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."

"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

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

Building the thing right

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

“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
Their Current Challenges

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

BBoM leads to high cognitive load

Messy model and fuzzy boundaries create high change coupling

High amount of effort to understand a piece of code

No clear ownership boundaries

Delivery bottlenecks impeding software delivery performance

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

Dr. Russell Ackoff

Online School
Search Engine
Data Storage
SMTP Server
Compute
VM
On-premises infrastructure requires high operational effort
Visualizing the Landscape w/ a Wardley Map
The Strategy Cycle of Wardley Mapping

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

Context dependent strategic decisions

Universal applicable principles regardless of landscape

External forces impacting the landscape

Our Why

A map of the competitive environment

Purpose

Doctrine

Landscape

Climate
The Strategy Cycle of Wardley Mapping

- **Purpose**: Universal applicable principles regardless of landscape
- **Doctrine**: External forces impacting the landscape
- **Climate**: Context dependent strategic decisions
- **Landscape**: Our Why
- **Leadership**: A map of the competitive environment

“A strategy cycle is a representation 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.
The Landscape of the Current State

1. Identify users
   - Identify user needs

Value Chain

Visible

- Create Course Content
- Planning Class
- Helping Students
- Evaluating Students
- Signup/Signin
- Studying Courses
- Asking for Help
- Receiving Evaluation Feedback

Invisible
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

Value Chain

Visible

Invisible

Online School

Search Engine

Data Storage

Message Broker

SMTP Server

Compute

VM

Create Course Content
Planning Class
Helping Students
Evaluating Students
Signup/Signin
Studying Courses
Asking for Help
Receiving Evaluation Feedback

Teacher

Student

BBoM
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

Value Chain

Visible

Invisible

Evolution

Genesis
Custom-Built
Product (+rental)
Commodity (+utility)

Create Course Content
Planning Class
Helping Students
Evaluating Students
Signup/Signin
Studying Courses
Asking for Help
Receiving Evaluation Feedback
Teacher
Student
Online School
Search Engine
Data Storage
Message Broker
Compute
SMTP Server
VM

Determine dependencies and position of the components in the value chain

Identify components fulfilling user needs

Asking for Help
Receiving Evaluation Feedback

Identify users

Identify user needs
The Climatic Patterns (extract)

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

Evolution

- Past
- Current
- Future

Value Chain

- Visible
- Invisible

Movement

Supply Competition
Demand Competition

Climate

Everything evolves

Genesis
Custom-Built
Product (+rental)
Commodity (+utility)
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

Uncharted

Genesis

Custom-Built

Product (+rental)

Commodity (+utility)

Evolution

Visible

Value Chain

Teacher

Create Course Content

Planning Class

Helping Students

Evaluating Students

Signup/Signin

Studying Courses

Asking for Help

Receiving Evaluation Feedback

Student

Search Engine

Data Storage

Message Broker

Compute

VM

SMTP Server

Industrialised

@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

Evolution

Visible

Value Chain

Invisible

Genesis

Custom-Built

Product (+rental)

Commodity (+utility)

@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

Value Chain

Evolution

Visible

Invisible

Genesis
Custom-Built
Product (+rental)
Commodity (+utility)

Applying Doctrinal Principles

Universal principles an organization should apply to be able to respond to changes
Applying Doctrinal Principles

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

Use appropriate methods per evolution stage

Visible

Value Chain

Invisible

Evolution

@suksr

Genesis

Custom-Built

Product (+rental)

Commodity (+utility)

Build in-house

Use/buy off-the-shelf product

Outsource to utility suppliers

Agile

Lean

Six Sigma

Applying Doctrinal Principles

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

Use appropriate methods per evolution stage

Visible

Value Chain

Invisible

Evolution

@suksr

Genesis

Custom-Built

Product (+rental)

Commodity (+utility)

Build in-house

Use/buy off-the-shelf product

Outsource to utility suppliers

Agile

Lean

Six Sigma
Applying Doctrinal Principles

Value Chain

Visible

Create Course Content  Planning Class  Helping Students  Evaluating Students  Signup/Signin  Studying Courses  Asking for Help  Receiving Evaluation Feedback

Invisible

Genesis  Custom-Built  Product (+rental)  Commodity (+utility)

Evolution

Use appropriate methods

@suksr

Build in-house / Agile
Use/buy off-the-shelf product / Lean
Outsource to utility suppliers / Six Sigma

Doctrine

Applying Doctrinal Principles

Asking for Help
Receiving Evaluation Feedback

Studying Courses

Signup/Signin

Evaluating Students

Helping Students

Planning Class

Create Course Content

Teacher

Student
Applying Doctrinal Principles

Value Chain

Visible
- Create Course Content
- Planning Class
- Helping Students
- Evaluating Students
- Signup/Signin
- Studying Courses
- Asking for Help
- Receiving Evaluation Feedback

Invisible
- Know the details
- Use appropriate methods

Doctrine

Evolution
- Build in-house / Agile
- Use/buy off-the-shelf product / Lean
- Outsource to utility suppliers / Six Sigma

Build in-house / Agile
Use/buy off-the-shelf product / Lean
Outsource to utility suppliers / Six Sigma

Know your users
Focus on user needs

Know the details

Use a common language
Challenge assumptions

Doctrine

Applying Doctrinal Principles
To optimize for flow of change from a team perspective requires:

- Cross-functional, autonomous teams
- No handover between teams
- Restricting high, ongoing 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

- Platform team
- Stream-aligned team

Fast flow of change
Four Team Types of Team Topologies

- Enabling team
- Platform team
- Stream-aligned team

Fast flow of change
Four Team Types of Team Topologies

- Complicated subsystem team
- Enabling team
- Platform team
- Stream-aligned team

Fast flow of change

@suksr
Four Team Types of Team Topologies

- Complicated subsystem team
- Enabling team
- Platform team
- Stream-aligned team

Increasing autonomy
Reducing cognitive load

Fast flow of change
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

Enabling team

Complicated subsystem team

Collaboration

X-as-a-Service

Facilitating

Promotes organizational effectiveness

Applies Wardley's Doctrinal Principles

Doctrine

@suksr
Architecture For Flow

Finding suitable streams of change

Teacher

Create Course Content
Planning Class
Helping Students
Evaluating Students
Signup/Signin
Studying Courses
Asking for Help
Receiving Evaluation Feedback

Visible

Value Chain

Online School

Search Engine
Data Storage
Message Broker
Compute
SMTP Server
VM

Invisible

Genesis
Custom-Built
Product (+rental)
Commodity (+utility)
Evolution

Data Storage
Message
Broker
Search
Engine
Compute
SMTP
Server
VM

BBoM

Compute
VM

Online School

Content
Receiving
Evaluation
Feedback

Studying
Courses
Asking
for Help

Signup/Signin
Evaluating
Students
Helping
Students
Planning
Course
Create

Evolution

Finding
suitable
streams
of change

@suksr
Understanding the problem domain and partitioning it into subdomains

Value Chain

Visible

Invisible

Architecture For Flow

Teacher

Problem Domain

Student

Finding suitable streams of change

Create Course Content
Planning Class
Helping Students
Evaluating Students
Signup/Signin
Studying Courses
Asking for Help
Receiving Evaluation Feedback

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

Visible

- Strategic Design
- Problem Space
- Analyse & Discover

Invisible

Value Chain

Evolution

Genesis

Custom-Built

Product (+rental)

Commodity (+utility)

Core Domain

Supporting Subdomain

Generic Subdomain

Problem Domain

Strategic Design (Problem Space)
DDD & Wardley Map

Strategic Design (Problem Space)

Problem Domain
- Supporting Subdomain
- Generic Subdomain

Core Domain

Evolution
- Genesis
- Custom-Built
- Product (+rental)
- Commodity (+utility)

Distilling the problem domain & discovering the core domain

Value Chain
- Competitive advantage
  - Complex
  - Changes often
  - Build in-house

Visible
- Strategic Design
  - Problem Space
  - Analyse & Discover

Invisible
- Build in-house
- Use/buy off-the-shelf product
- Outsource to utility suppliers

@suksr
Distilling the problem domain & discovering the core domain

- Competitive advantage: Complex, Changes often, Build in-house
- No competitive advantage: Quite simple, Does not change often, Prefer to buy/use off-the-shelf

Build in-house
Use/buy off-the-shelf product
Outsource to utility suppliers

@suksr
Distilling the problem domain & discovering the core domain

---

**Strategic Design (Problem Space)**

- **Core Domain**
  - Competitive advantage
    - Complex
    - Changes often
    - Build in-house
  - No competitive advantage
    - Quite simple
    - Does not change often
    - Prefer to buy/use off-the-shelf

- **Supporting Subdomain**
  - Problem Domain
  - Generic Subdomain
  - Evolution

---

**Value Chain**

- **Genesis**
  - Build in-house
- **Custom-Built**
  - Use/buy off-the-shelf product
- **Product (+rental)**
  - Outsource to utility suppliers
- **Commodity (+utility)**

---

**Evolution**

@suksr
Architecture For Flow

Find suitable streams of change

Visible

Strategic Design

Problem Space

Problem and Discover

Partioning the Problem Domain into Subdomains

Value Chain

Core Star

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

Partitioning

Course Content
Planning Class
Helping Students
Evaluating Students
Signup/Signin
Studying Courses
Asking for Help
Receiving Evaluation Feedback

Discovering the Core

@suksr
Architecture For Flow

<table>
<thead>
<tr>
<th>Core</th>
<th>Supporting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differentiation</td>
<td>high</td>
</tr>
<tr>
<td>Complexity</td>
<td>high</td>
</tr>
<tr>
<td>Change Rate</td>
<td>high</td>
</tr>
<tr>
<td>Ubiquity</td>
<td>low</td>
</tr>
<tr>
<td>Strategic Investment</td>
<td>high</td>
</tr>
</tbody>
</table>
### Architecture For Flow

- **Teacher**
- **Student**

**Problem Domain**

#### Create Course Content
- **Supporting Students**
- **Helping Students**
- **Evaluating Students**

#### Planning Class
- **Signup/Signin**
- **Studying Courses**

#### Helping Students
- **Asking for Help**
- **Receiving Evaluation Feedback**

---

#### Core

<table>
<thead>
<tr>
<th></th>
<th>Core</th>
<th>Supporting</th>
<th>Generic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differentiation</td>
<td>high</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>Complexity</td>
<td>high</td>
<td>low</td>
<td>medium-high</td>
</tr>
<tr>
<td>Change Rate</td>
<td>high</td>
<td>low-medium</td>
<td>low</td>
</tr>
<tr>
<td>Ubiquity</td>
<td>low</td>
<td>medium</td>
<td>high</td>
</tr>
<tr>
<td>Strategic Investment</td>
<td>high</td>
<td>low-medium</td>
<td>low</td>
</tr>
</tbody>
</table>
Architecture For Flow

Finding suitable streams of change

Visible

Strategic Space

Problem Space

Solution Space

Decompose & Discover

Analyze & Discover

Invisible

Evolution

@suksr

Value Chain

Online School

Decomposing the BBoM into Bounded Contexts

Course Content

Planning Class

Helping Students

Evaluating Students

Signup/Signin

Studying Courses

Asking for Help

Receiving Evaluation Feedback

Partitioning the problem domain into subdomains

Discovering the Core

Genesis

Custom-Built

Product (+rental)

Commodity (+utility)
A bounded context defines where a single domain model can be applied.

A bounded context forms a unit of mastery, purpose, autonomy.

A bounded context can provide different types of boundaries.

A bounded context can provide different types of boundaries per context possible.
Architecture For Flow

- **Finding suitable streams of change**
- **Find suitable team boundaries**
- **Bounded Contexts as team boundaries for stream-aligned teams**

**Value Chain**

- Providing purpose, mastery, autonomy
- Optimize for team cognitive load
- Limit the number, type, size of components per team

**Visible**

- **Create Course Content**
- **Planning Class**
- **Helping Students**
- **Evaluating Students**
- **Signup/Signin**
- **Studying Courses**
- **Asking for Help**
- **Receiving Evaluation Feedback**

**Invisible**

- **Content Creation**
- **Course Studies**
- **Class Mngmnt.**
- **Learning Support**
- **Student Evaluation**
- **Identity & Access**
- **Notification H.**

**Partitioning the problem domain into subdomains**

**Decompose into Bounded Contexts**

**Evolution**

- **Genesis**
- **Custom-Built**
- **Product (+rental)**
- **Commodity (+utility)**
Architecture For Flow

Finding suitable streams of change

Professional Content

Planning Class

Helping Students

Evaluating Students

Signup/Signin

Studying Courses

Asking for Help

Receiving Evaluation Feedback

Partitioning the problem domain into subdomains

Discovering the Core

Decompose into Bounded Contexts

Visible

Finding suitable team boundaries

Bounded Contexts as team boundaries for stream-aligned teams

Value Chain

Providing purpose, mastery, autonomy

Optimize for team cognitive load

Limit the number, type, size of components per team

Create clear ownership boundaries

Evolution

@suksr
Architecture For Flow

Finding suitable streams of change

Find suitable team boundaries

Bounded Contexts as team boundaries for stream-aligned teams

Provider purpose, mastery, autonomy

Limit the number, type, size of components per team

Create clear ownership boundaries

Optimize for team cognitive load

Creation

Course Studies & Scoring

Notification H.

Search Engine

Message Broker

Data Storage

SMTP Server

Compute

VM

Evolution

Aim for Thinnest Viable Platform (TVP)

Identify services needed to support reliable flow of change

Services form platform-as-a-service for stream-aligned teams

Mind dependencies & communication bandwidth between teams

Identify Capability Gaps

Discovering the Core

Partitioning the problem domain into subdomains

Decompose into Bounded Contexts

Digital platform with self-service APIs, tools, services, knowledge, support

Value Chain

Visible

Finding suitable streams of change

Discovering the Core

Partitioning the problem domain into subdomains

Decompose into Bounded Contexts

Mind dependencies & communication bandwidth between teams

Identify Capability Gaps

Evolution

Invisible

Finding suitable team boundaries

Bounded Contexts as team boundaries for stream-aligned teams

Provider purpose, mastery, autonomy

Limit the number, type, size of components per team

Create clear ownership boundaries

Optimize for team cognitive load

Creation

Course Studies & Scoring

Notification H.

Search Engine

Message Broker

Data Storage

SMTP Server

Compute

VM

Evolution

Aim for Thinnest Viable Platform (TVP)

Identify services needed to support reliable flow of change

Services form platform-as-a-service for stream-aligned teams

Mind dependencies & communication bandwidth between teams

Identify Capability Gaps

Discovering the Core

Partitioning the problem domain into subdomains

Decompose into Bounded Contexts

Digital platform with self-service APIs, tools, services, knowledge, support
Architecture For Flow

Value Chain

Visible

Invisible

Teacher

Create Course Content
Planning Class
Helping Students
Evaluating Students
Signup/Signin
Studying Courses
Asking for Help
Receiving Evaluation Feedback

Content Creation
Course Studies
Class Mgmt.
Learning Support
Student Evaluation

Identity & Access

Platform team(s)

Stream-aligned team 1
Stream-aligned team 2
Stream-aligned team 3

Aspect H.

Message Broker
Search E.
Data Storage
SMTP Server
Compute
VM

VM @suksr

A possible team constellation
How to transition?
Implementing Flow Optimization

Backend team

New platform team

Assessing cloud migration strategies

Determining migration path

Determining new platform

Modifying underlying infrastructure

Replatforming

Value Chain

Visible

Invincible

Genesis

Custom-Built

Product (+rental)

Commodity (+utility)

Online School

SEaaS

MBaaS

Elastic Compute

DBaaS

SMTPaaS

BBOM

Custom-Built Product (+rental)

Commodity (+utility)

Implementing Flow Optimization

Backend team

New platform team

Infrastructure team
Implementing Flow Optimization

- Previous platform team
- Remaining infrastructure team
- Merged platform team

Can split later into smaller teams

Shares previously gained cloud knowledge
Implementing Flow Optimization

Online School

Refactoring

Content Creation
Course Studies
Class Mgmt.
Learning Support
Student Evaluation
Identity & Access
Notification H.
Implementing Flow Optimization

Online School

Refactoring

Content Creation
Course Studies
Class Mngmnt.
Learning Support
Student Evaluation
Identity & Access
Notification H.

Backend team

Frontend team

Stream-aligned team 1

Starts the refactoring journey

@suksr
Implementing Flow Optimization

1. **Discover and assess cloud options for the future bounded context**
   - Platform team
   - Stream-aligned team 1
   - Close Collaboration

2. **Occasional on-demand collaboration and help**
   - Platform team
   - Stream-aligned team 1
   - Limited Collaboration / Facilitating

3. **Best practices, standards, tools, APIs to easily consume cloud services**
   - Platform team
   - Stream-aligned team 1
   - X-As-A-Service

4. **Refactoring**
   - UI Team
   - Backend Team
   - Stream-aligned team 1
   - Serverless Compute
   - Elastic Compute
   - SMTPaaS
   - MBaaS
   - SEaaS
   - DBaaS
   - BBoM

Value Chain:
- Genesis
- Custom-Built
- Product (+rental)
- Commodity (+utility)

@suksr
Implementing Flow Optimization

Remaining frontend team

Remaining backend team

Merges into stream-aligned teams

Remaining monolith as one team

Stream-aligned team 2

Stream-aligned team 3

Stream-aligned team 4
Implementing Flow Optimization
What are we leaving behind?

- Functional silo teams w/ handover
- High communication & coordination effort between teams

What are we adopting?

- Team Topologies
  - Restricted communication bandwidth between teams

- Value Chain
  - Visible
  - Invisible

- Functional silos w/ handover
  - High communication & coordination effort between teams

Genesis
Custom-Built
Product (+rental)
Commodity (+utility)

SA team 1
SA team 2
SA team 3
SA team 4
Platform team(s)

IDPaaS
SEaaS
DBaaS
SMTPaaS
Serverless
Compute

What are we leaving behind?

What are we adopting?

@suksr
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

Value Chain

Visible

- UI Team
- Backend Team
- Infrastructure Team
- Online School
- Search Engine
- Data Storage
- SMTP
- Server
- Message Broker
- Compute
- VM

Invisible

- Genesis
- Custom-Built
- Product (+rental)
- Commodity (+utility)

Platform team(s)

- SMTPaaS
- IDPaaS
- SaaS
- MBaaS
- Serverless Compute
- Compute

SA team 1

SA team 2

SA team 3

SA team 4

Restricted communication bandwidth between teams

Clear team ownership boundaries

Restricted communication bandwidth between teams

Clear team ownership boundaries

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

@suksr
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

Team Topologies

Decomposed system w/ clear models & bounded contexts

Restricted communication bandwidth between teams

Optimized team cognitive load

Loosely coupled system

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 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

- Value Chain
  - Visible: UI Team, Backend Team, Infrastructure Team
  - Invisible: Data Storage, Search Engine, Message Broker, Compute, VM

- Genesis
  - Custom-Built
  - Product (+rental)
  - Commodity (+utility)

- Tight change coupling
- High team cognitive load
- Delivery bottlenecks and impeded software delivery performance
- High operational efforts due to management of on-premises infrastructure

- Loosely coupled system
- Optimized team cognitive load
- Eliminating delivery bottlenecks and increasing software delivery performance
- Offloading management of infrastructure components to cloud providers
Key Takeaways

- 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

- 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

- 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
THANK YOU

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