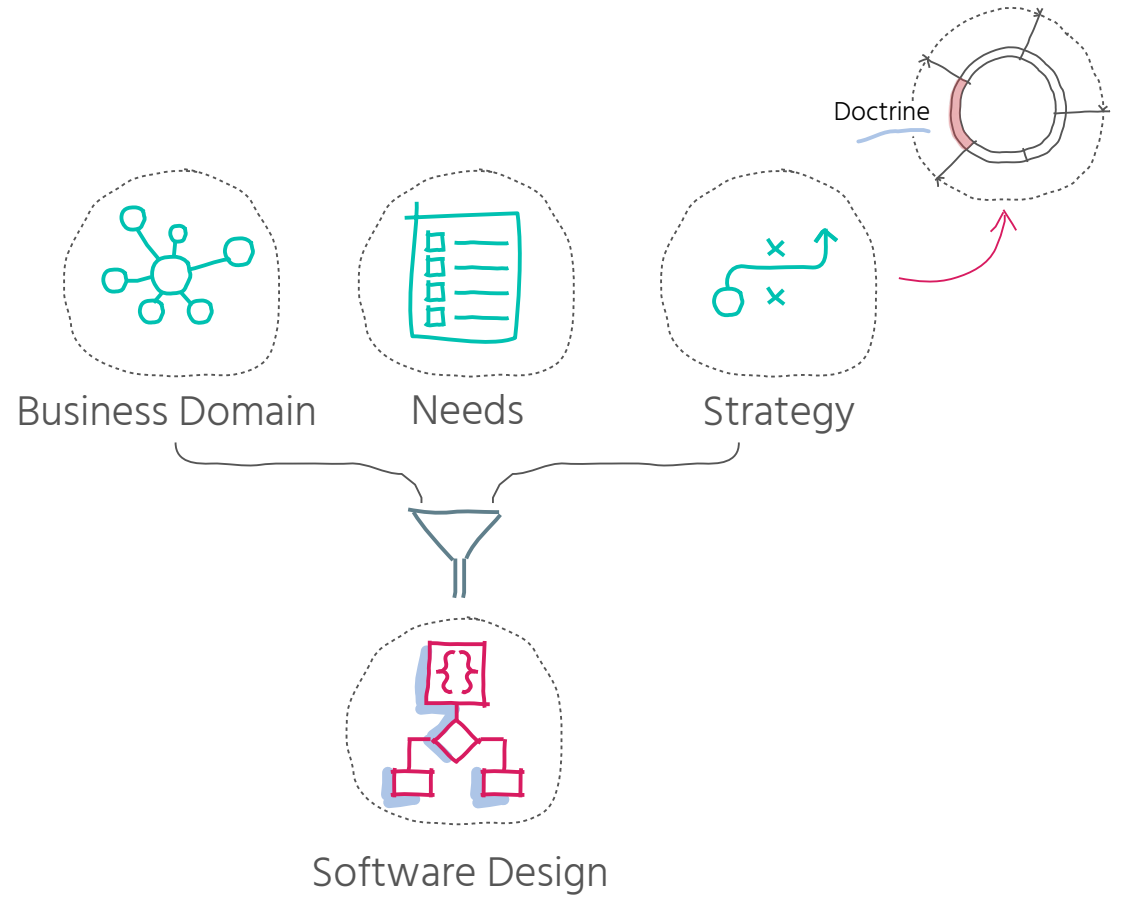
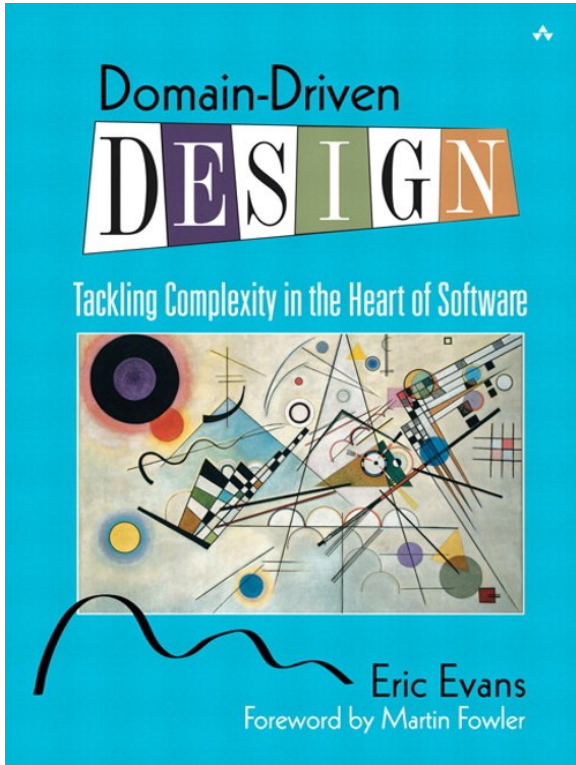
Value Chain

Invisible



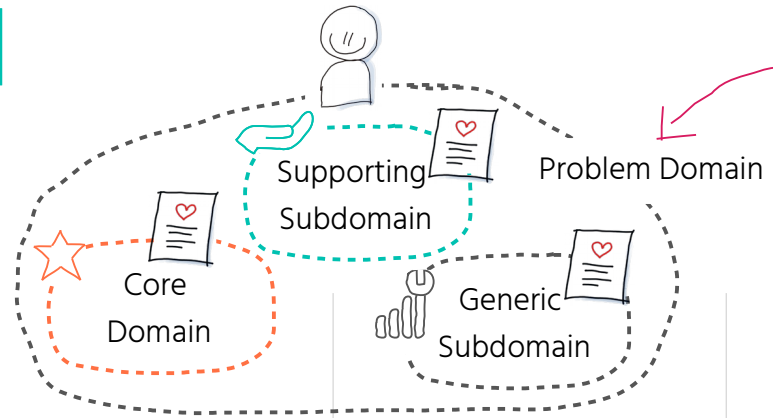
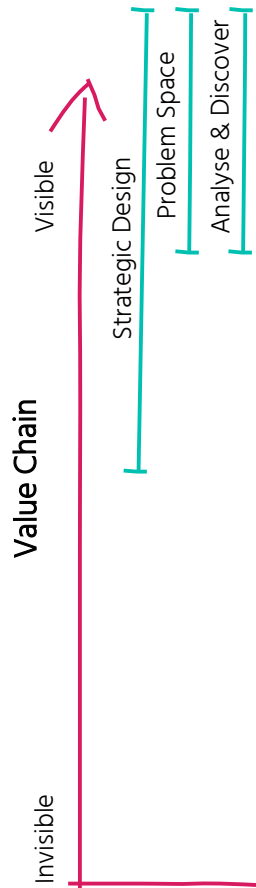
Understanding the problem domain and partitioning it into subdomains

# Domain-Driven Design (DDD)



# DDD & Wardley Map

Strategic Design (Problem Space)



Distilling the problem domain & discovering the core domain

Genesis

Custom-Built

Product (+rental)

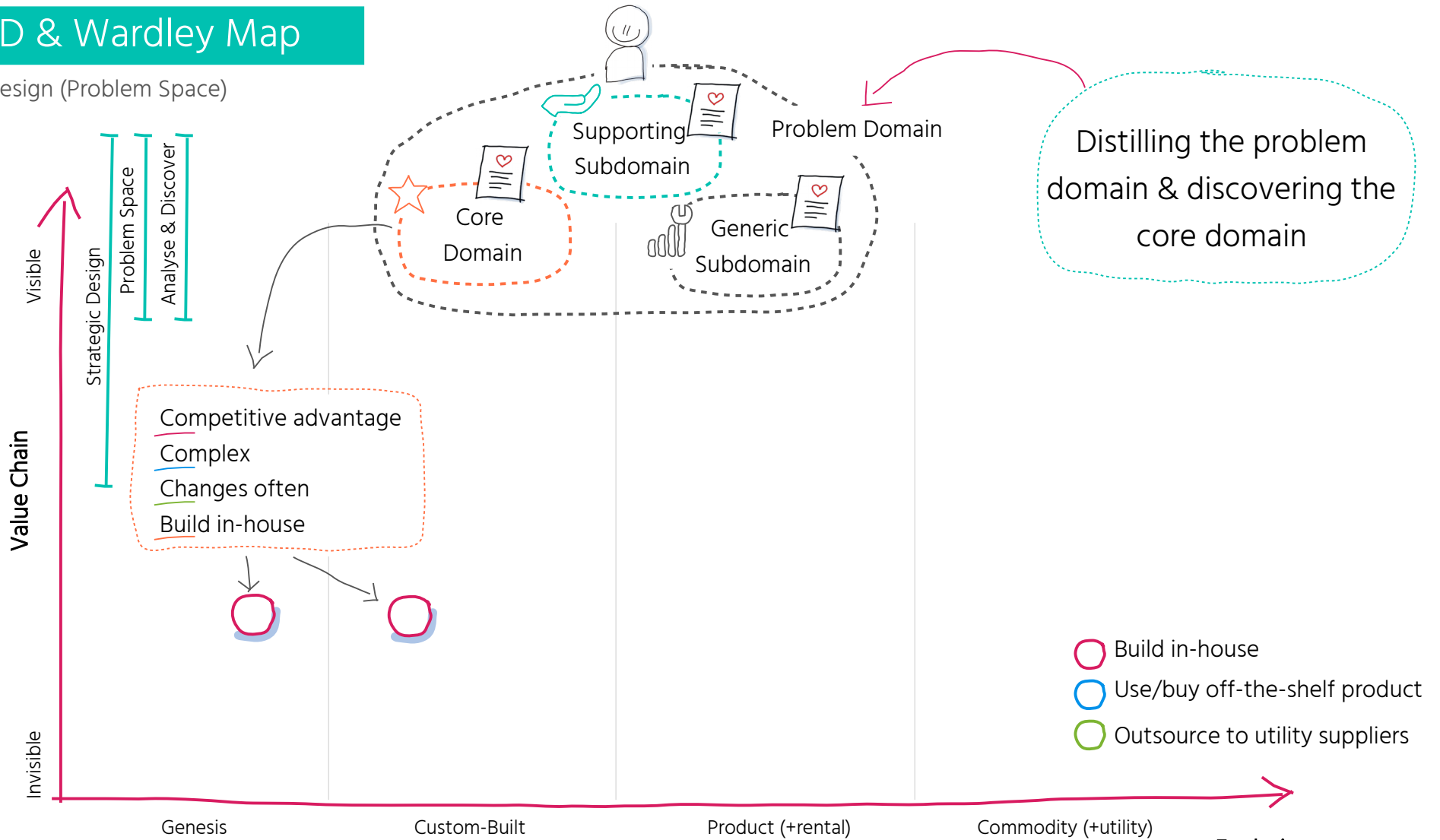
Commodity (+utility)

Evolution



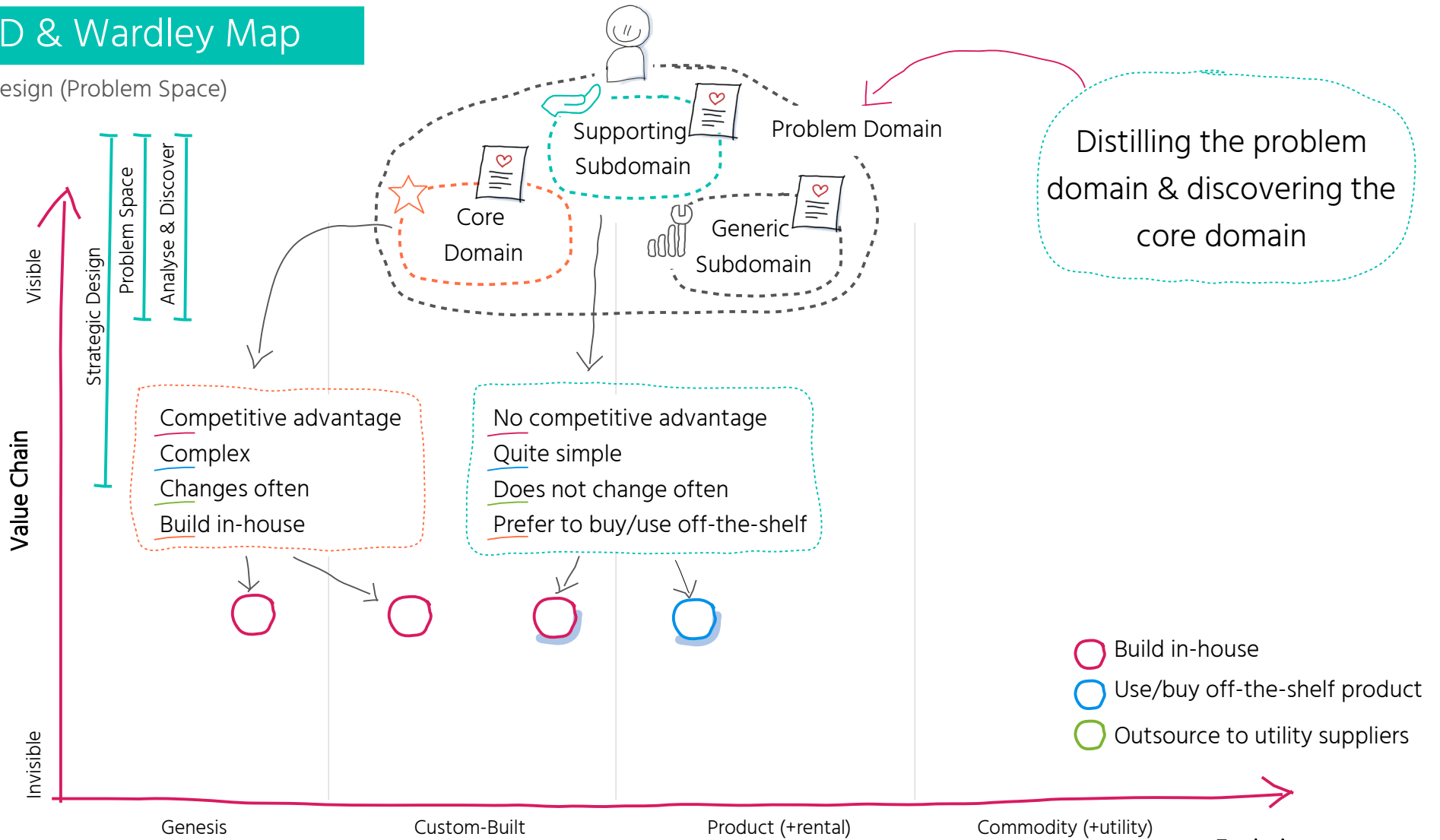
# DDD & Wardley Map

Strategic Design (Problem Space)



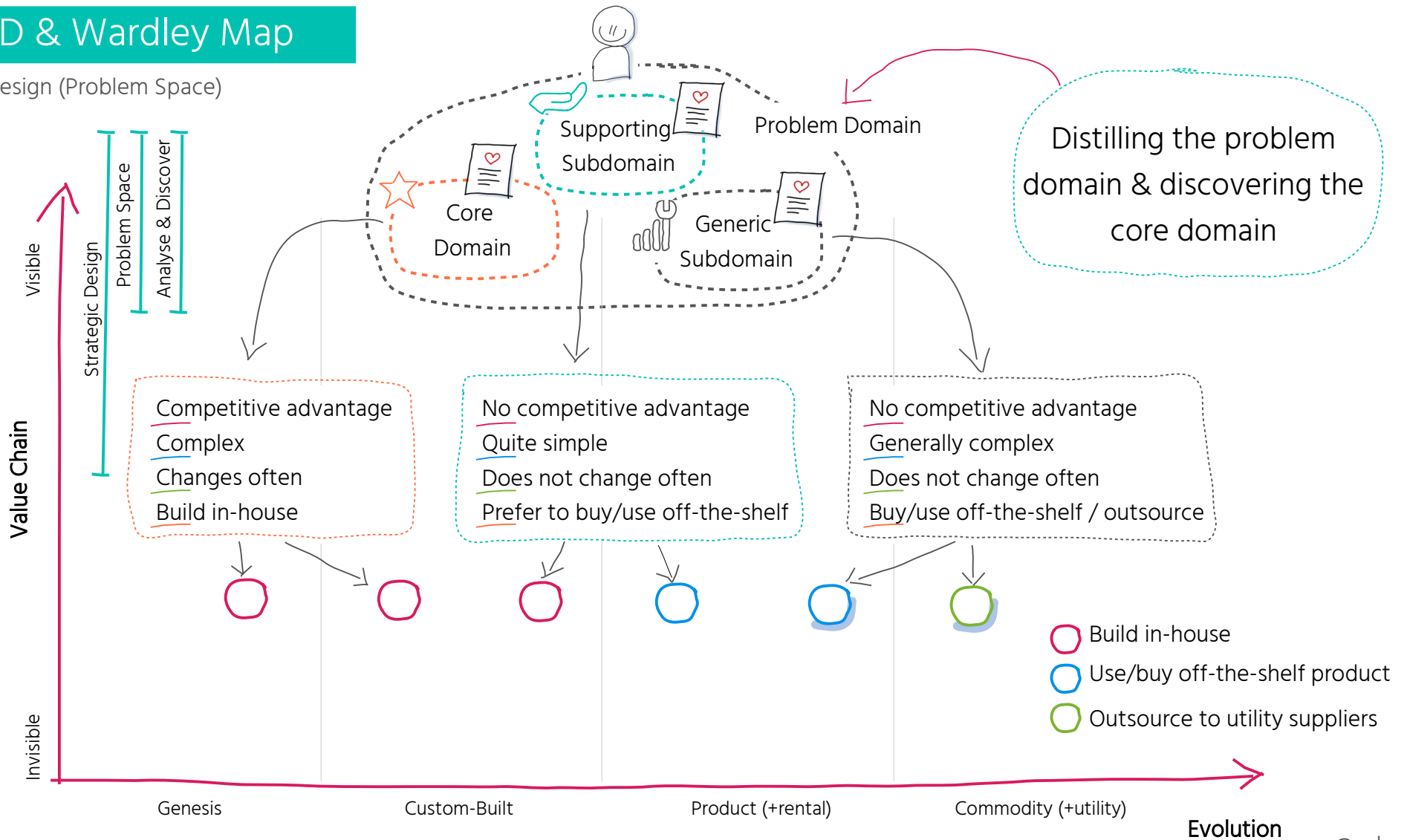
# DDD & Wardley Map

Strategic Design (Problem Space)



# DDD & Wardley Map

Strategic Design (Problem Space)



# Architecture For Flow

Finding suitable streams of change

Visible

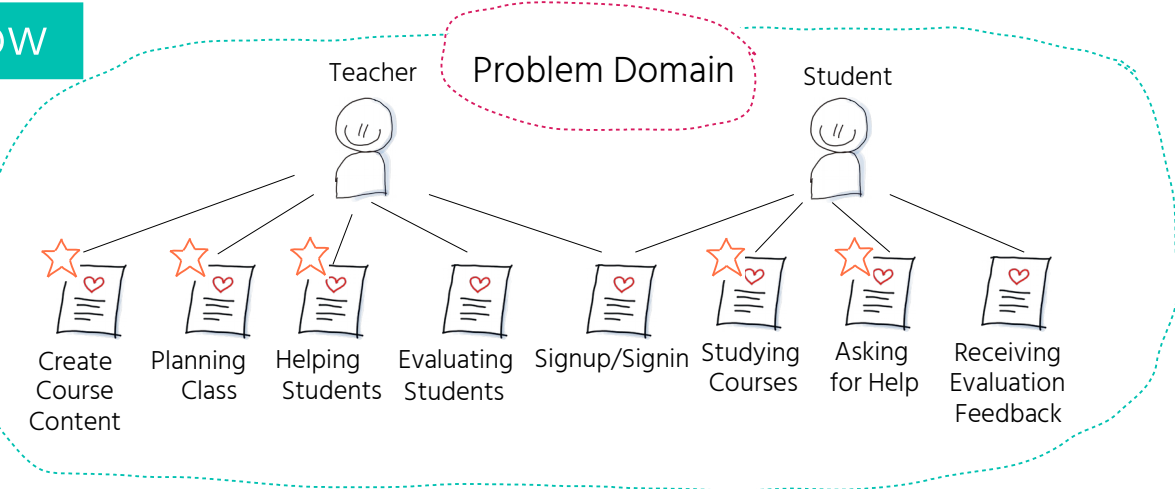
Value Chain

Invisible

Strategic Design


Problem Space

Analyse & Discover



Partitioning the Problem Domain into Subdomains

Discovering the Core

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

# Architecture For Flow

Finding suitable streams of change

Visible

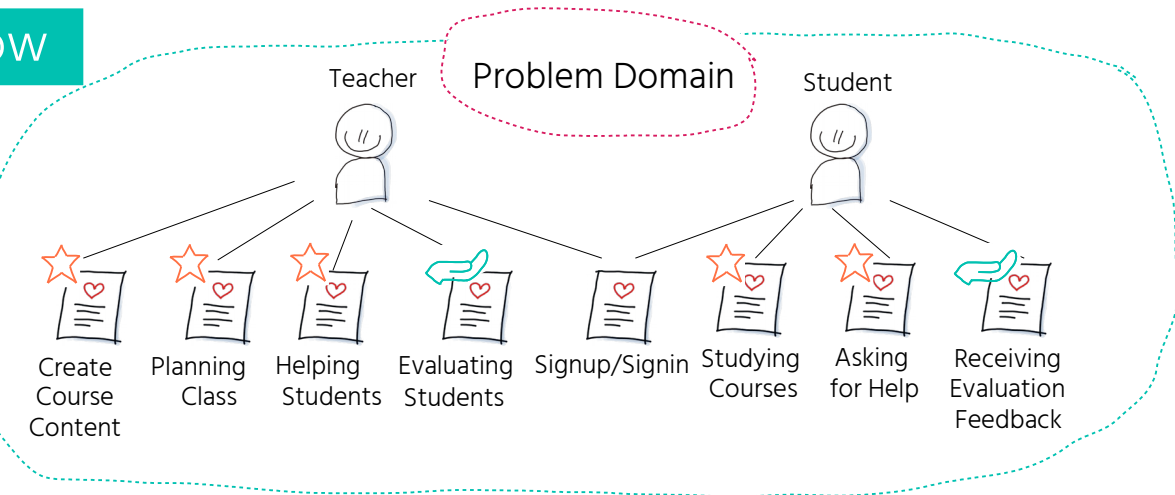
Value Chain

Invisible

Strategic Design



Problem Space

Analyse & Discover



Partitioning the Problem Domain into Subdomains

Discovering the Core

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

# Architecture For Flow

Finding suitable streams of change

Visible

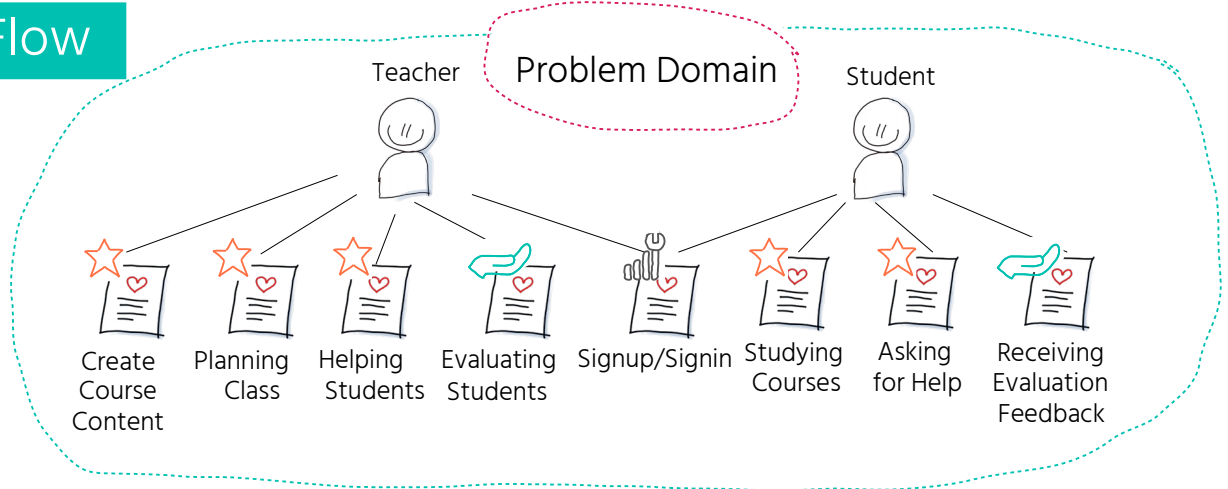
Value Chain

Invisible

Strategic Design




Problem Space

Analyse & Discover

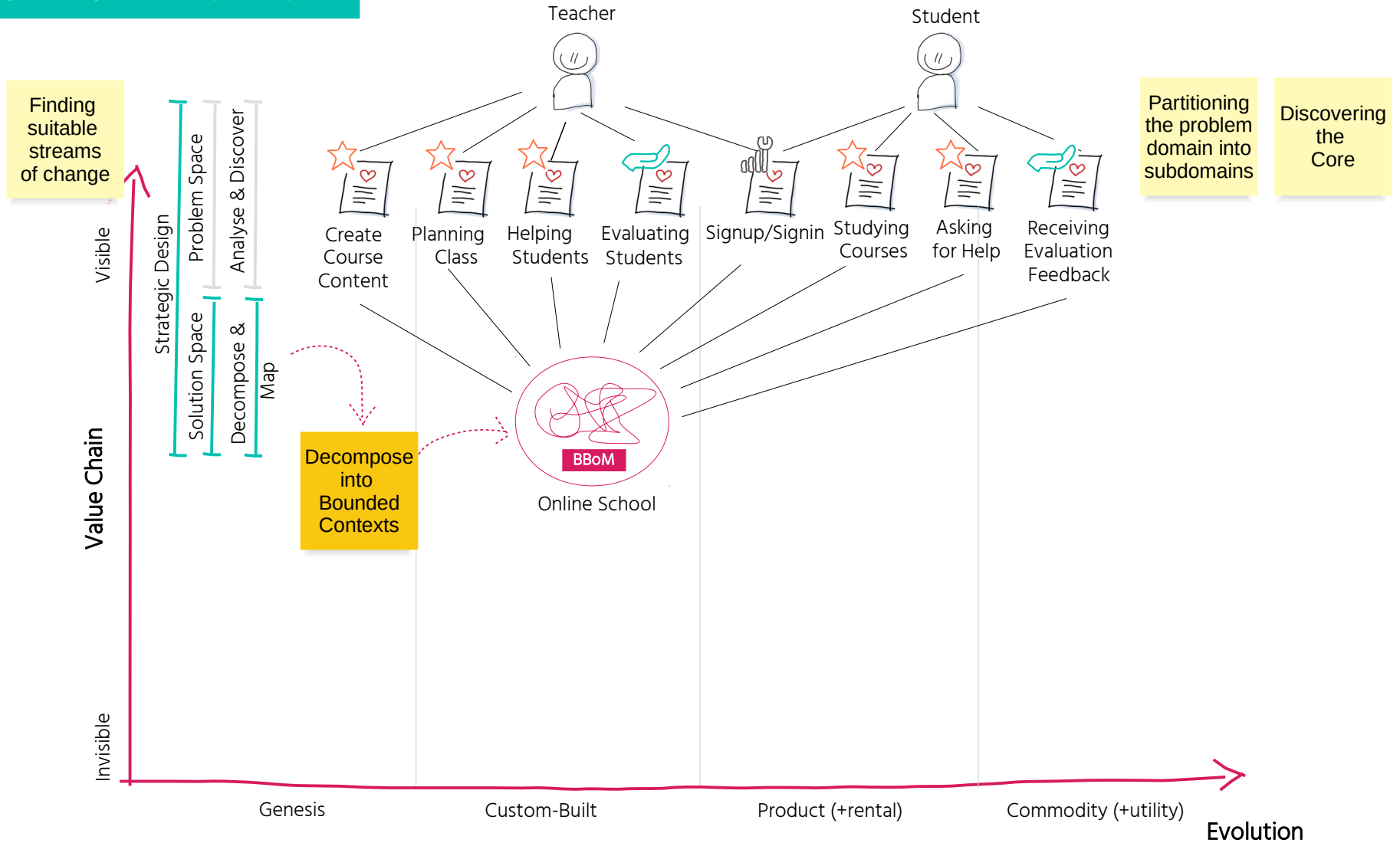


Partitioning the Problem Domain into Subdomains

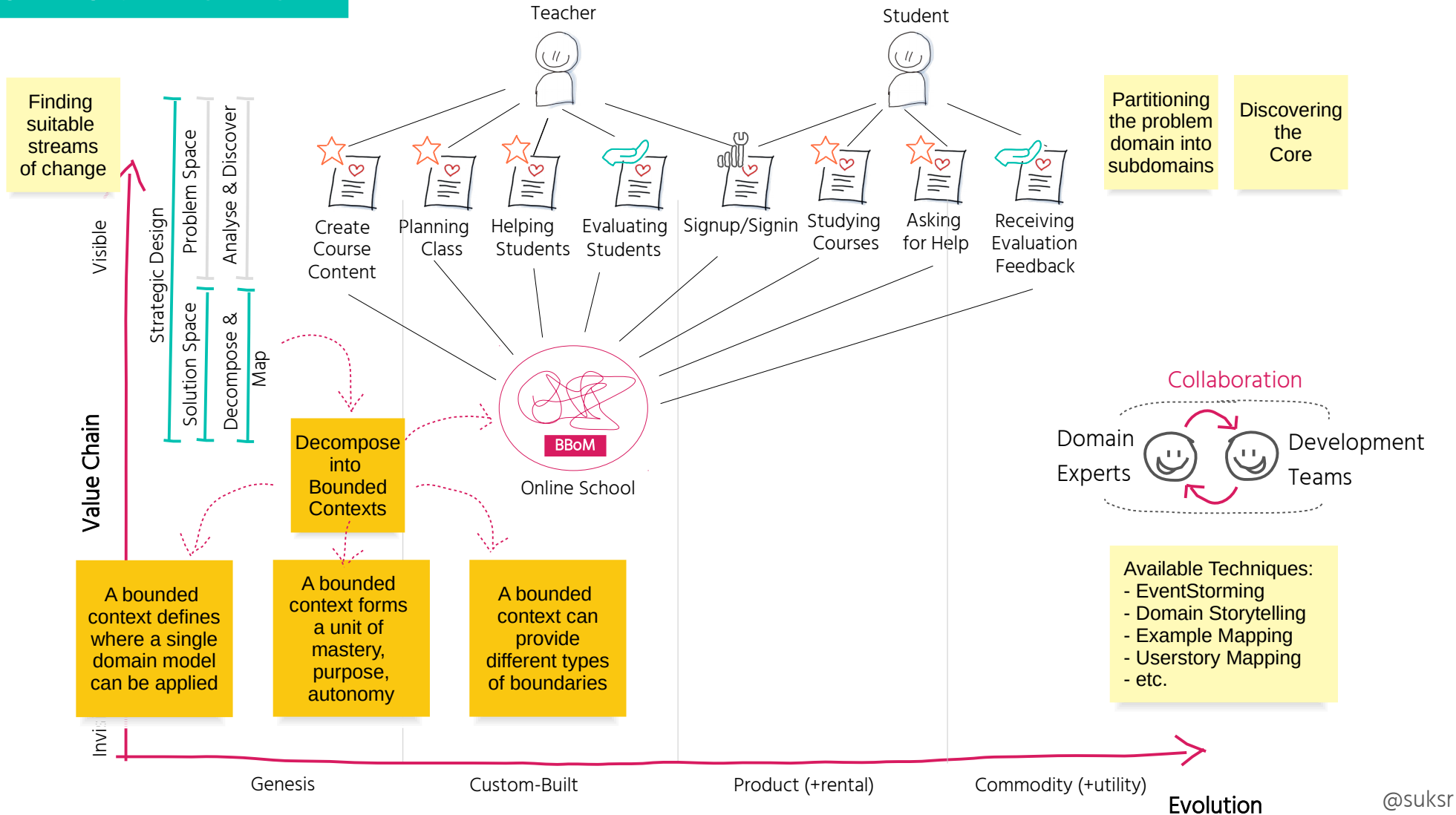
Discovering the Core

	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

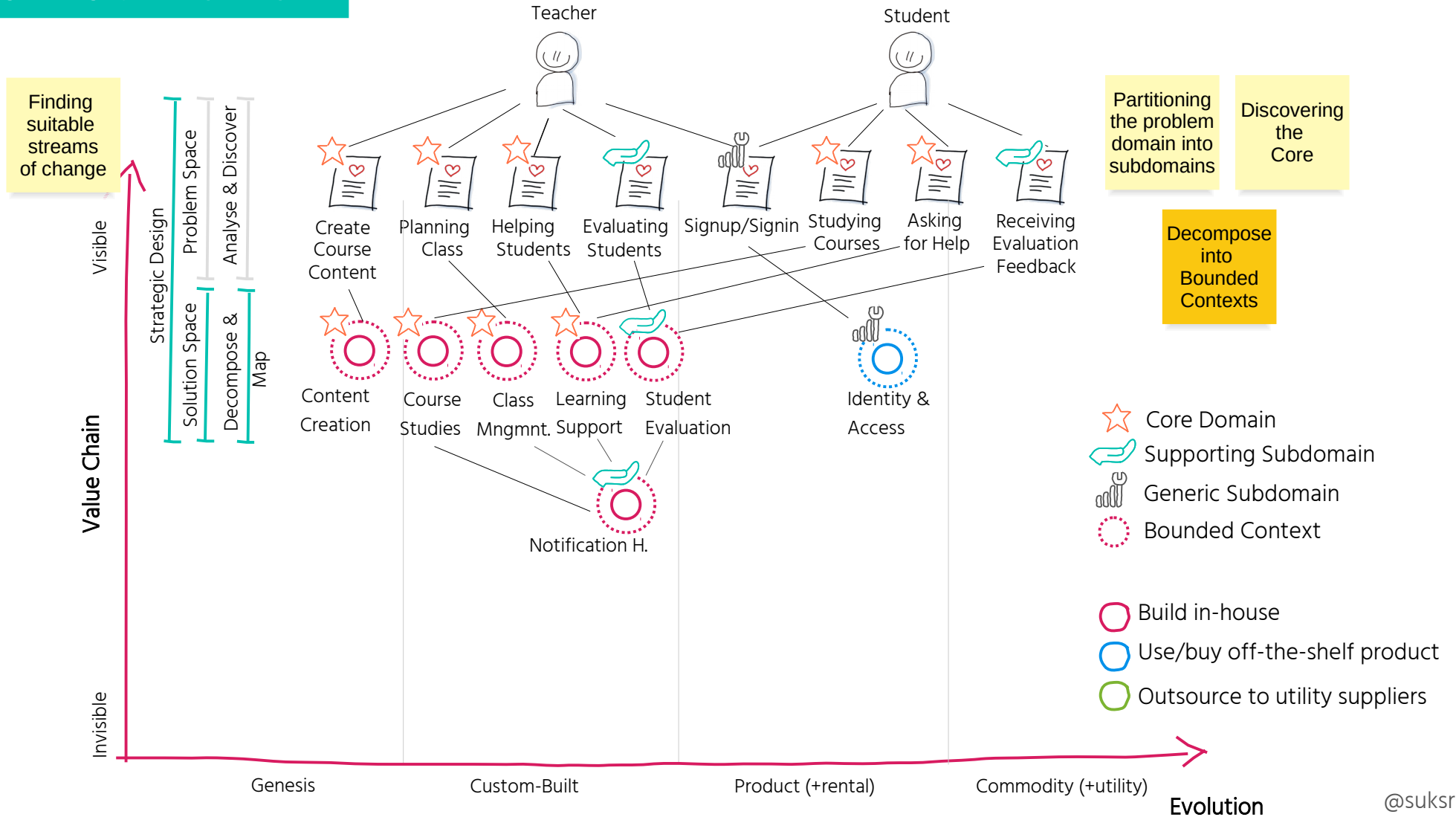


# Architecture For Flow

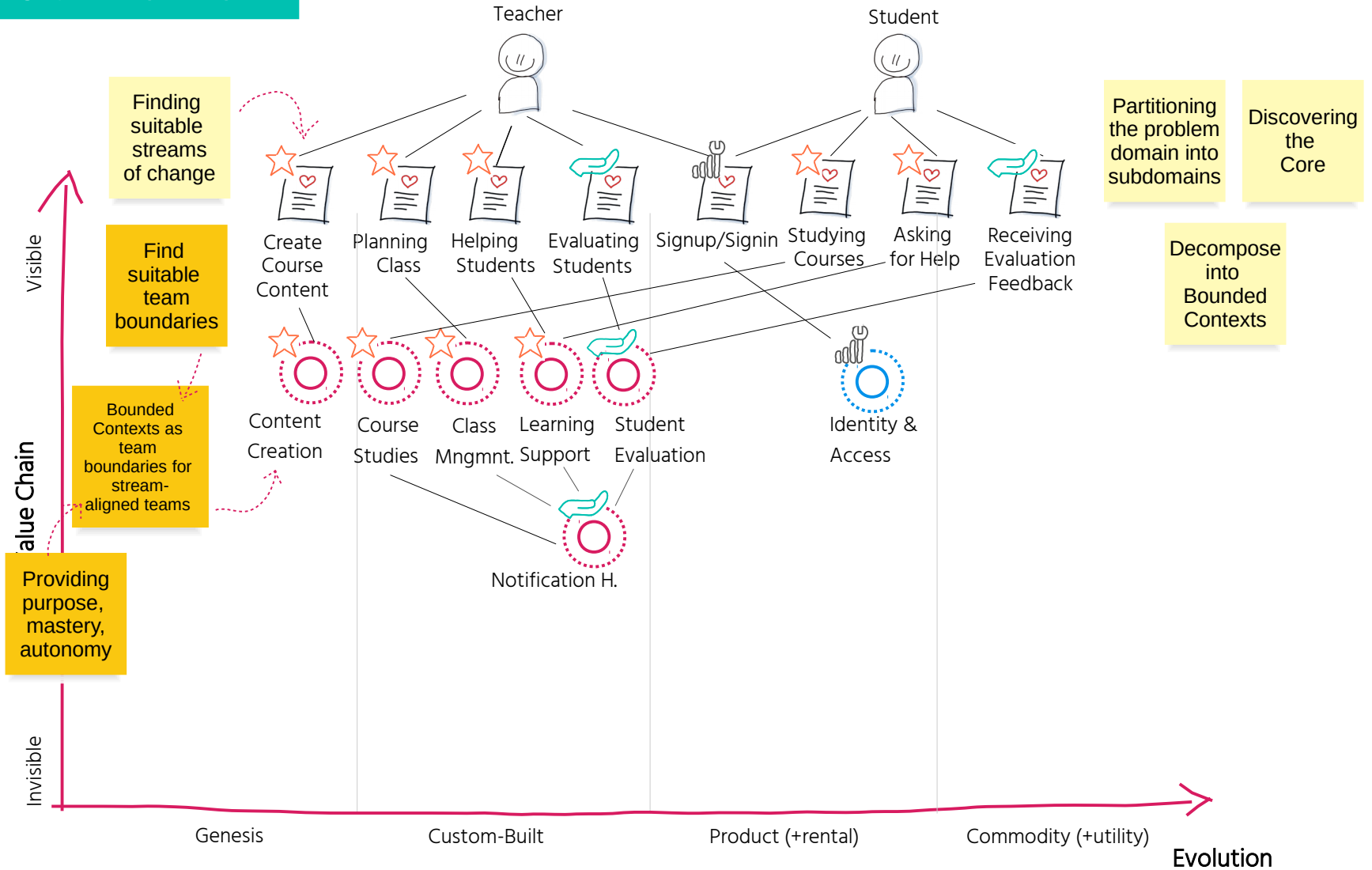




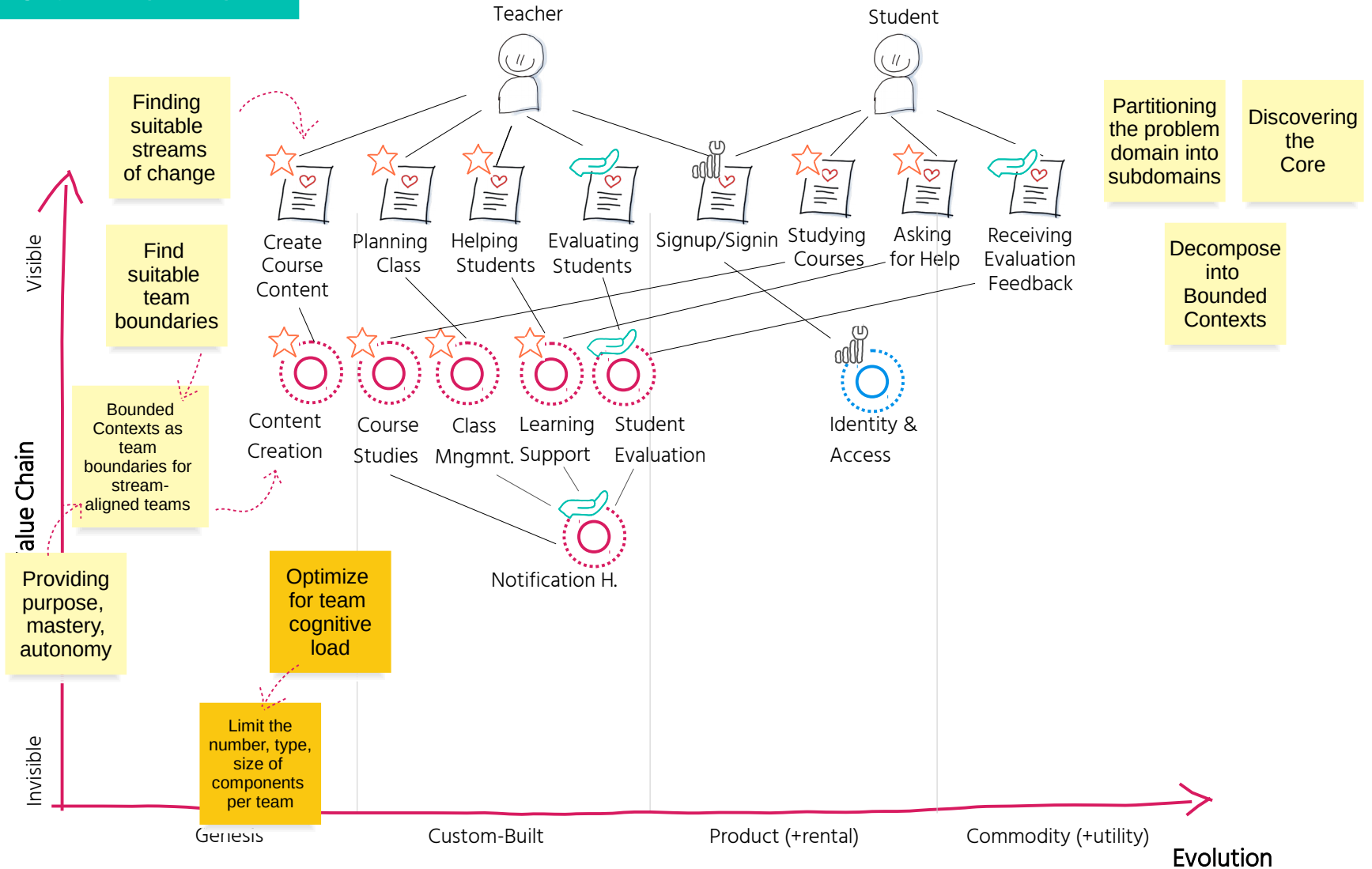
# Architecture For Flow



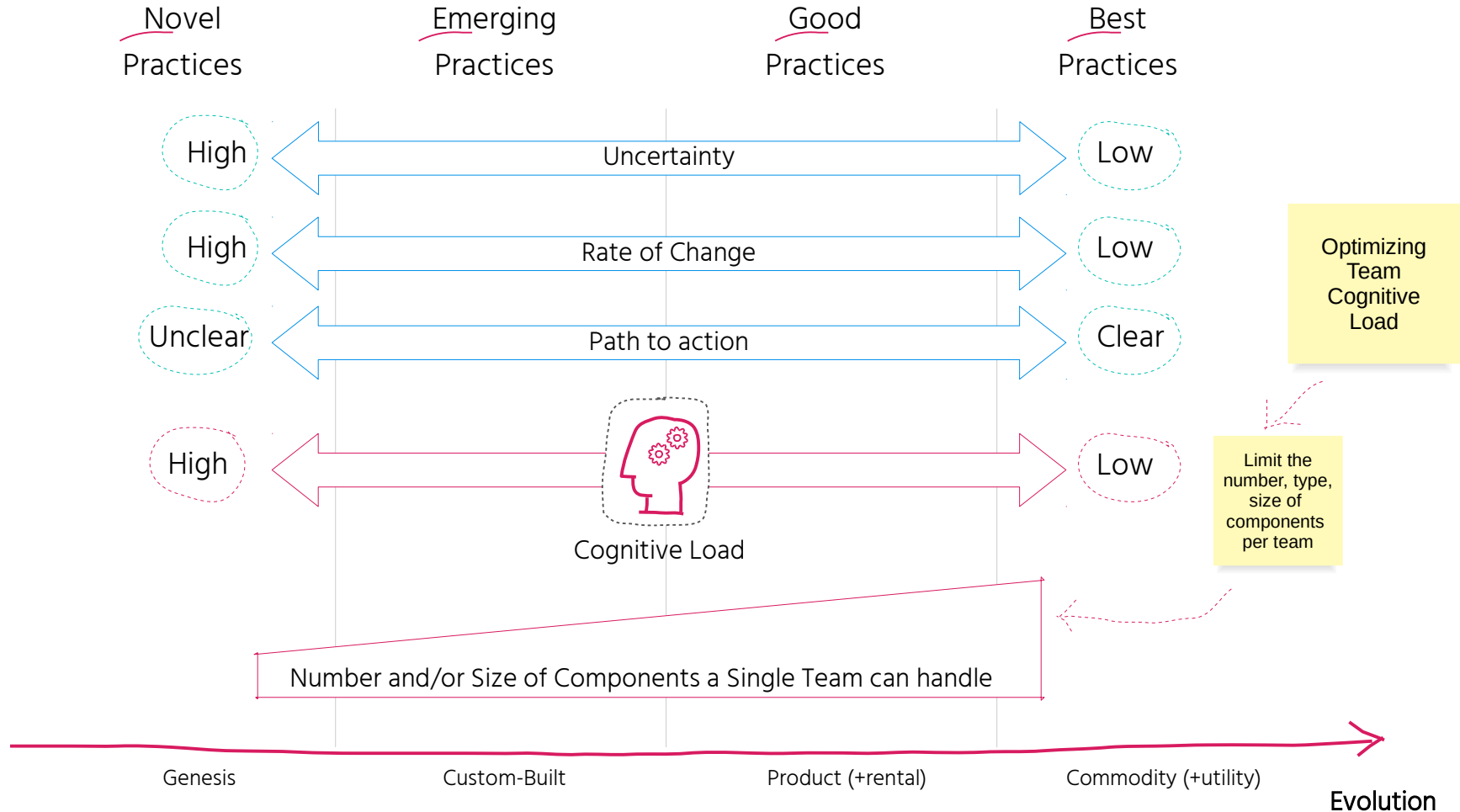
# Architecture For Flow



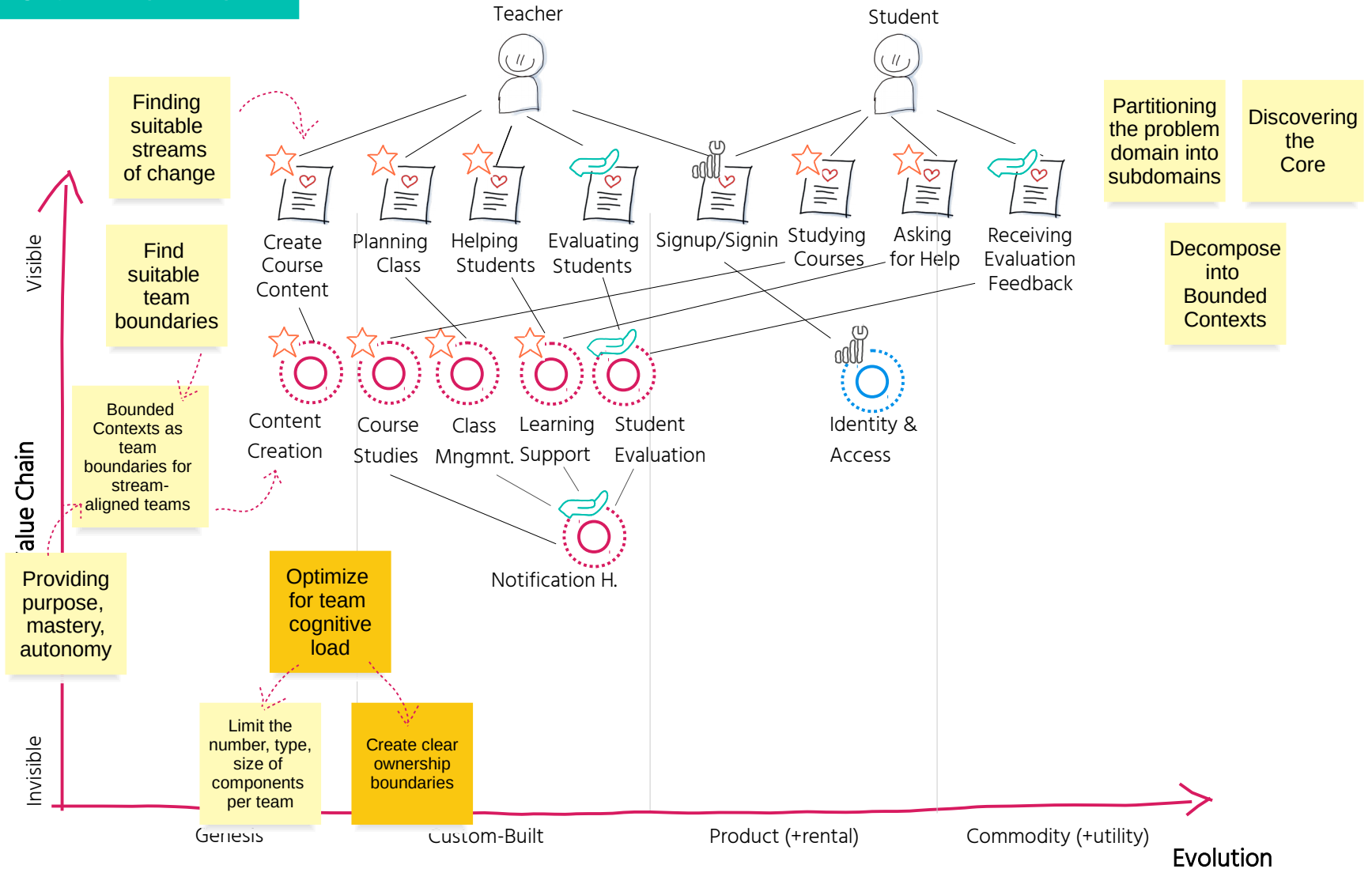
# Architecture For Flow

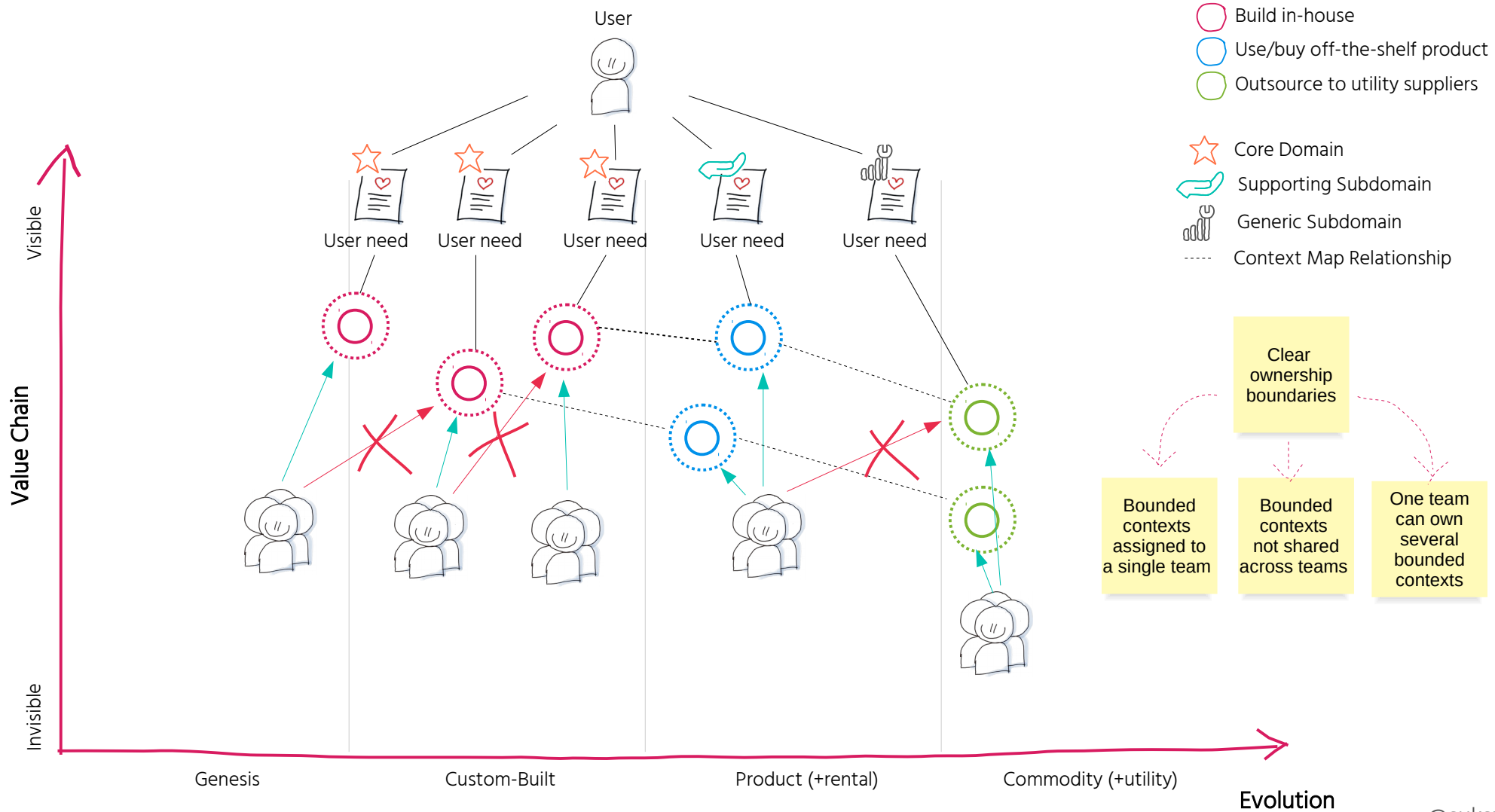


# Architecture For Flow

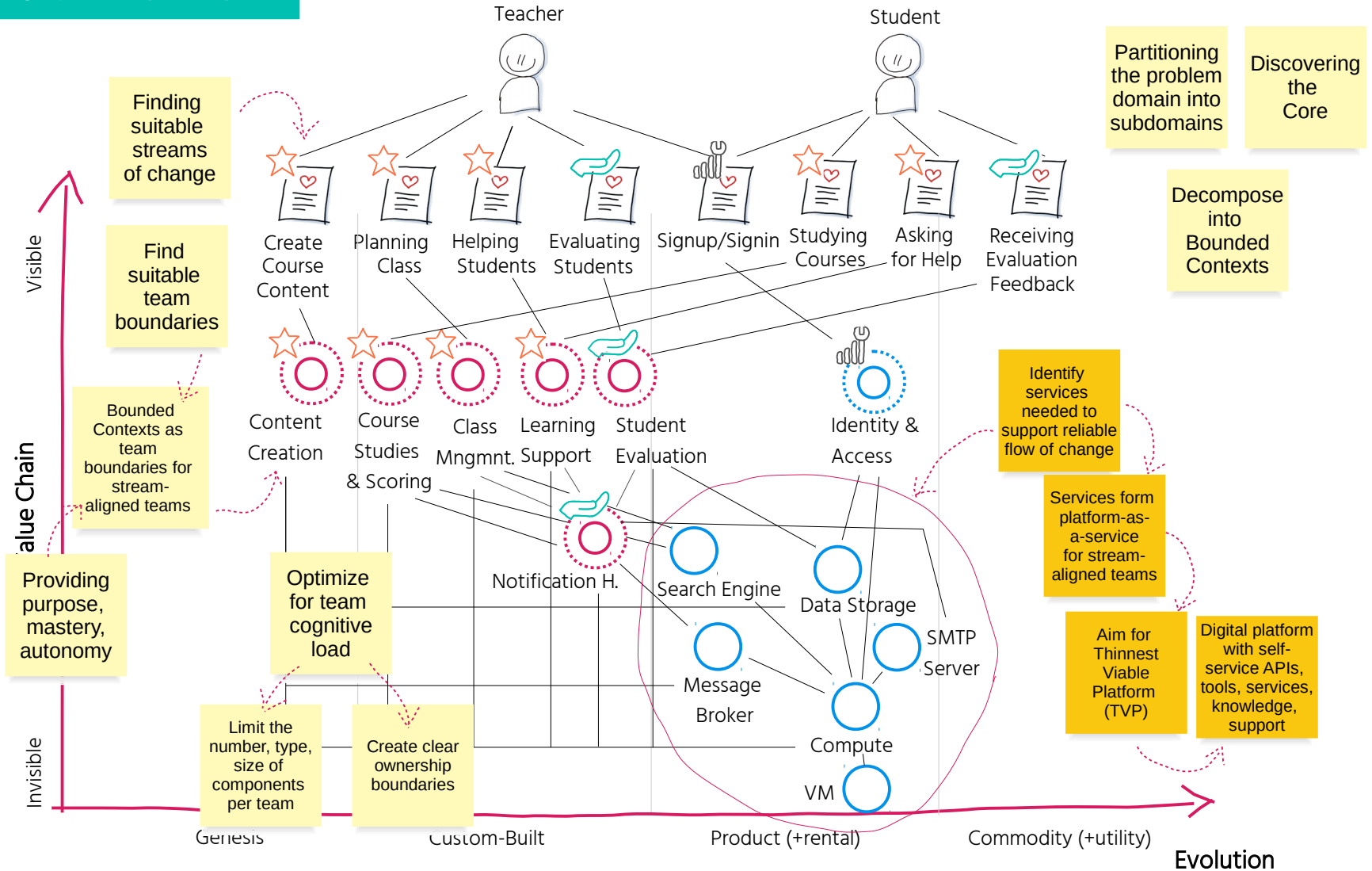


# Architecture For Flow

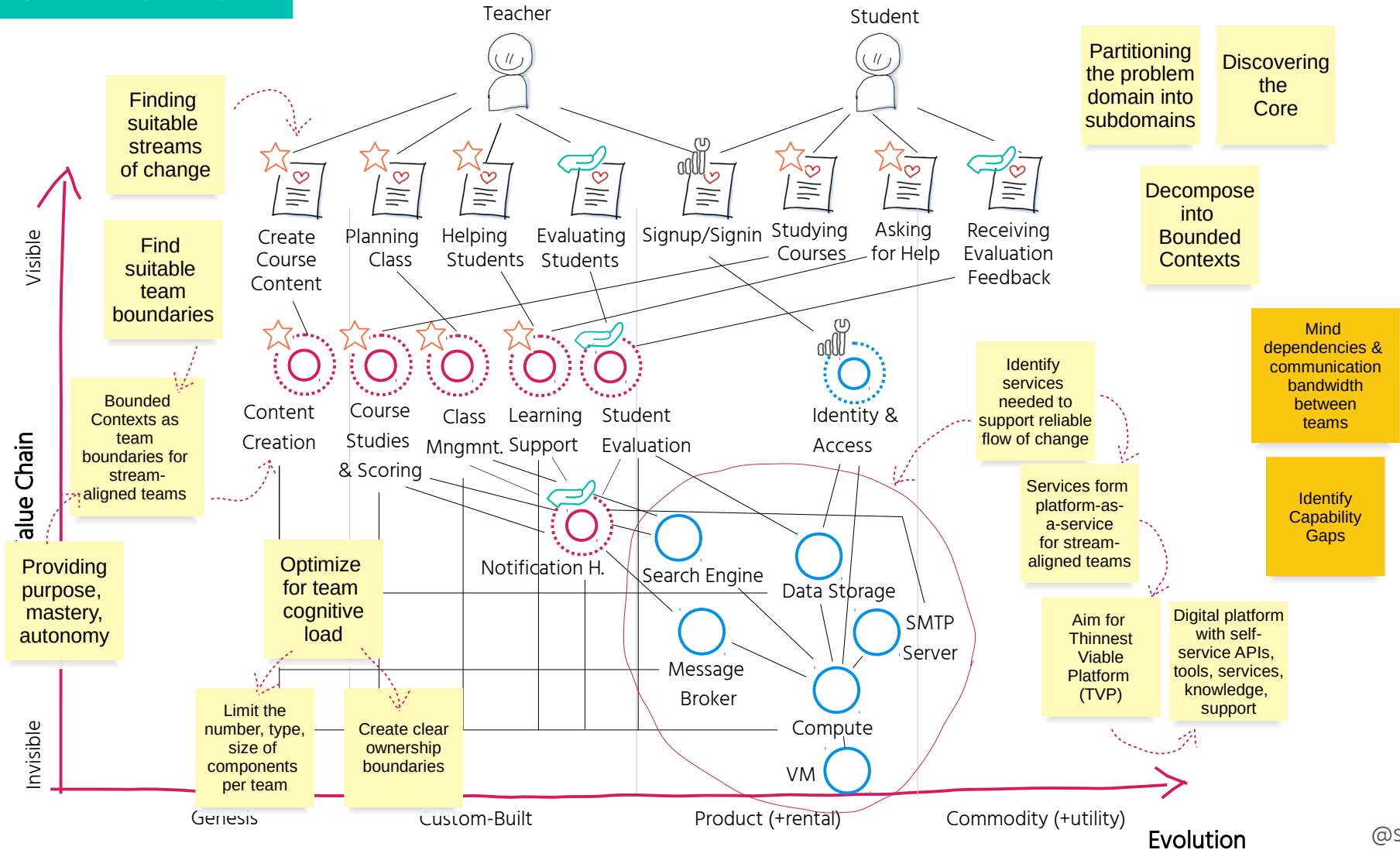




# Architecture For Flow

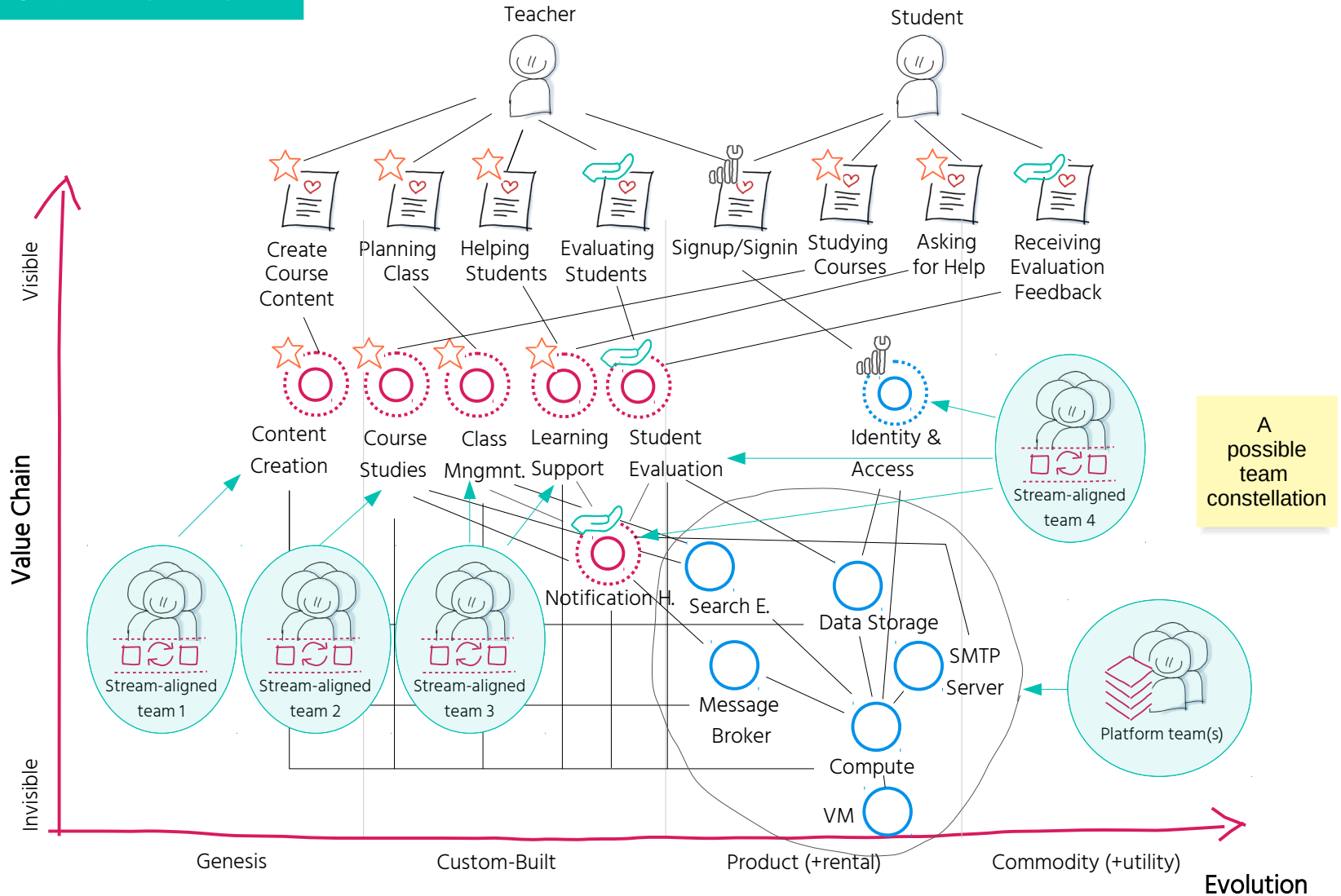


# Architecture For Flow

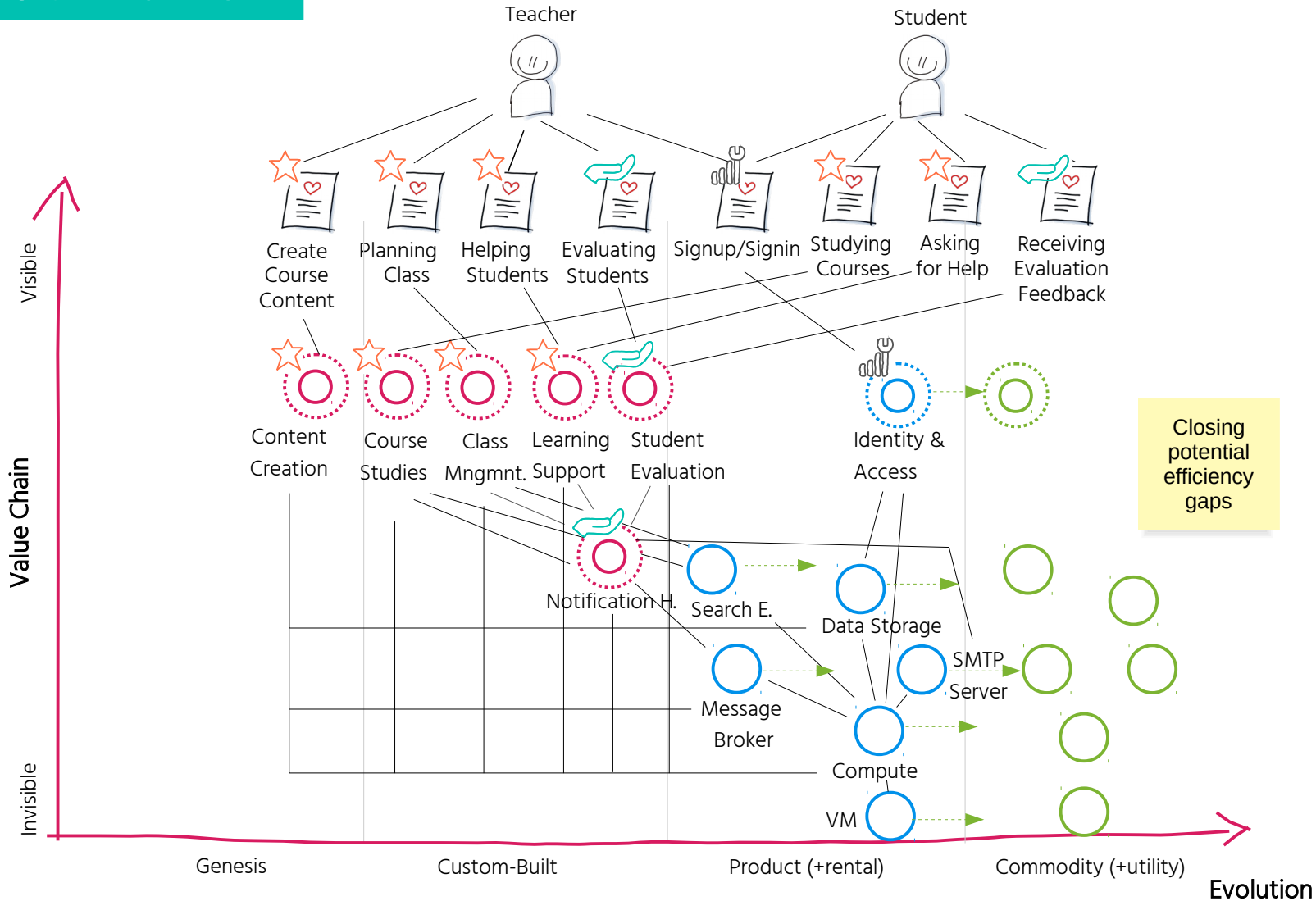




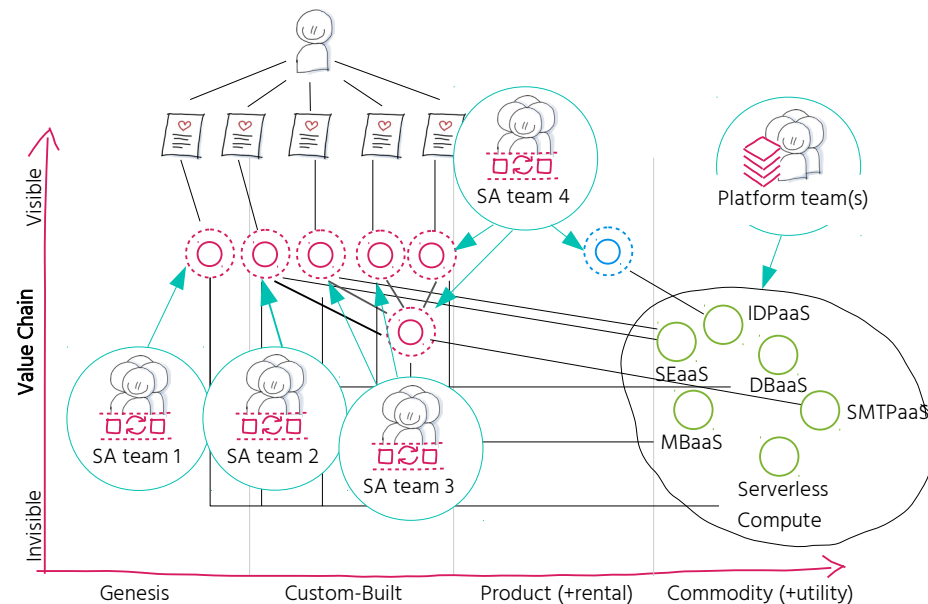
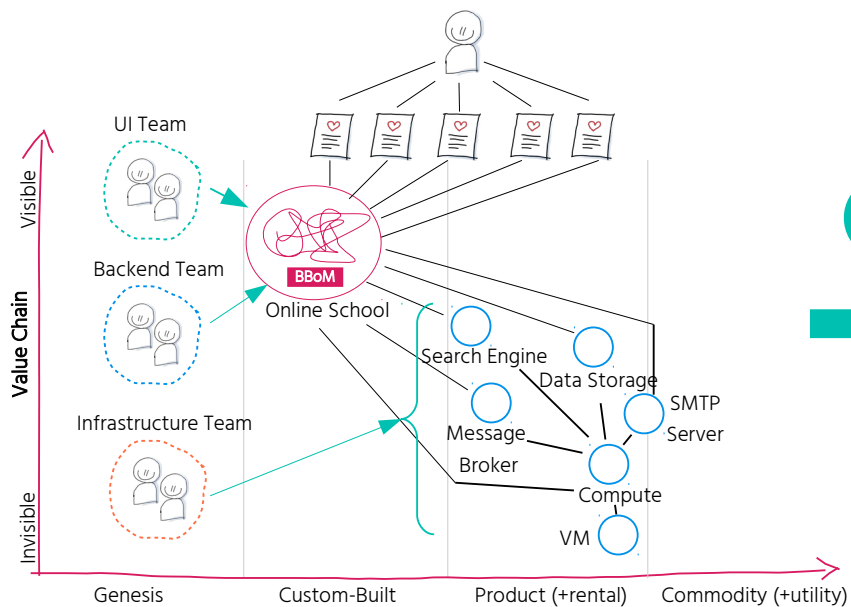
# Architecture For Flow



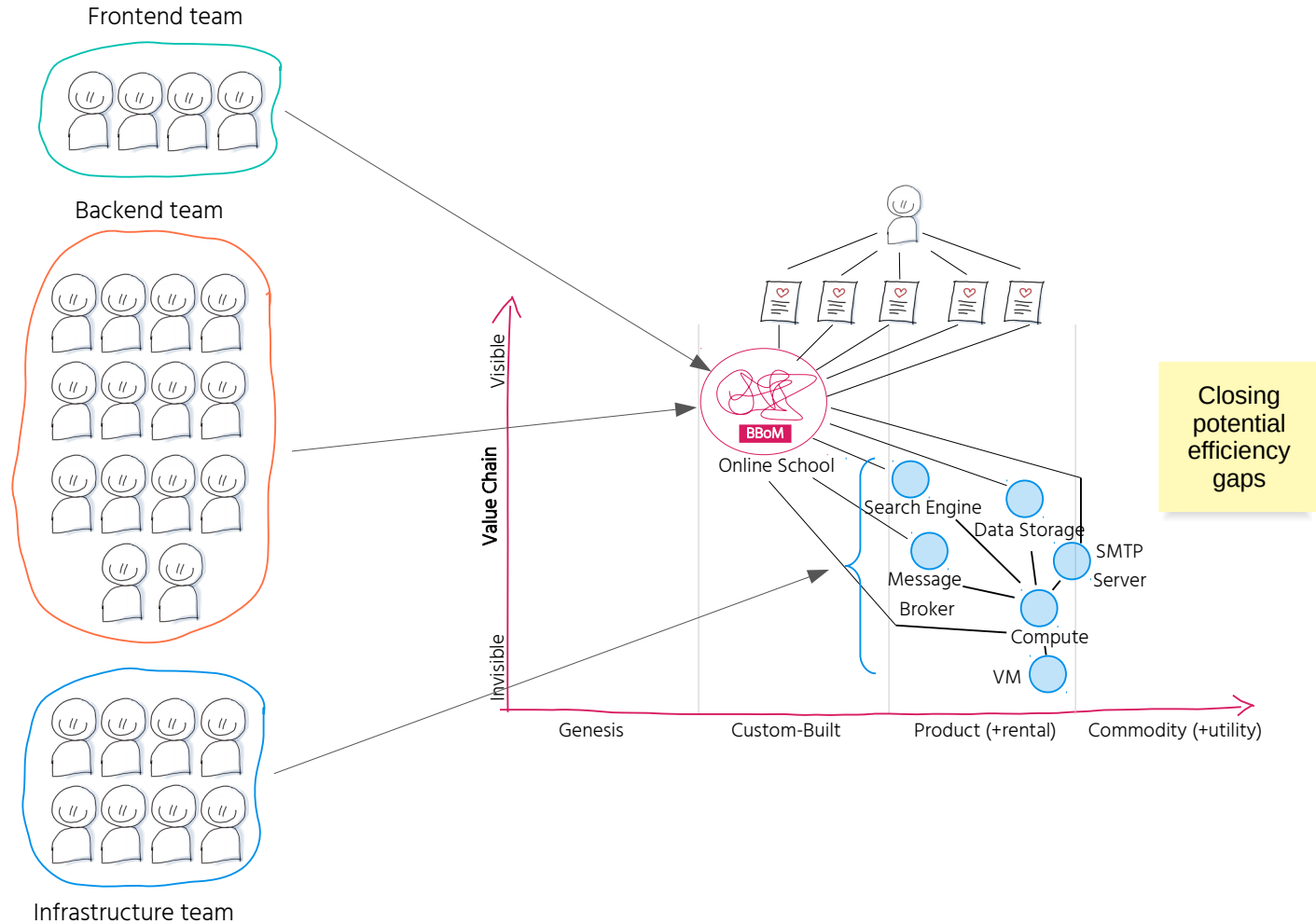
# Architecture For Flow



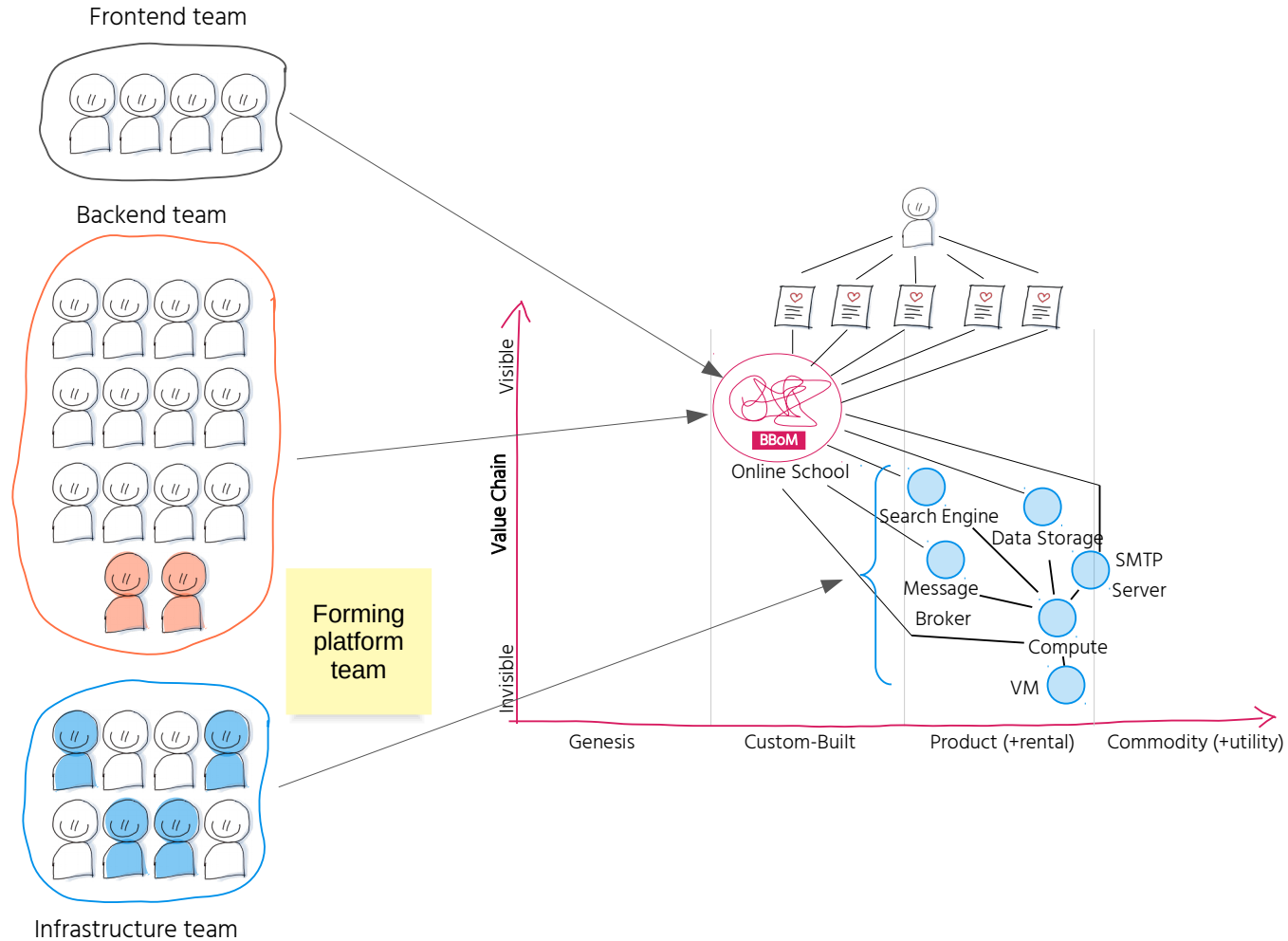
# How to transition?



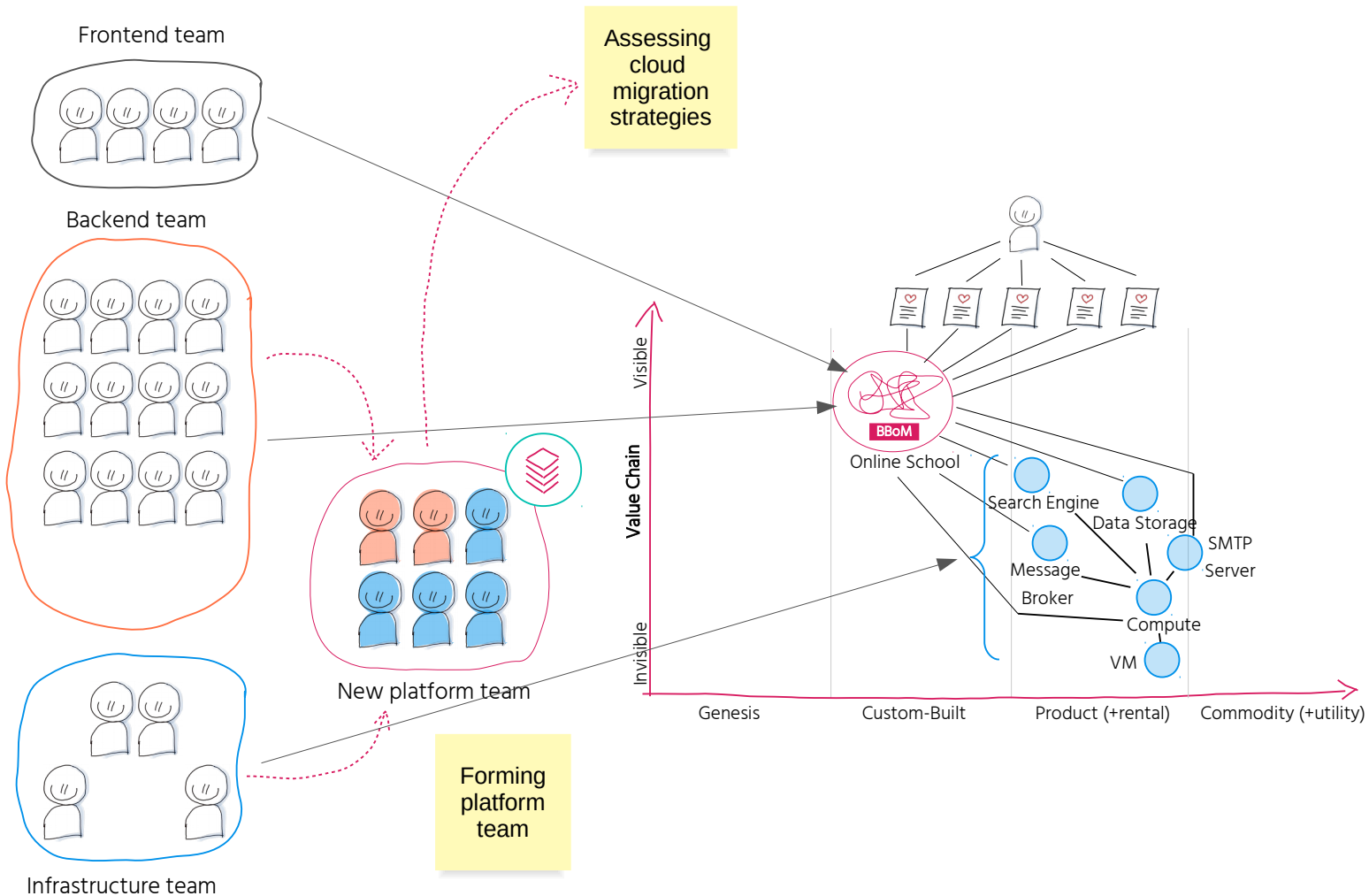
# Implementing Flow Optimization



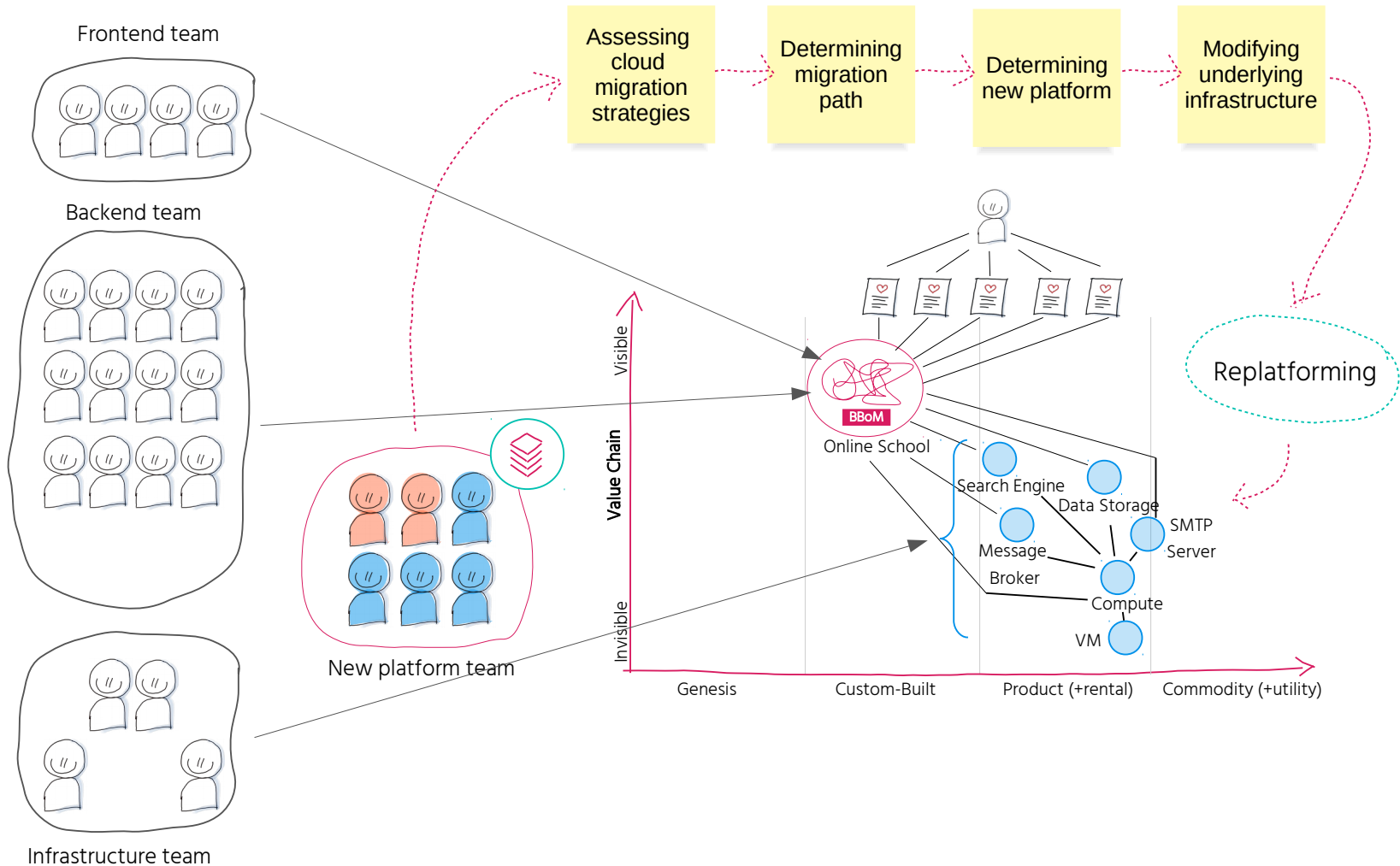
# Implementing Flow Optimization



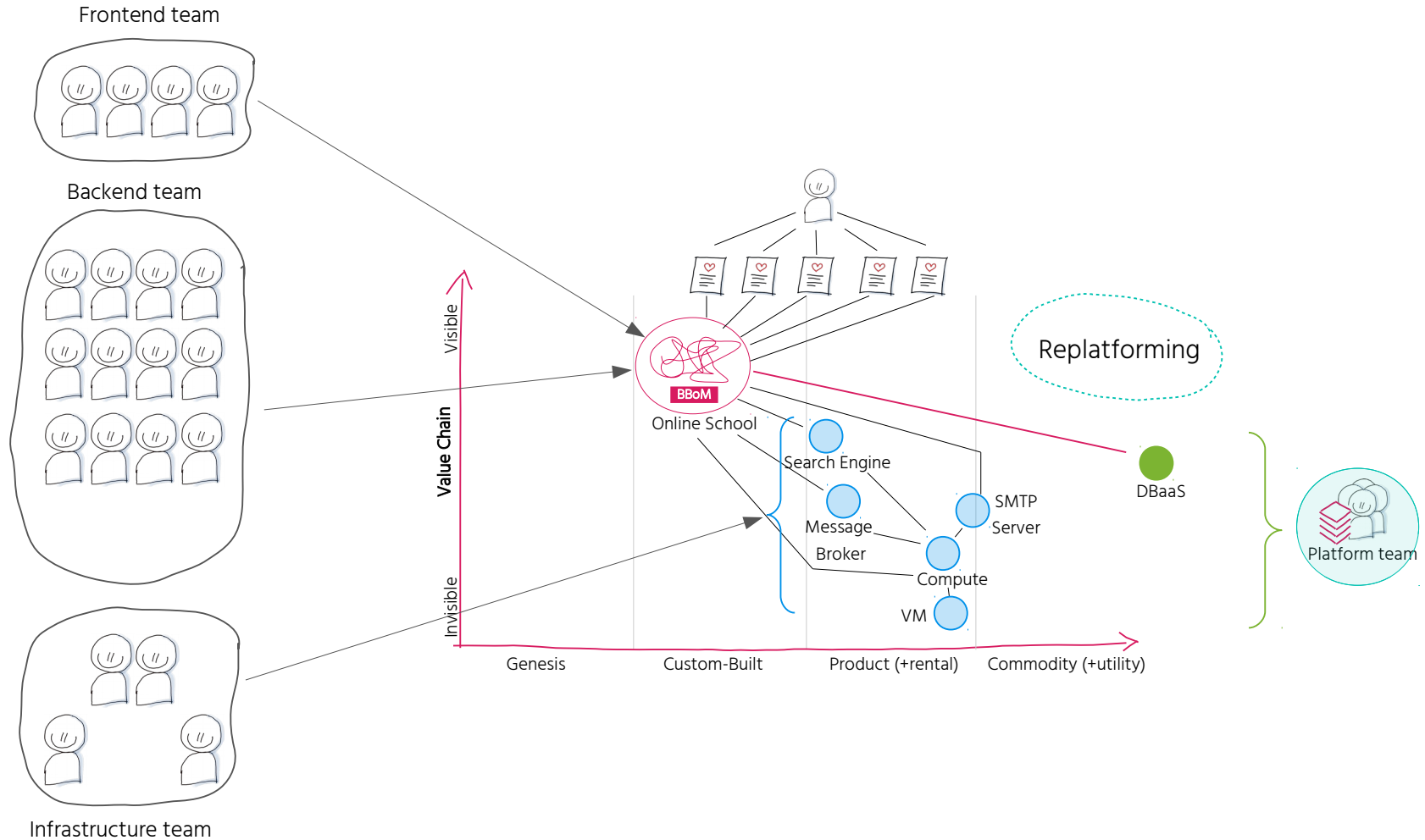
# Implementing Flow Optimization



# Implementing Flow Optimization

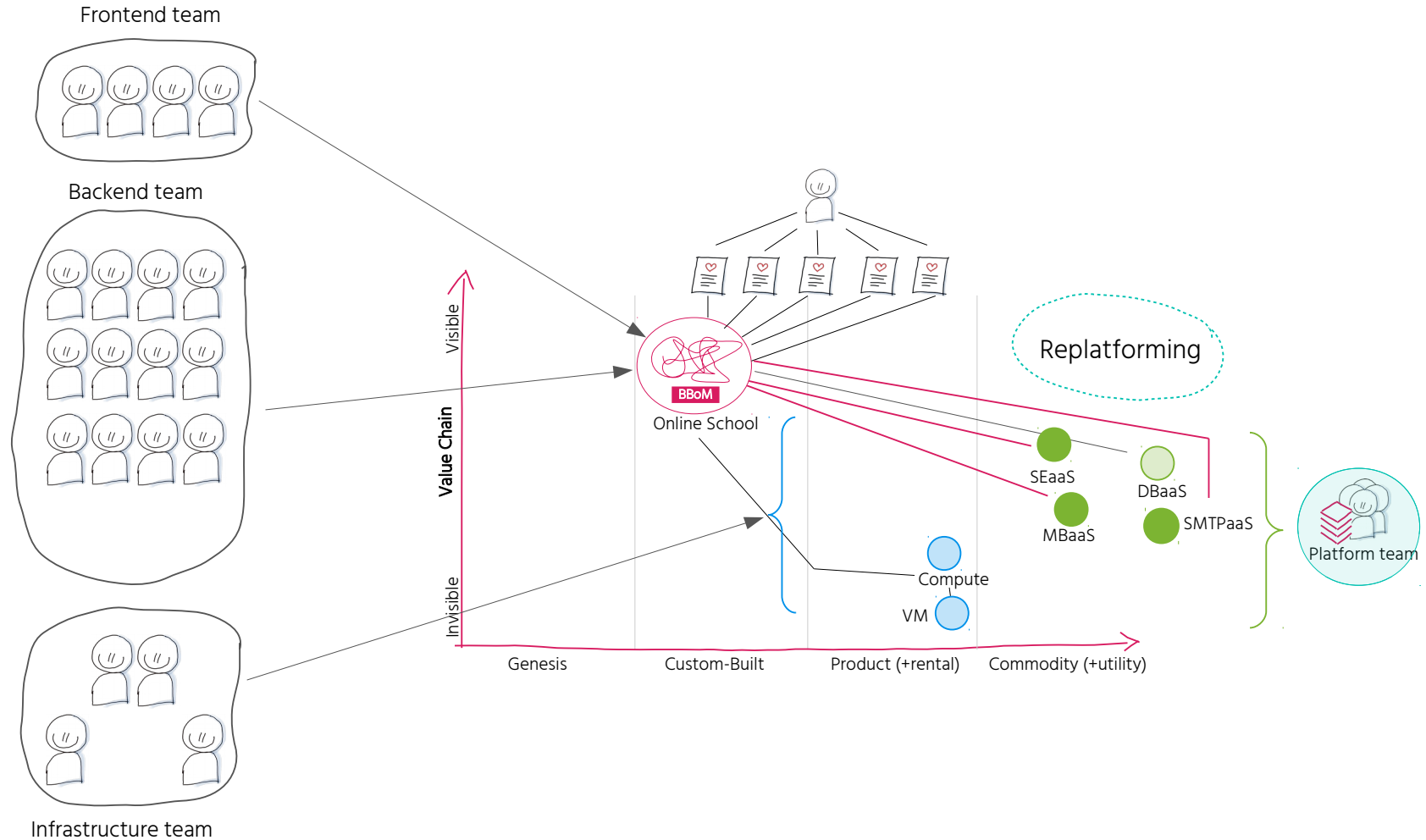


# Implementing Flow Optimization

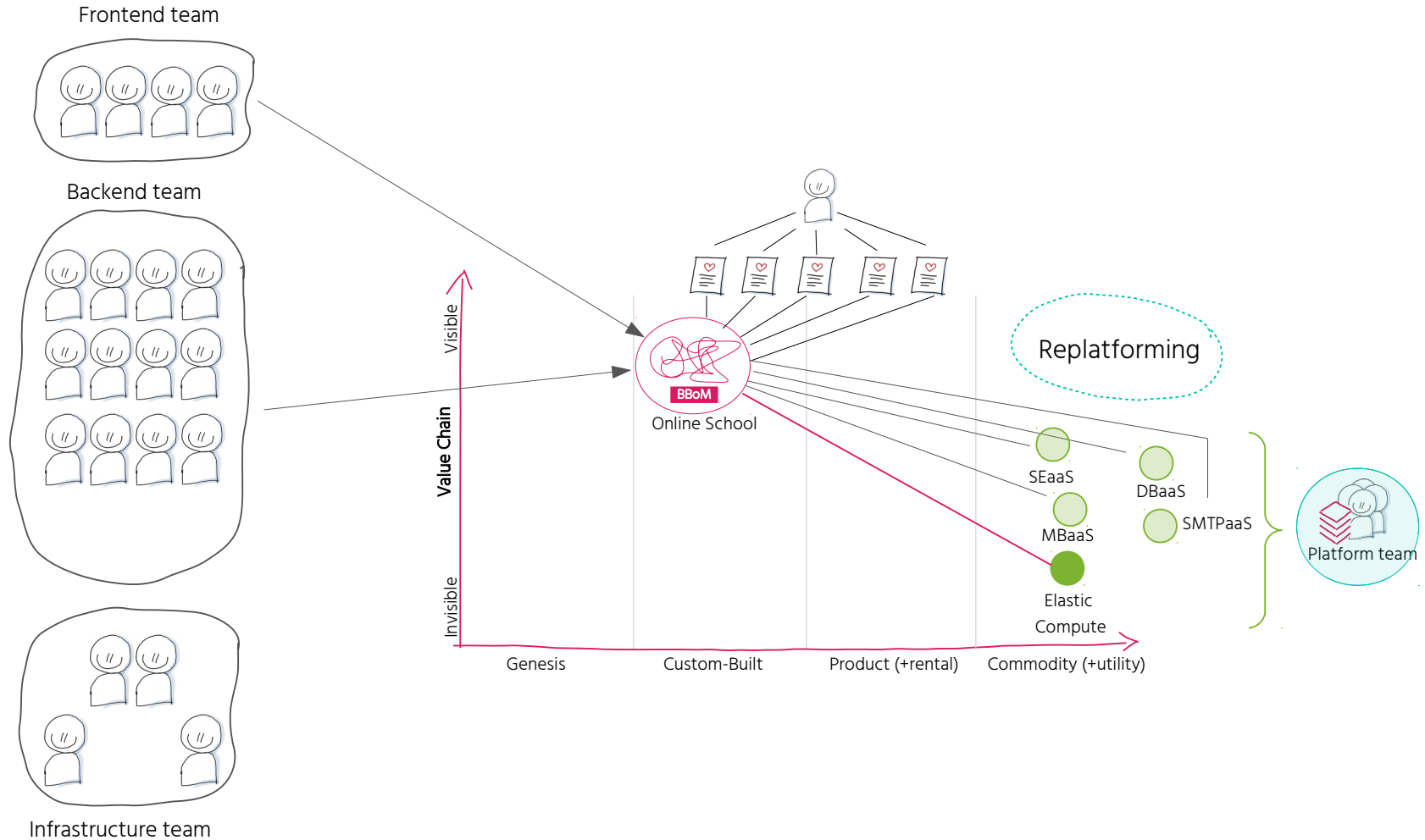




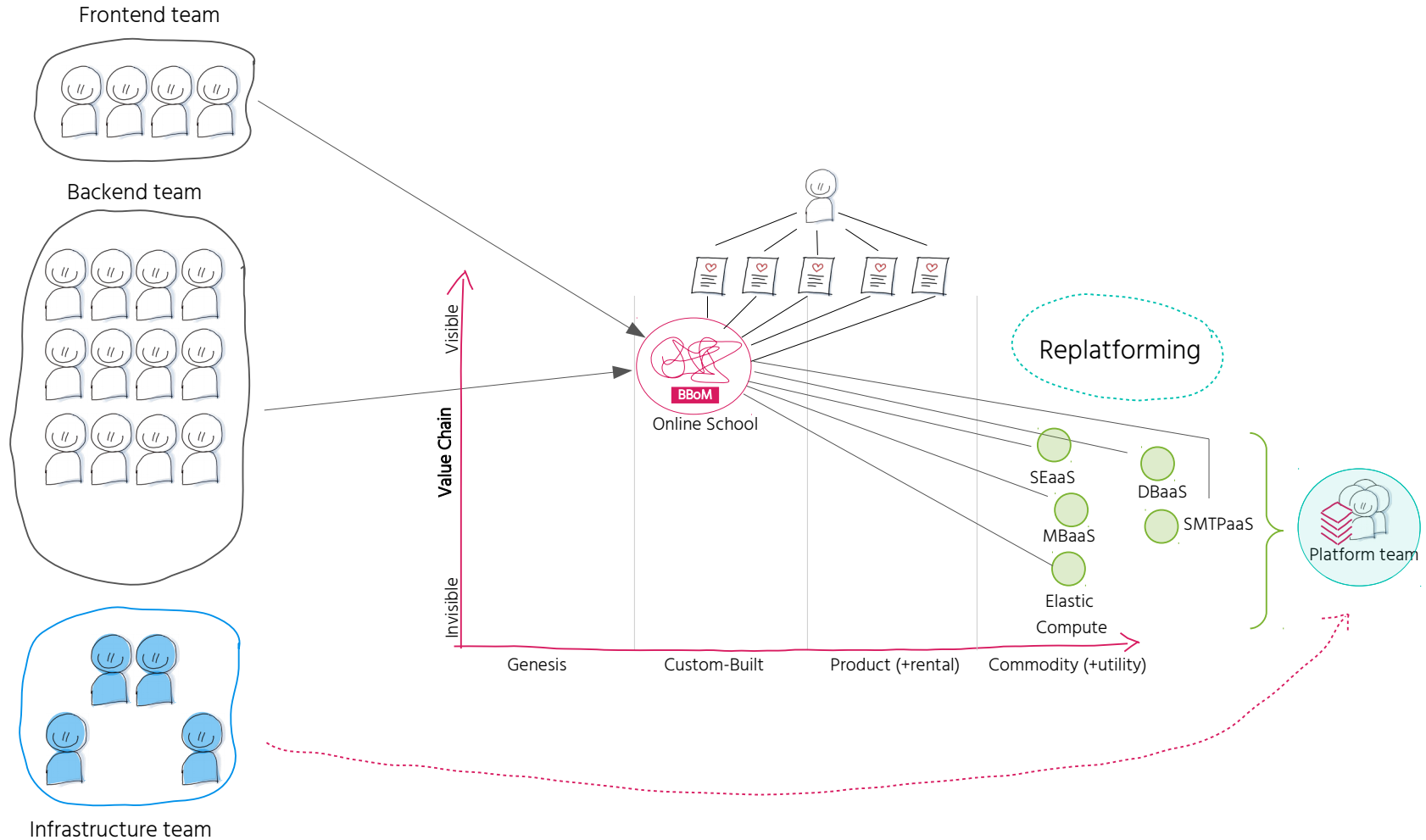
# Implementing Flow Optimization



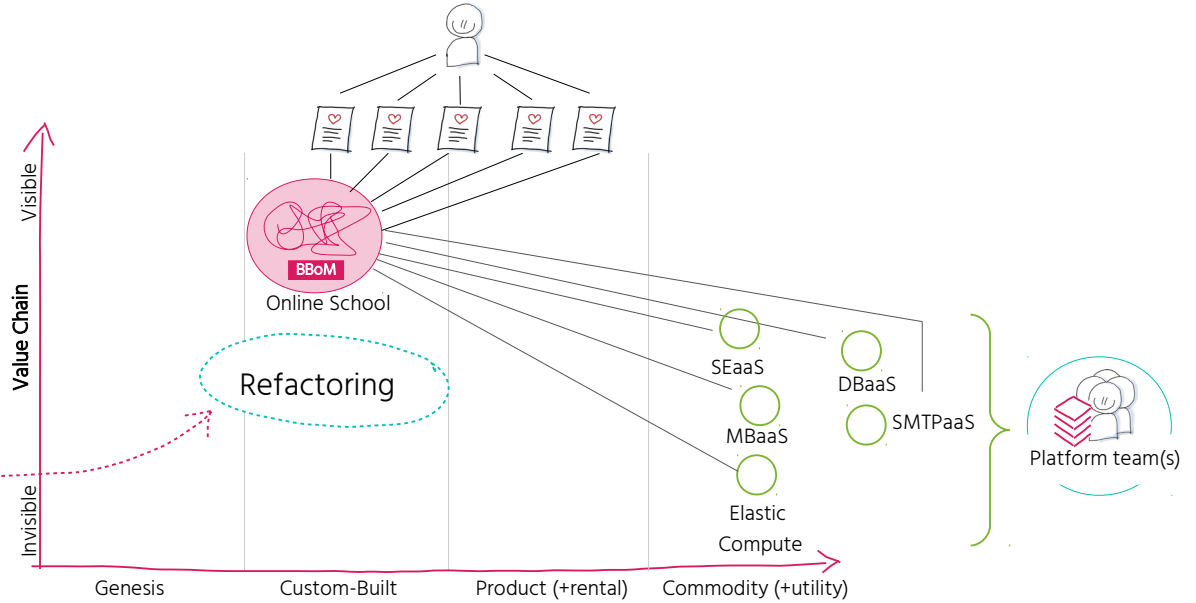
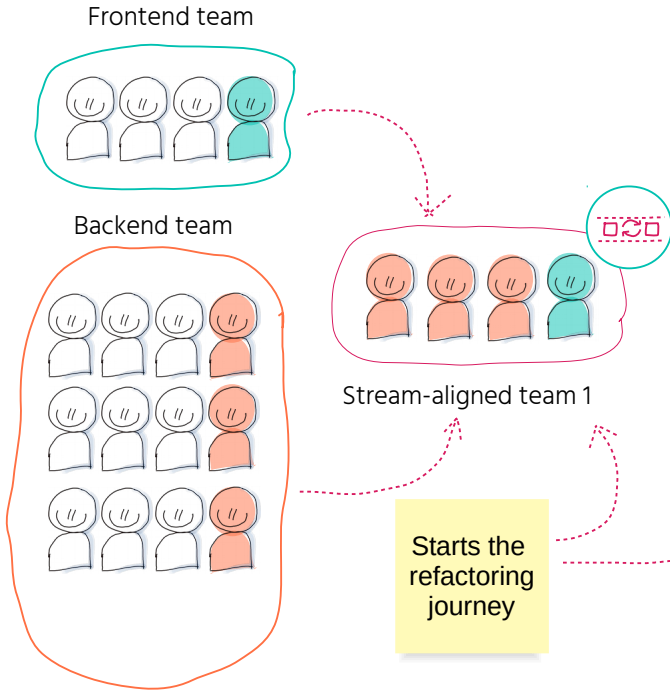
# Implementing Flow Optimization



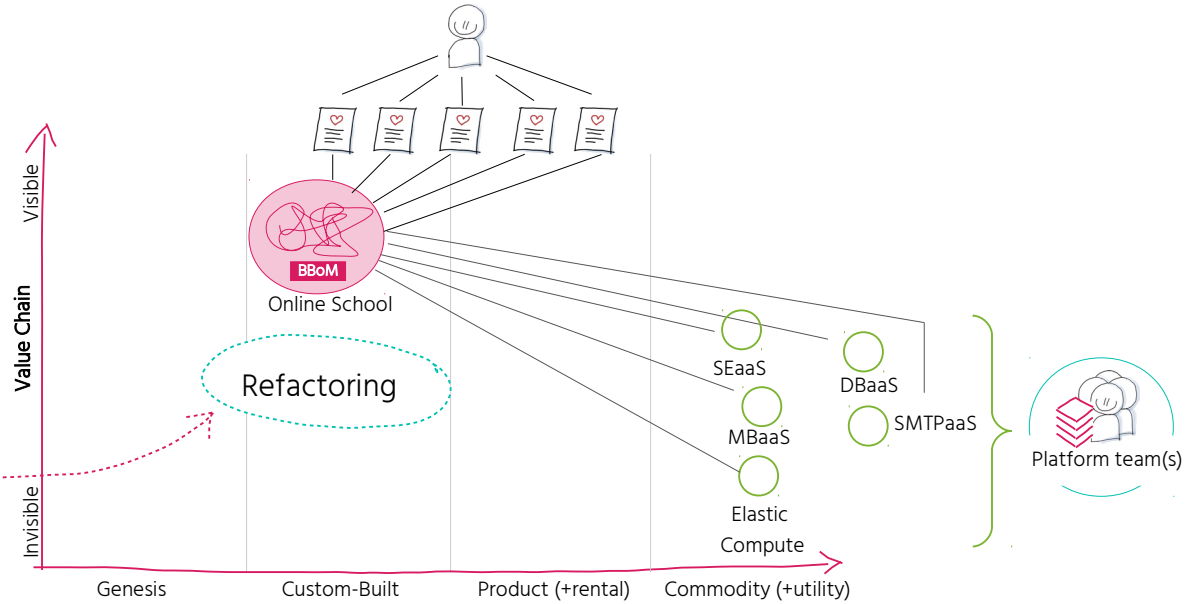
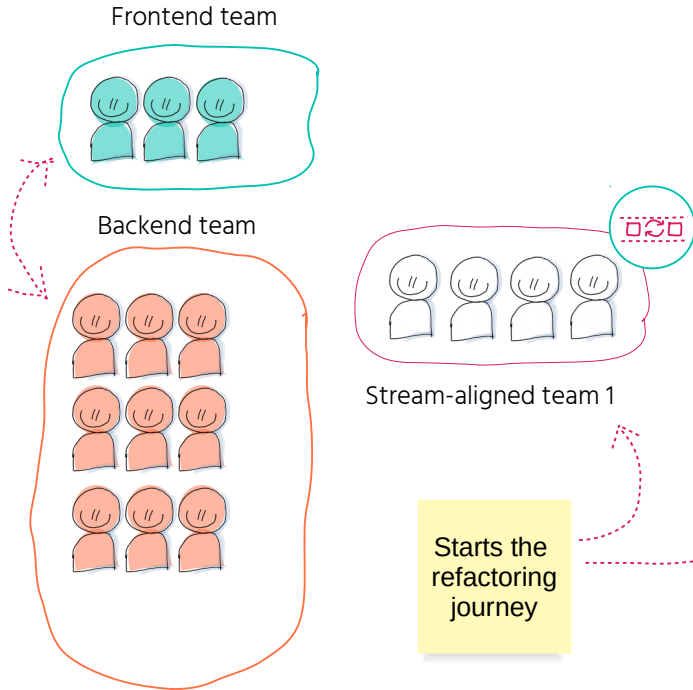
# Implementing Flow Optimization



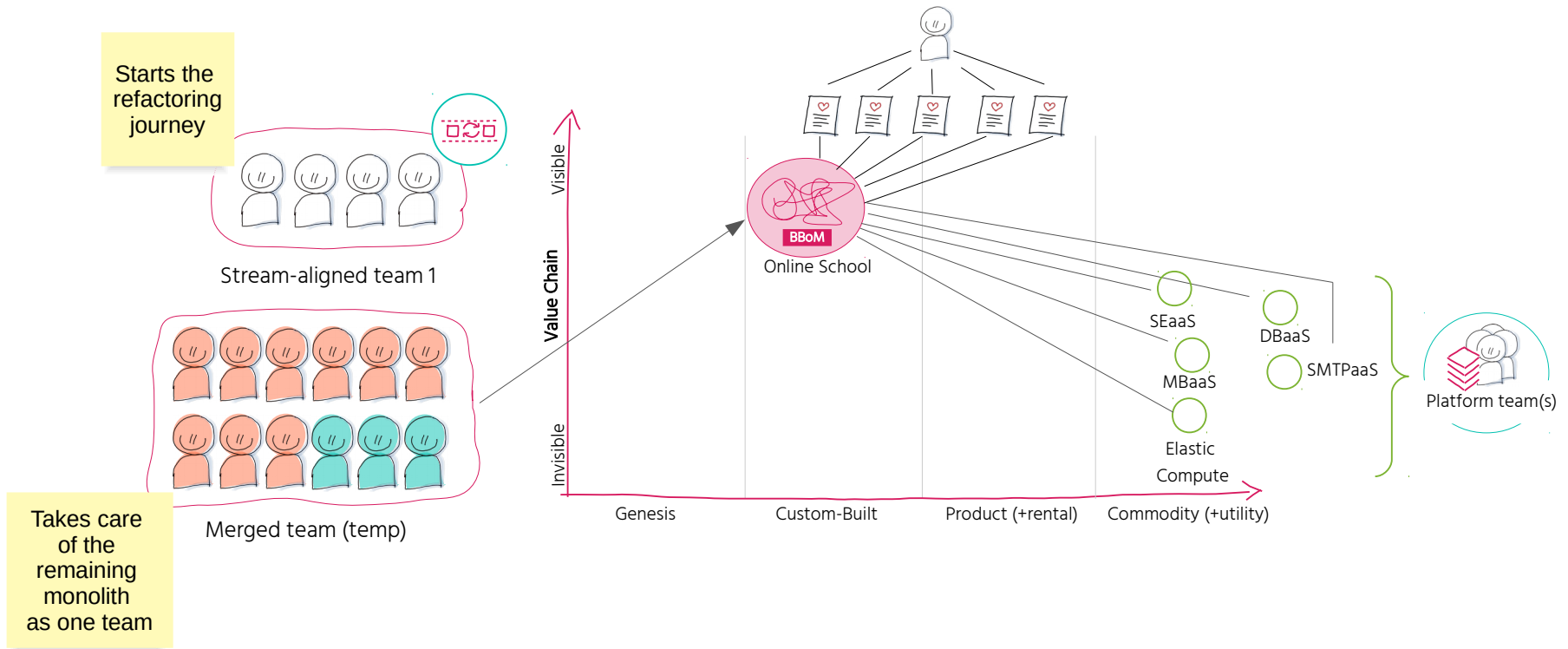
# Implementing Flow Optimization



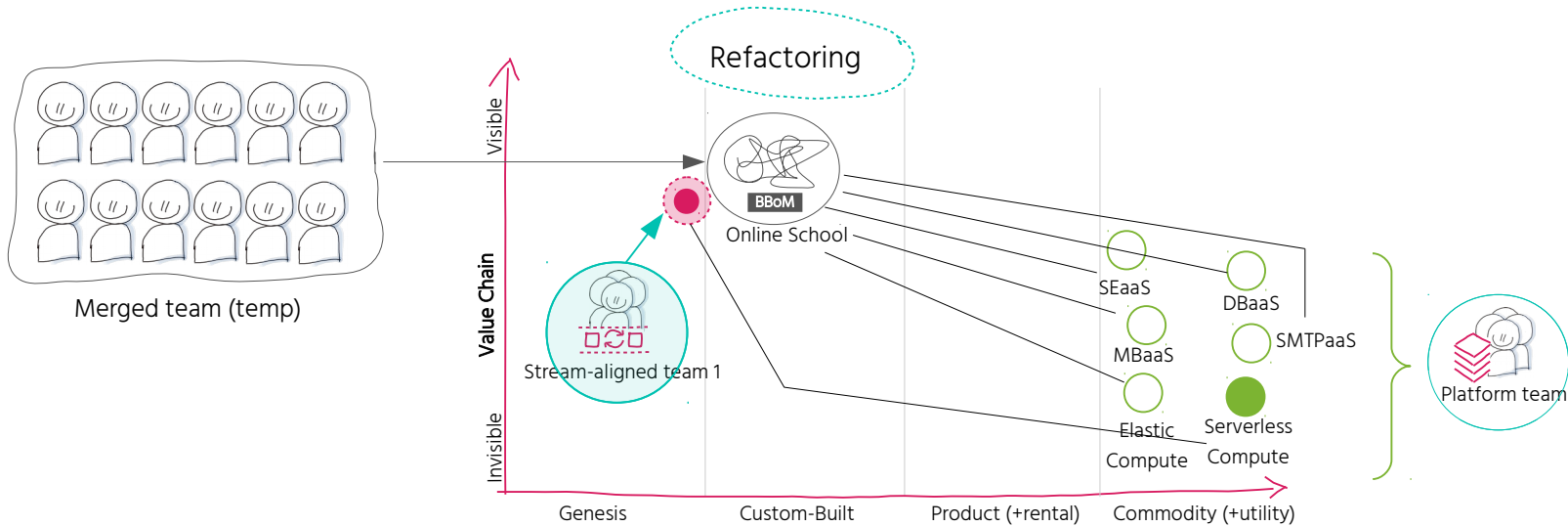
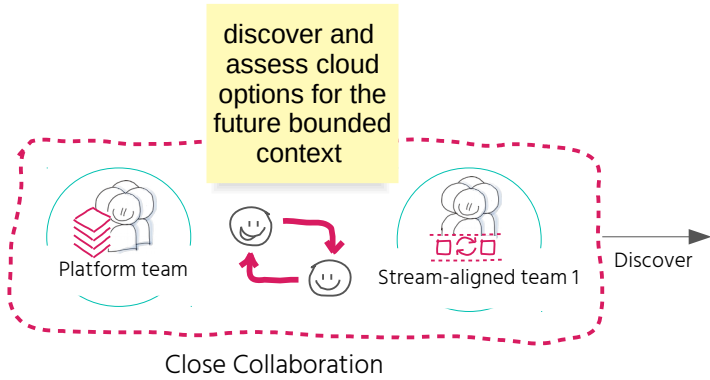
# Implementing Flow Optimization



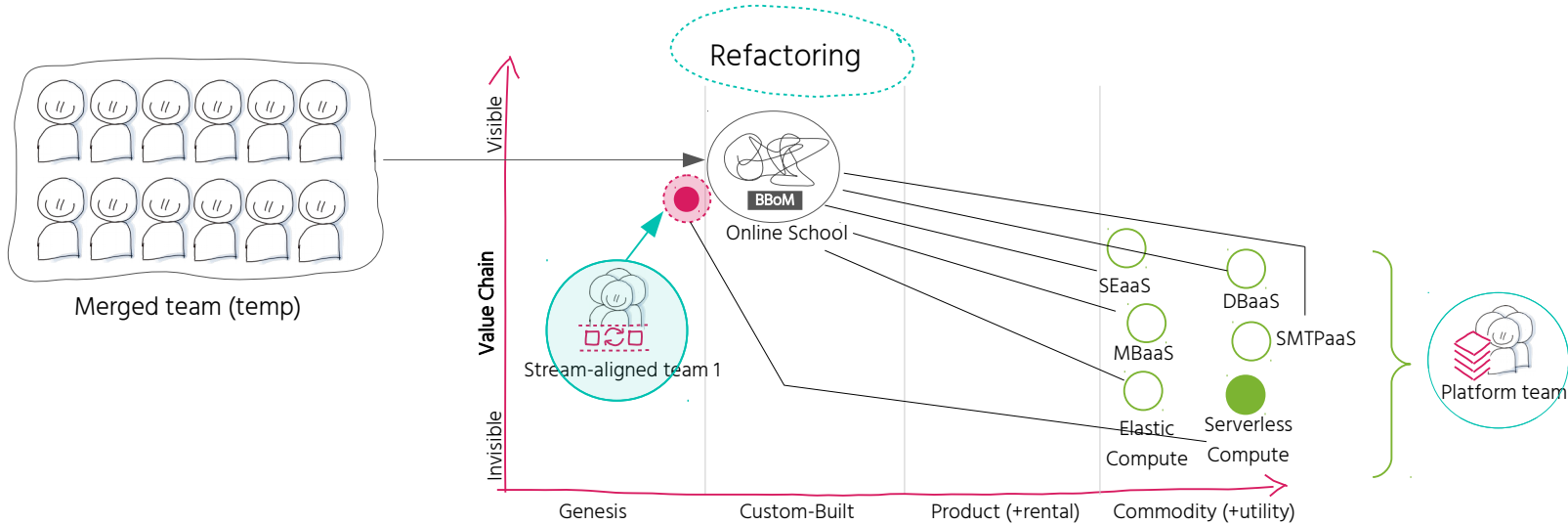
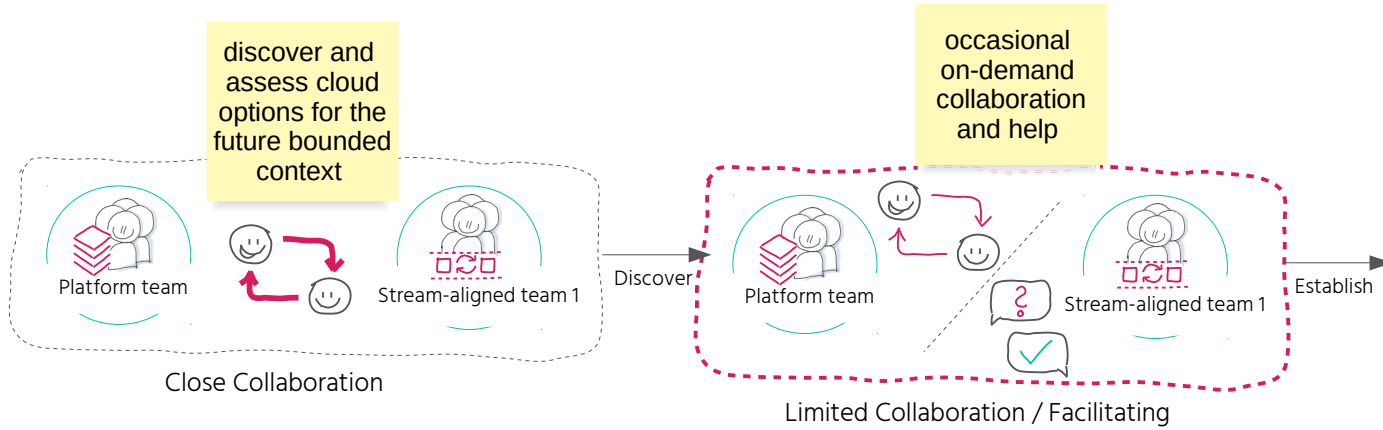
# Implementing Flow Optimization



# Implementing Flow Optimization

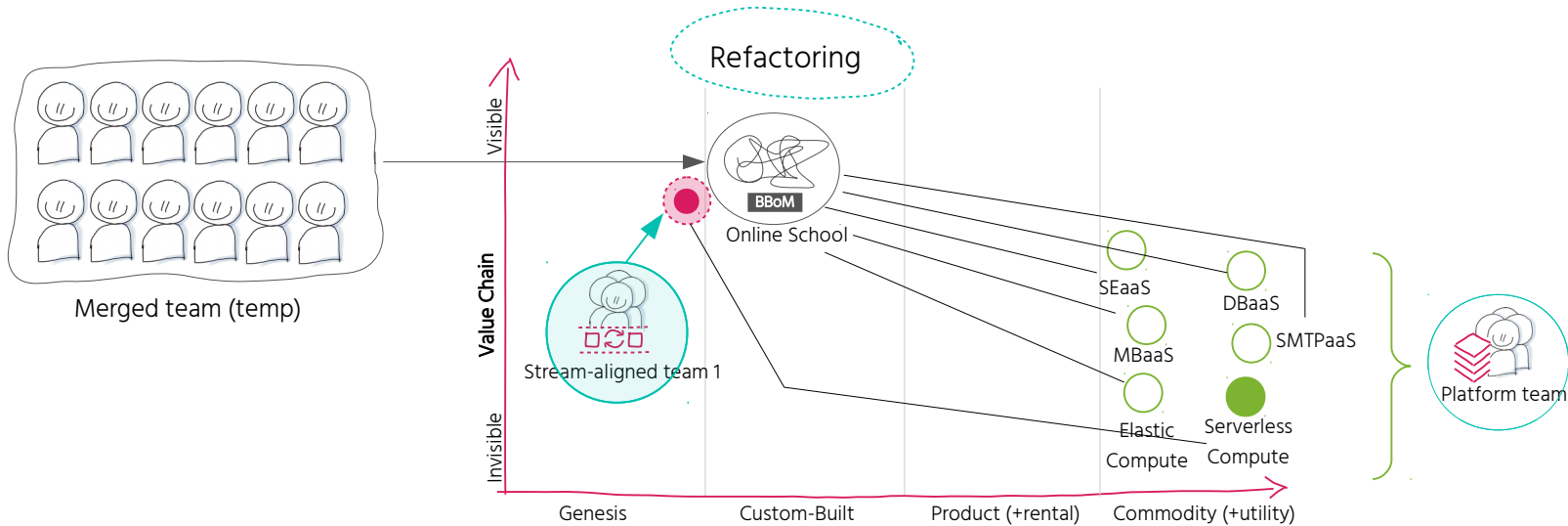
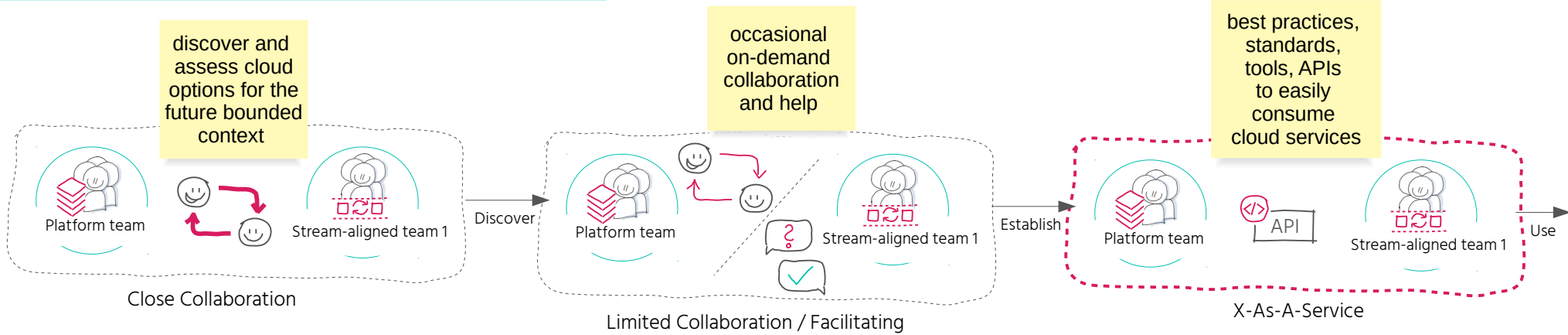


# Implementing Flow Optimization

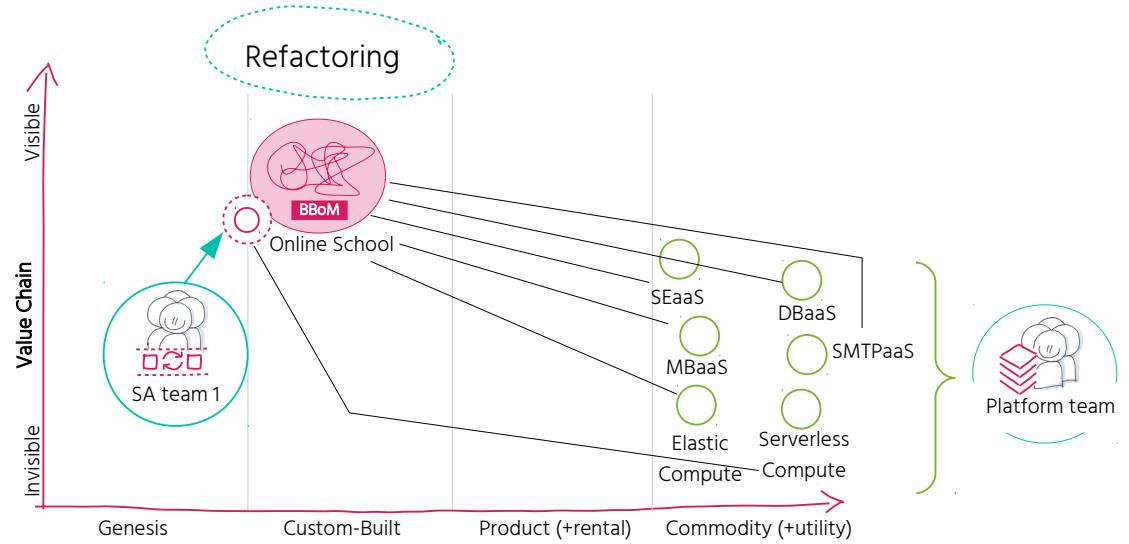
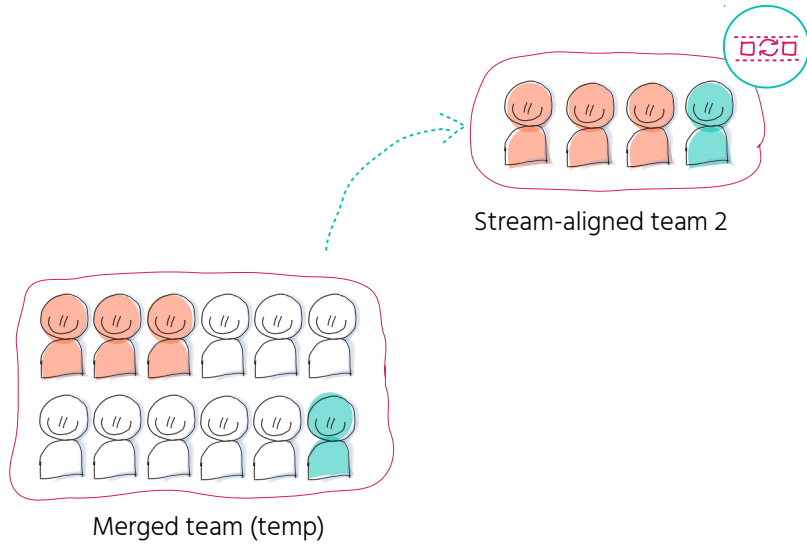




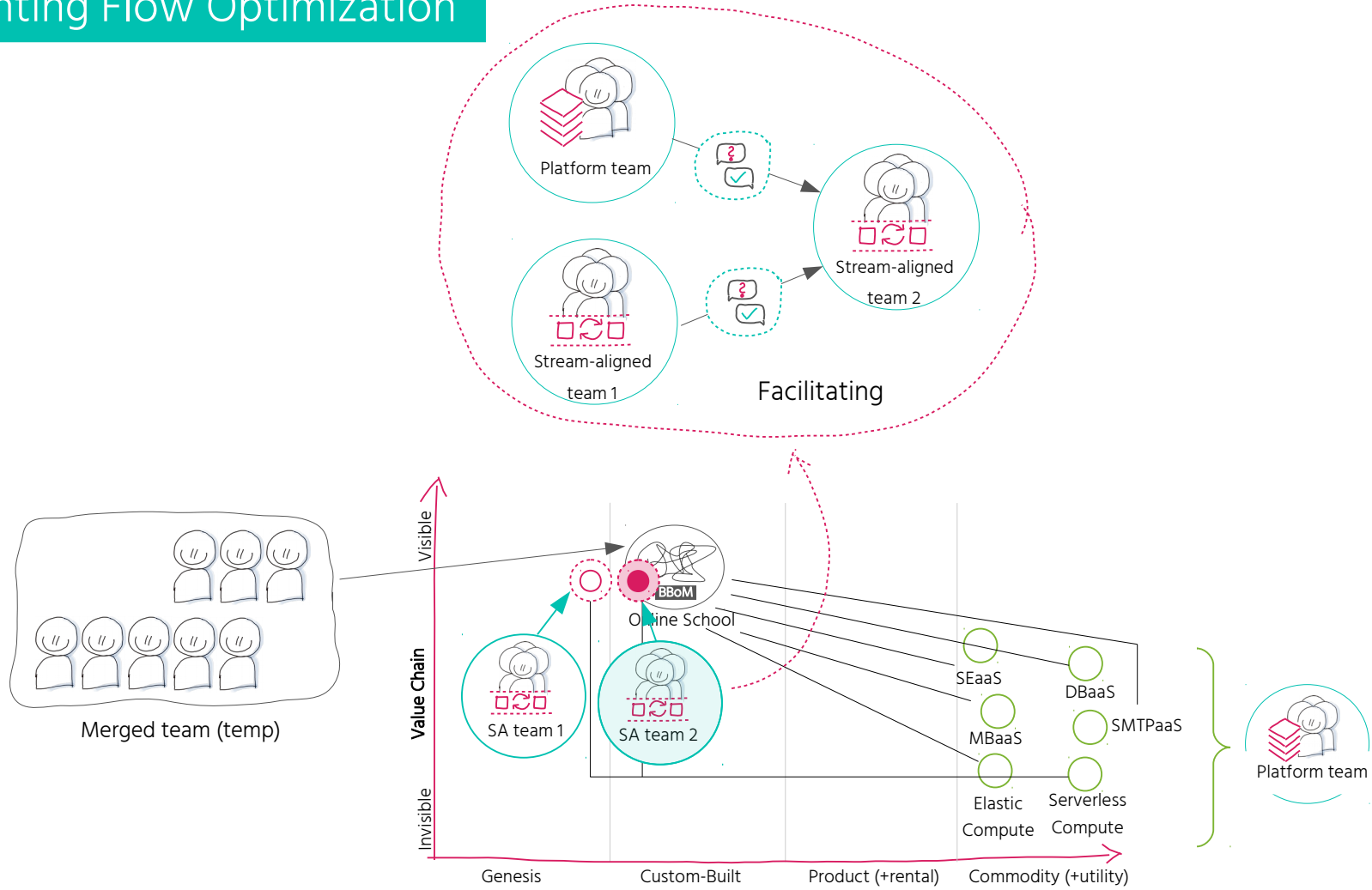
# Implementing Flow Optimization



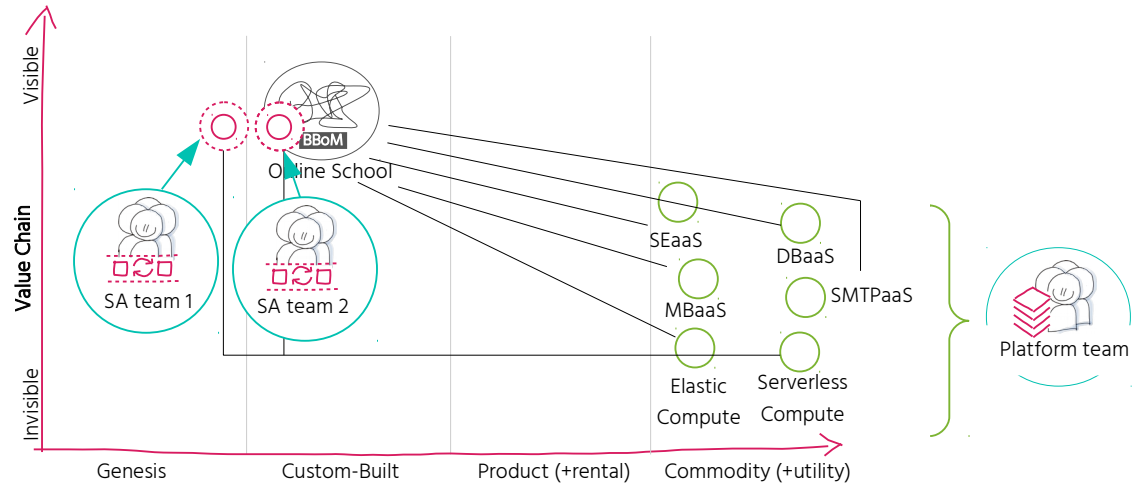
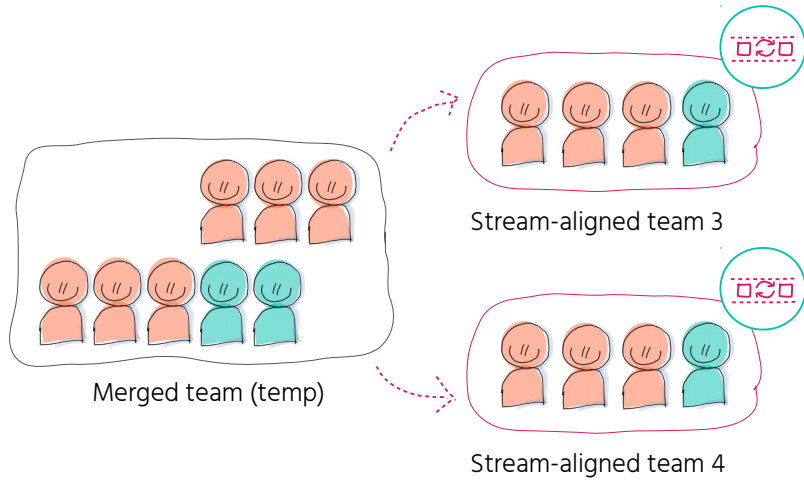
# Implementing Flow Optimization



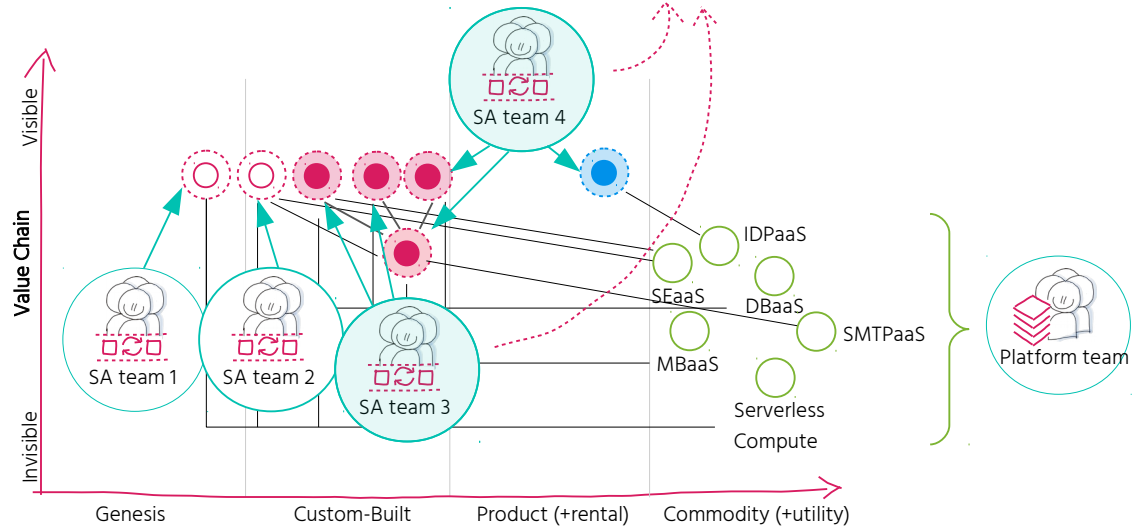
# Implementing Flow Optimization



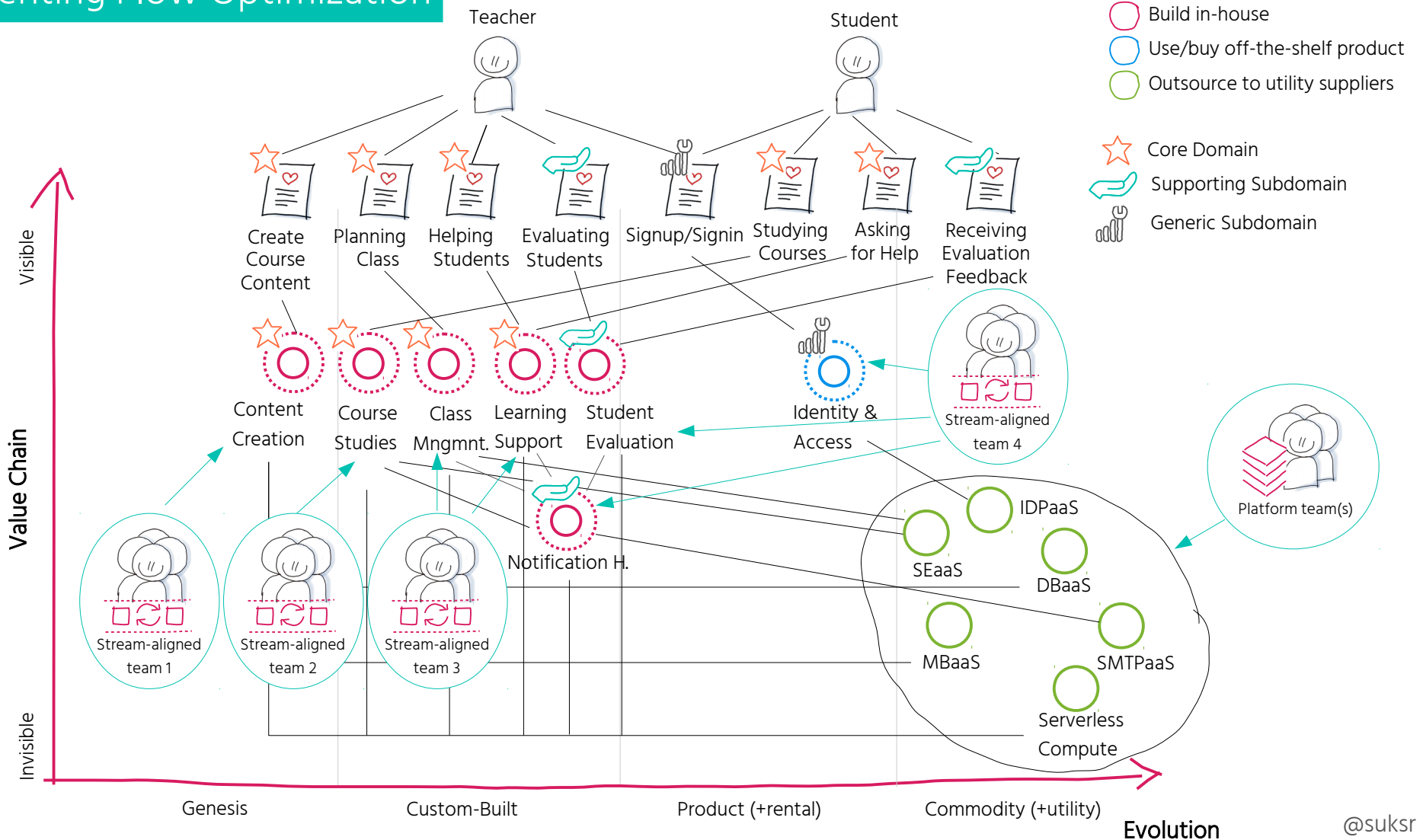
# Implementing Flow Optimization



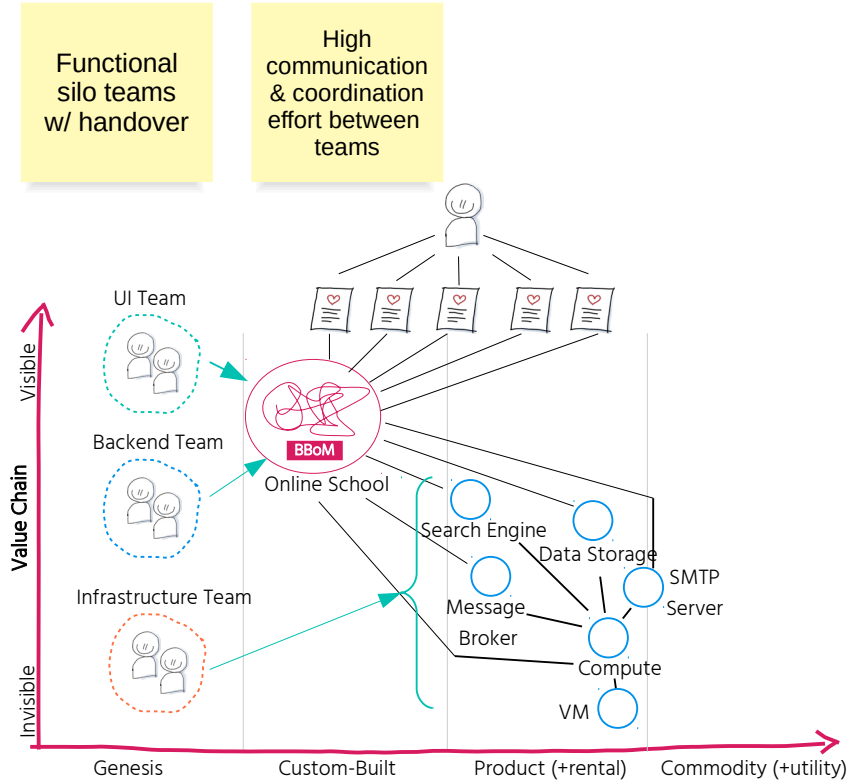
# Implementing Flow Optimization



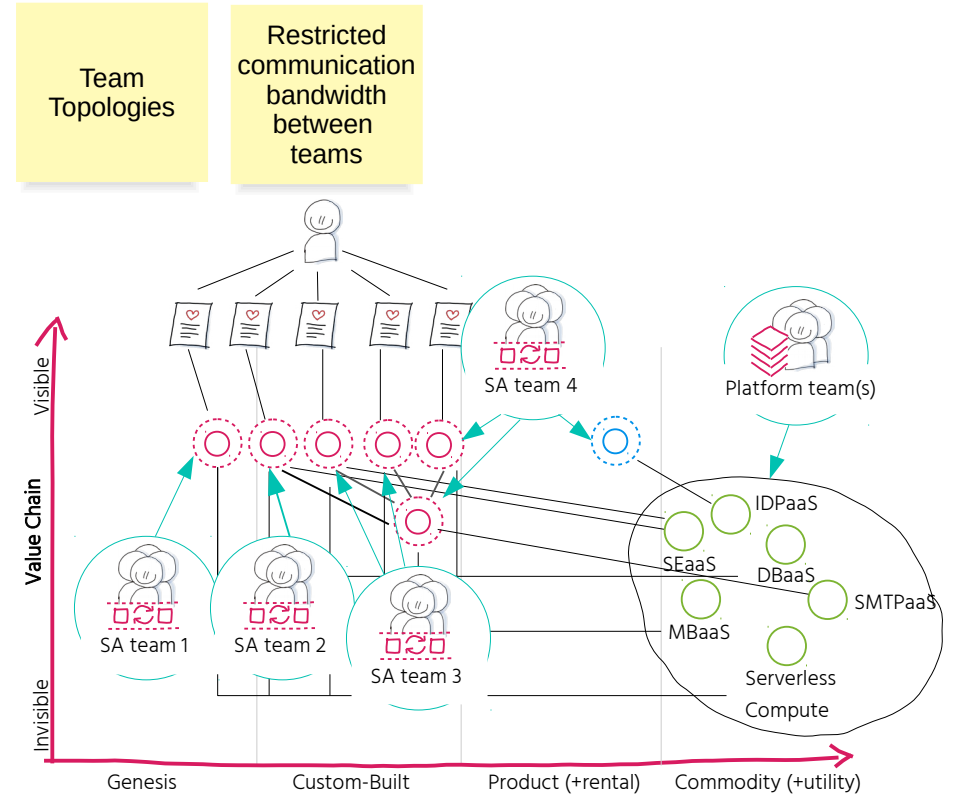
# Implementing Flow Optimization



# What are we leaving behind?



# What are we adopting?



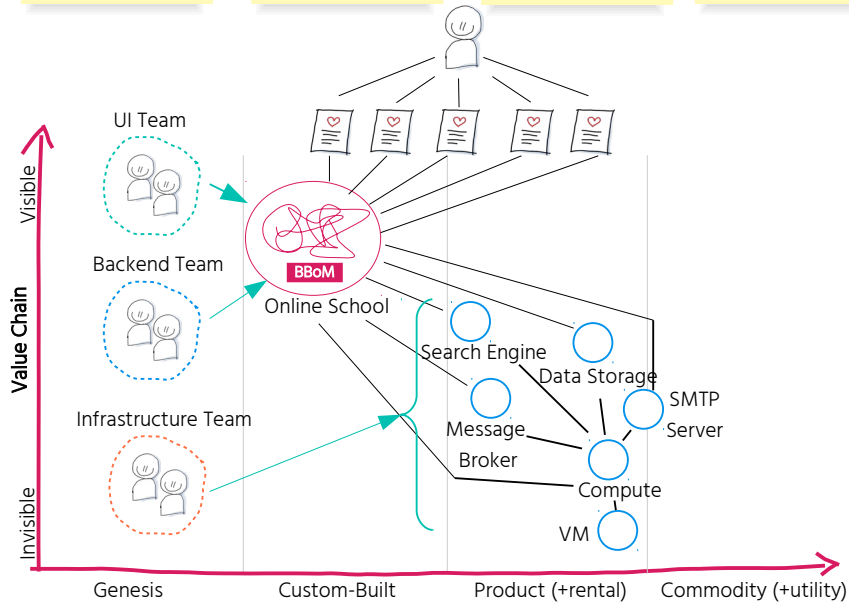
# What are we leaving behind?

Functional silo teams w/ handover

High communication & coordination effort between teams

Monolithic big ball of mud w/ messy model and fuzzy boundaries

No clear ownership boundaries



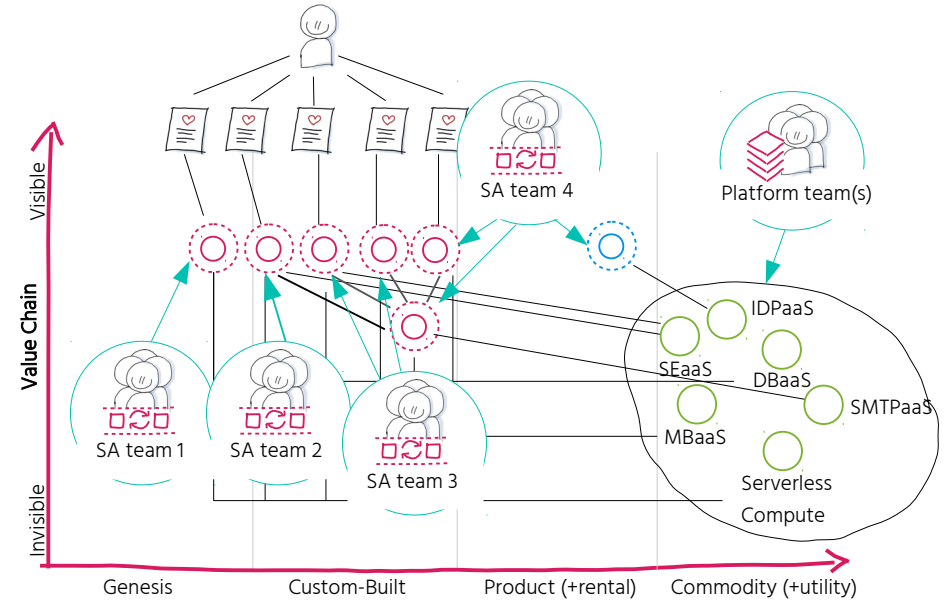
# What are we adopting?

Team Topologies

Restricted communication bandwidth between teams

Decomposed system w/ clear models & bounded contexts

Clear team ownership boundaries





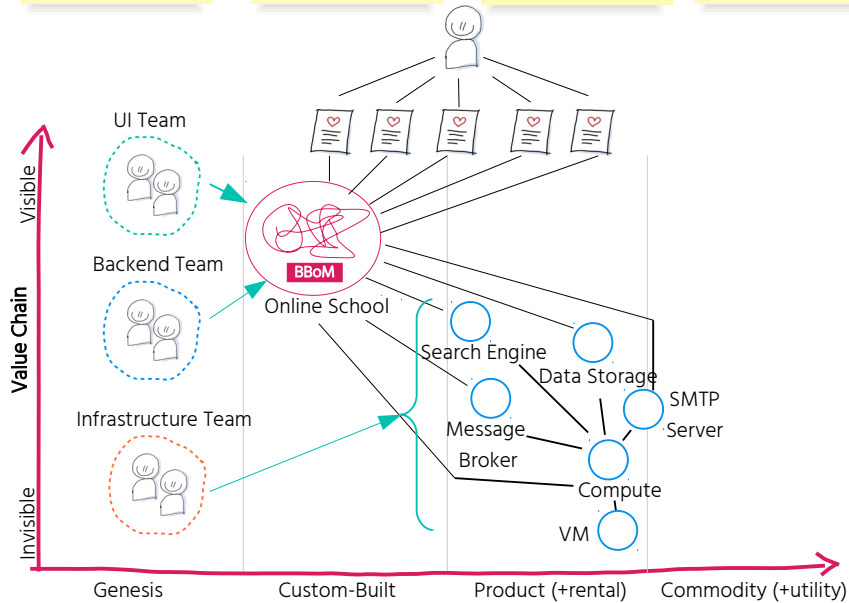
# What are we leaving behind?

Functional silo teams w/ handover

High communication & coordination effort between teams

Monolithic big ball of mud w/ messy model and fuzzy boundaries

No clear ownership boundaries



Tight change coupling

High team cognitive load

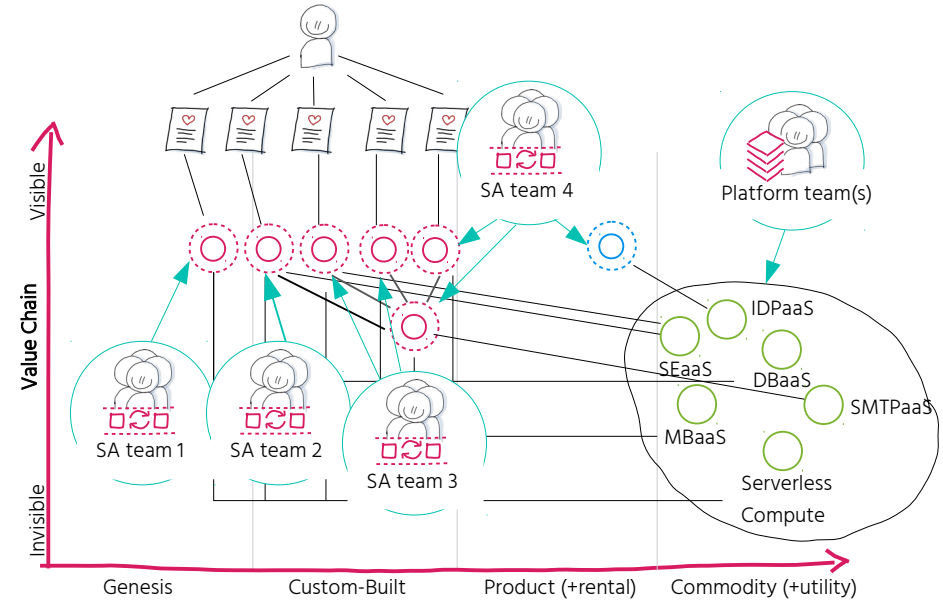
# What are we adopting?

Team Topologies

Restricted communication bandwidth between teams

Decomposed system w/ clear models & bounded contexts

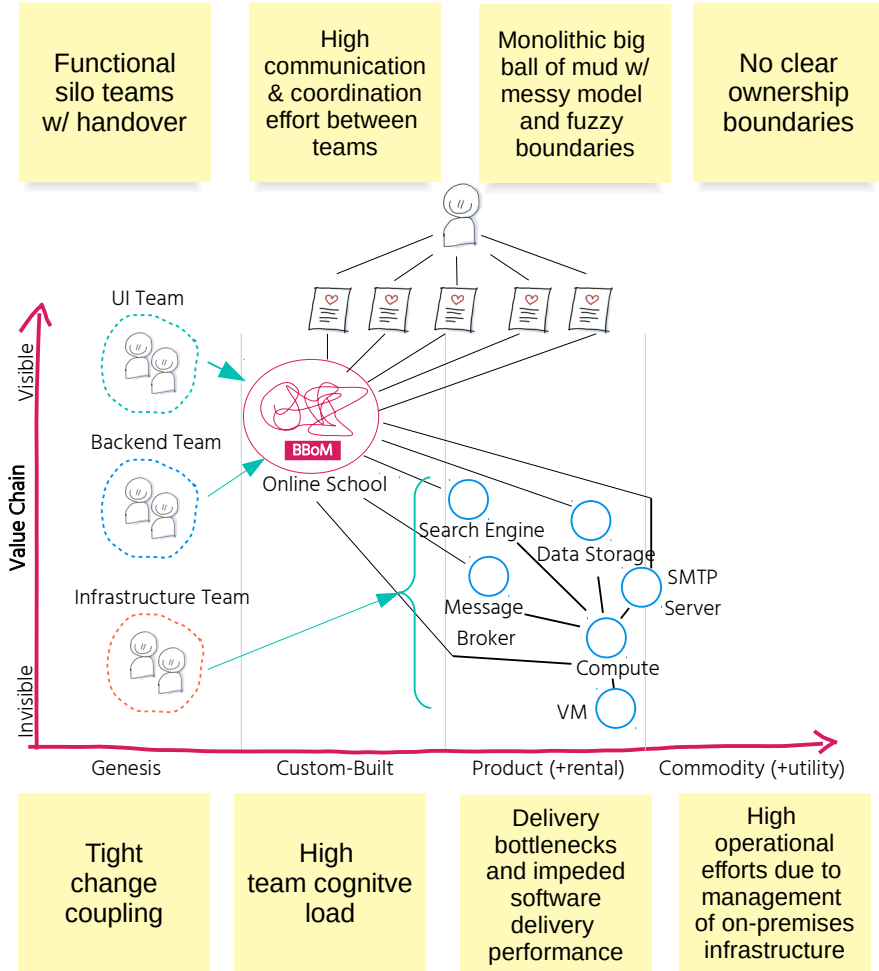
Clear team ownership boundaries



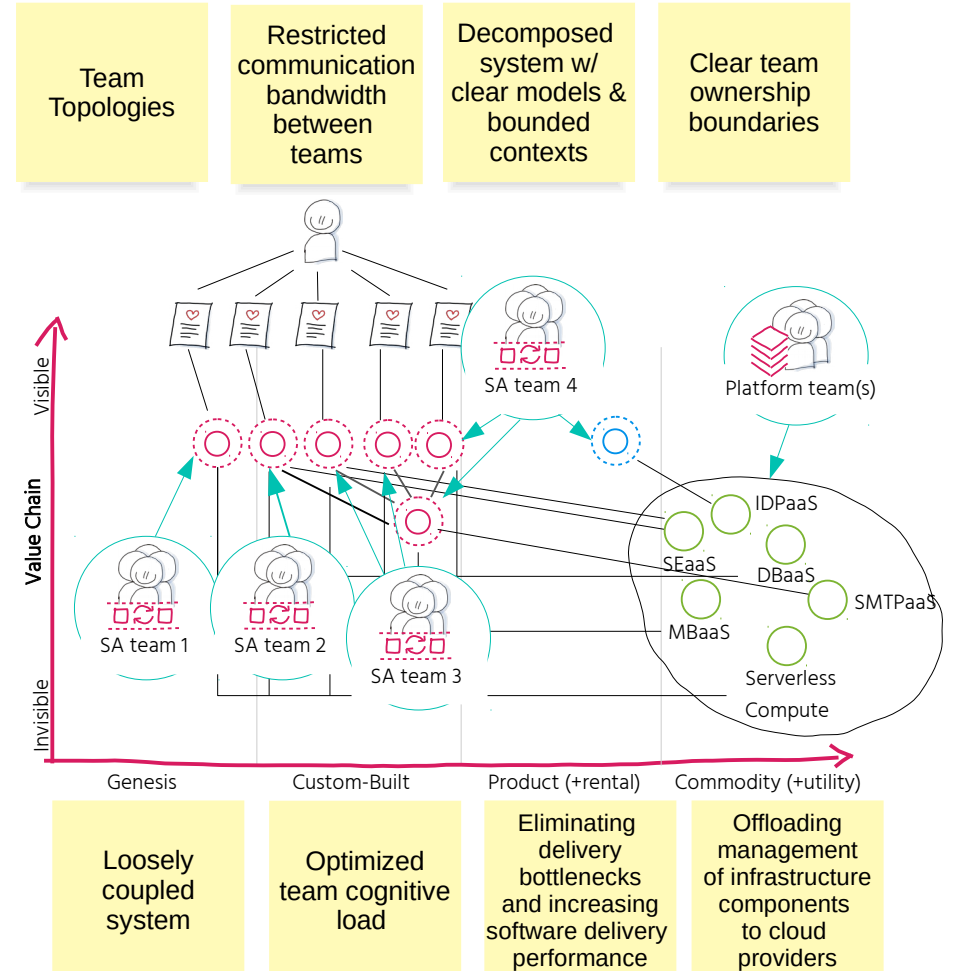
Loosely coupled system

Optimized team cognitive load

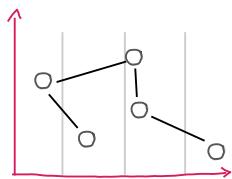
# What are we leaving behind?



# What are we adopting?



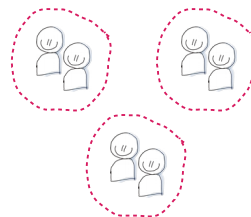
# Key Takeaways



Wardley Mapping



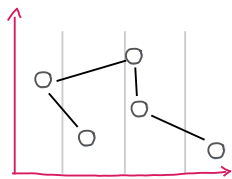
Domain-Driven Design



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

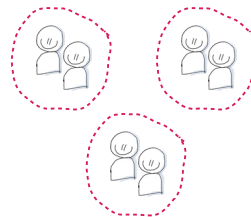
# Key Takeaways



Wardley Mapping



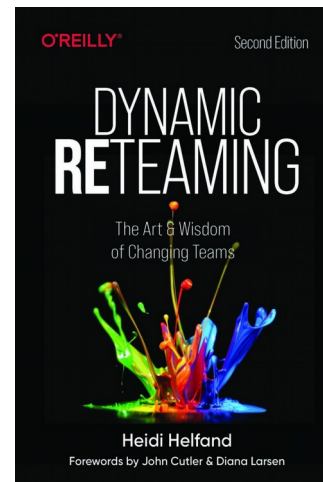
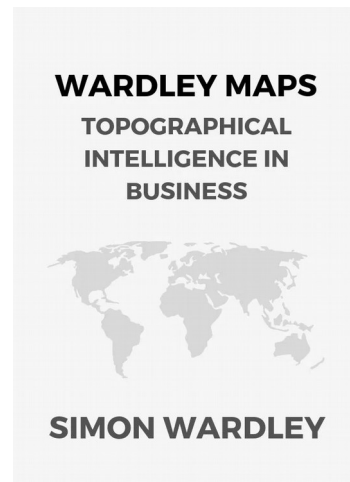
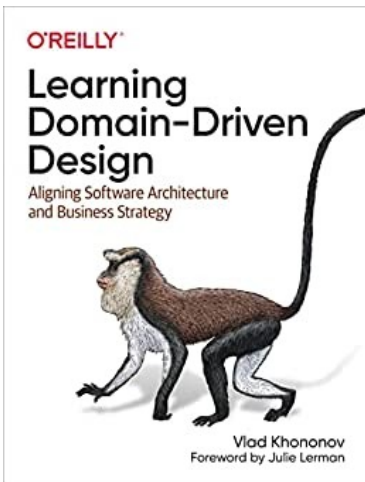
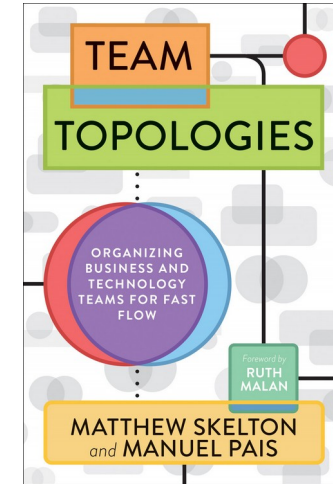
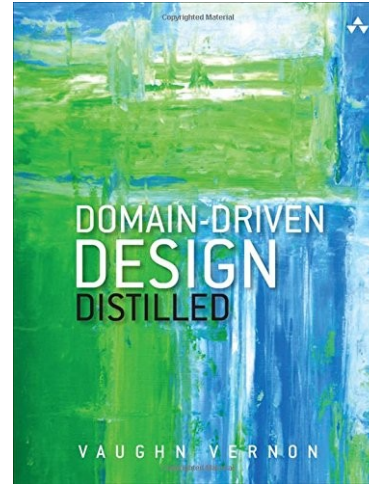
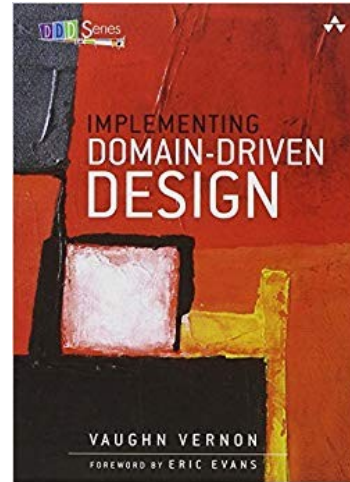
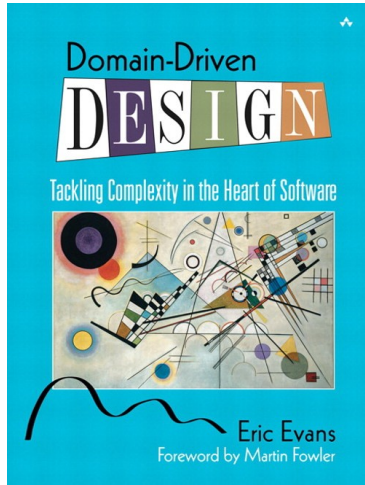
Domain-Driven Design



Team Topologies

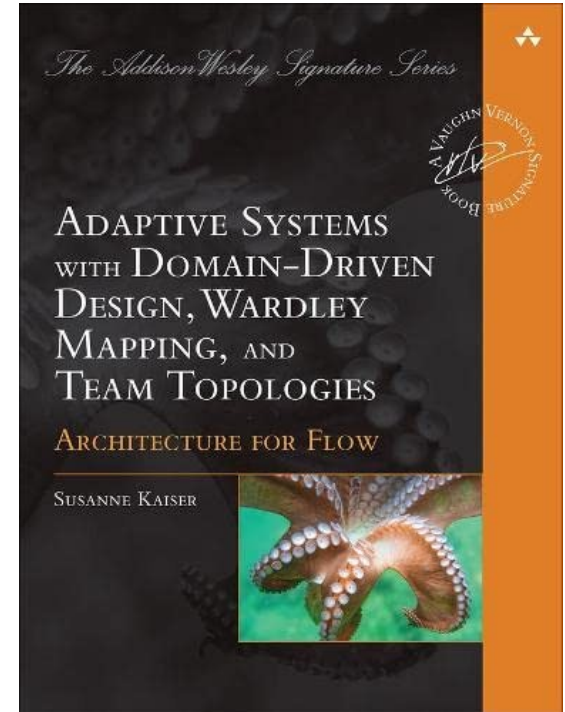
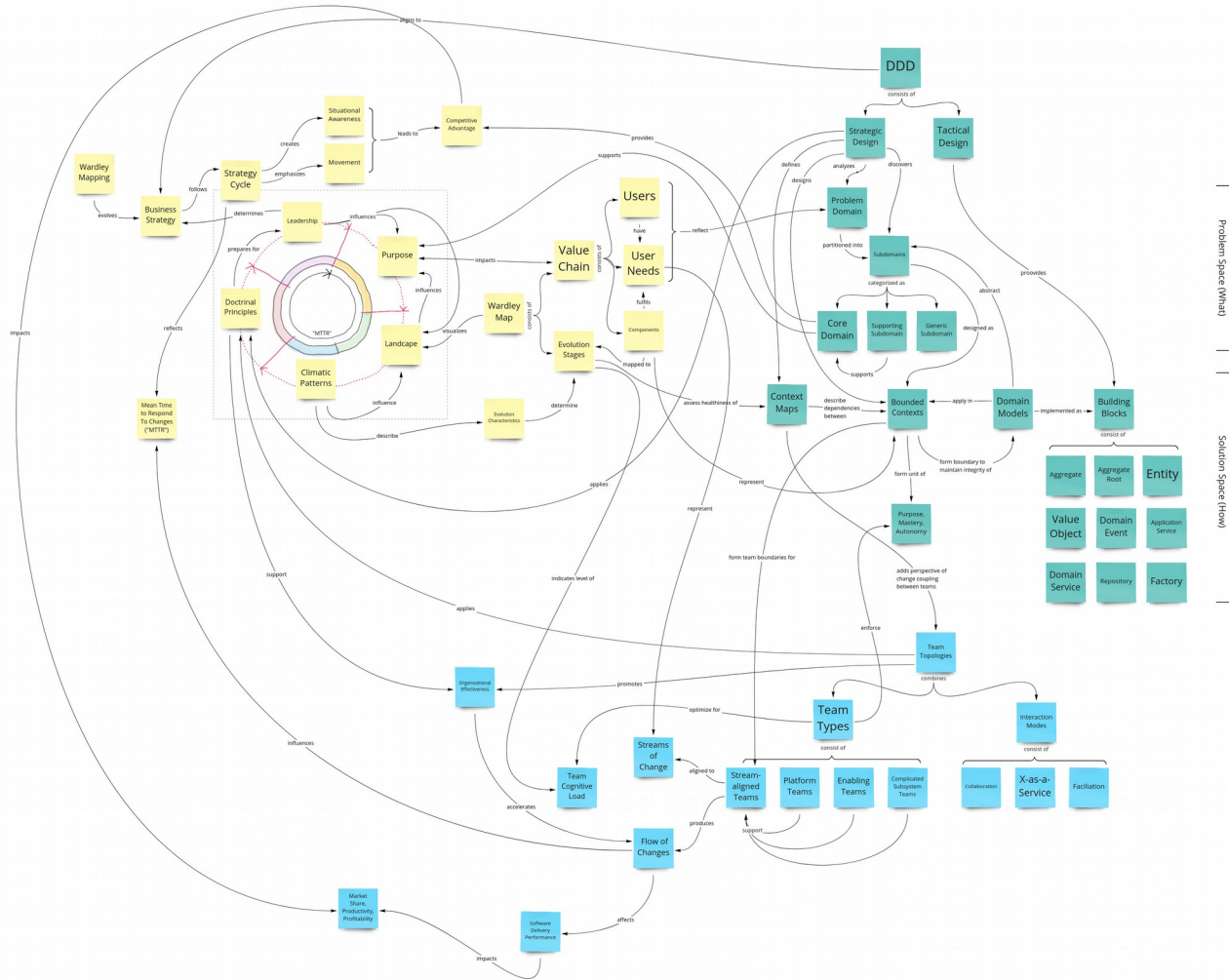
- Understanding the environment an organization is operating & competing in
- Gaining domain knowledge & discovering the core
- Knowing what components to build, buy/use, or outsource
- Decomposing the problem domain into modular bounded contexts
- Aligning teams and evolving their interactions to the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



THANK YOU

Susanne Kaiser  
Independent Tech Consultant  
@suksr  
susanne@susannekaiser.net