

Designing a Serverless Application with Domain Driven Design

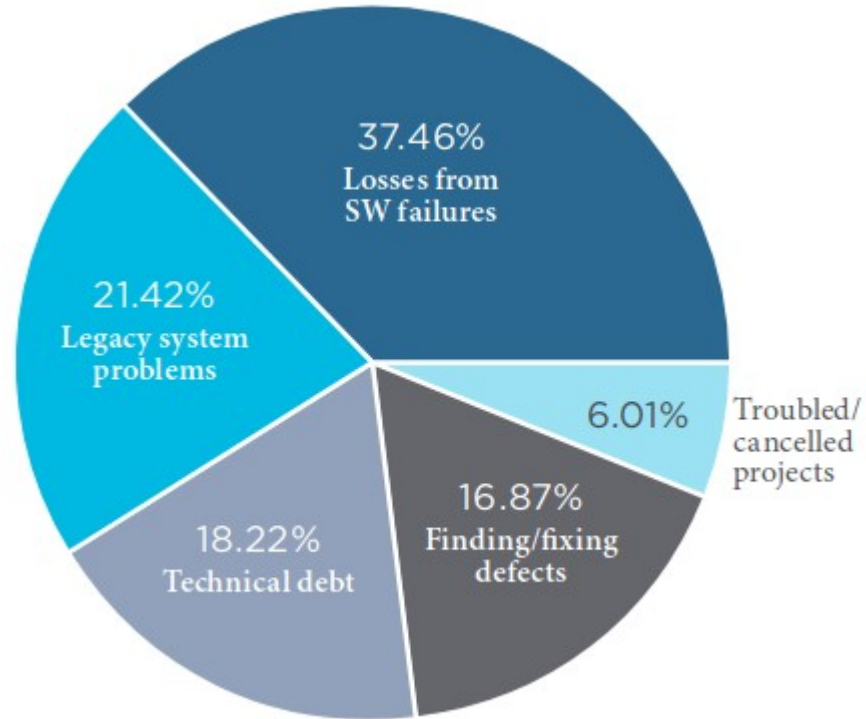
Susanne Kaiser
Independent Tech Consultant
@suksr

Costs of Poor Software Quality in the US in 2018 (by CISQ report)

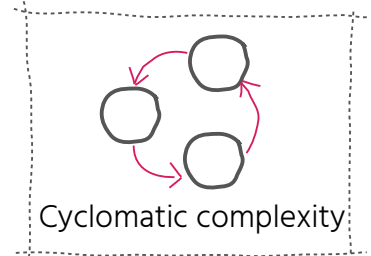
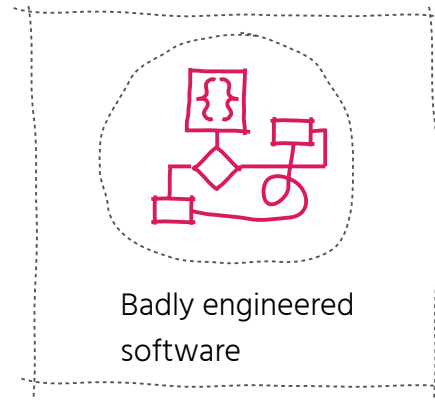
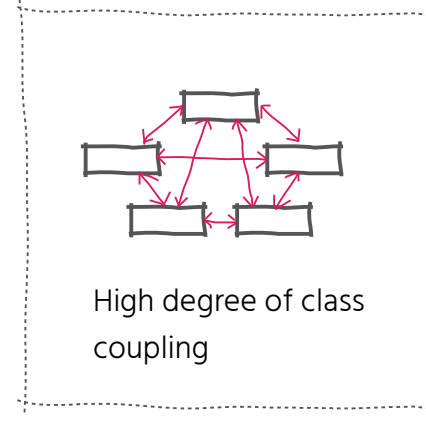
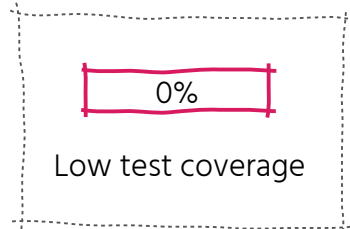
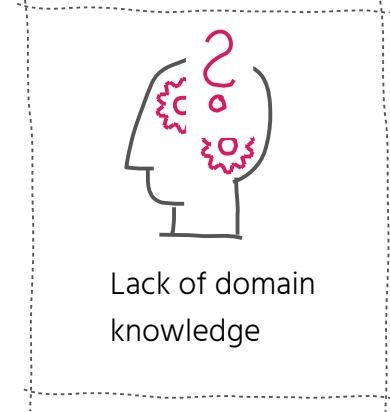
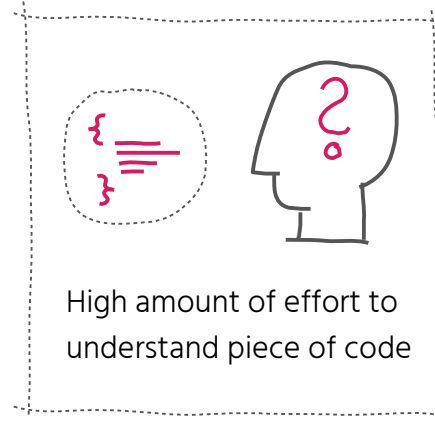
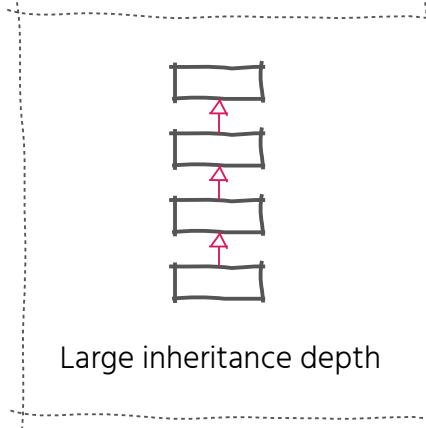
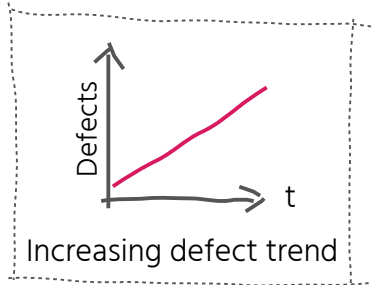
\$2,840,000,000,000

TWOTRILLIONEIGHTHUNDREDFOURTYBILLION USD

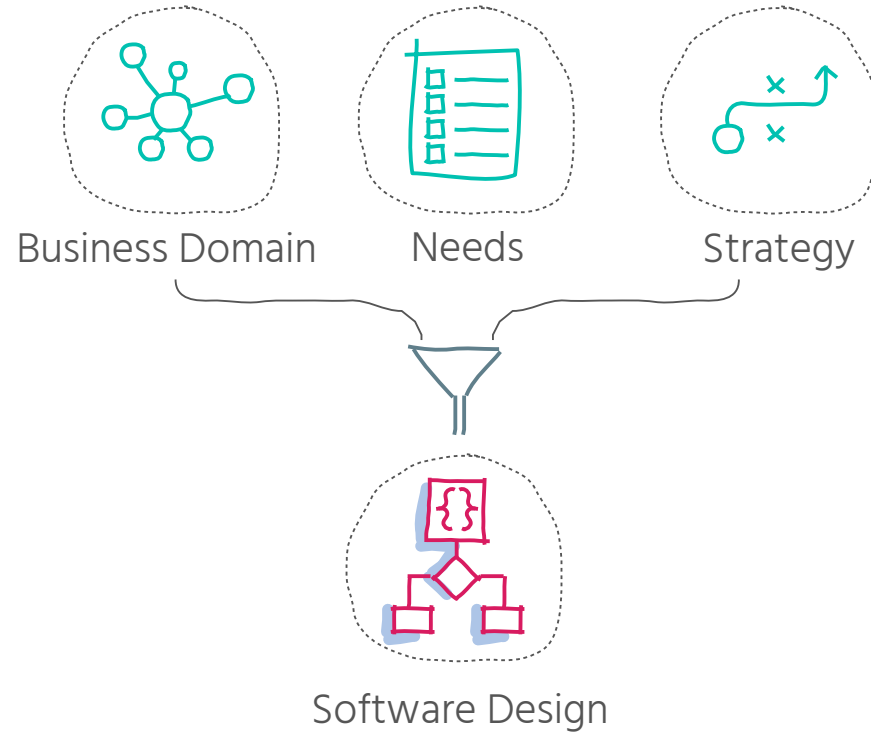
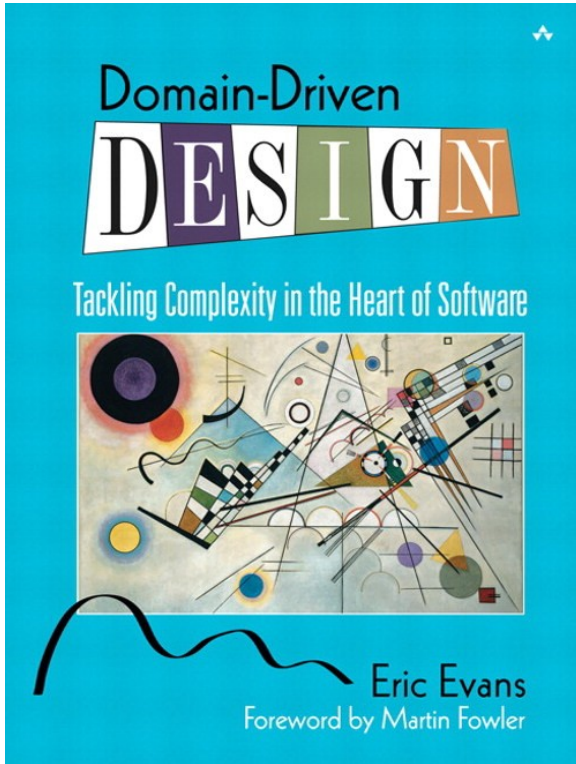
Areas of Cost Relating To Poor Software Quality



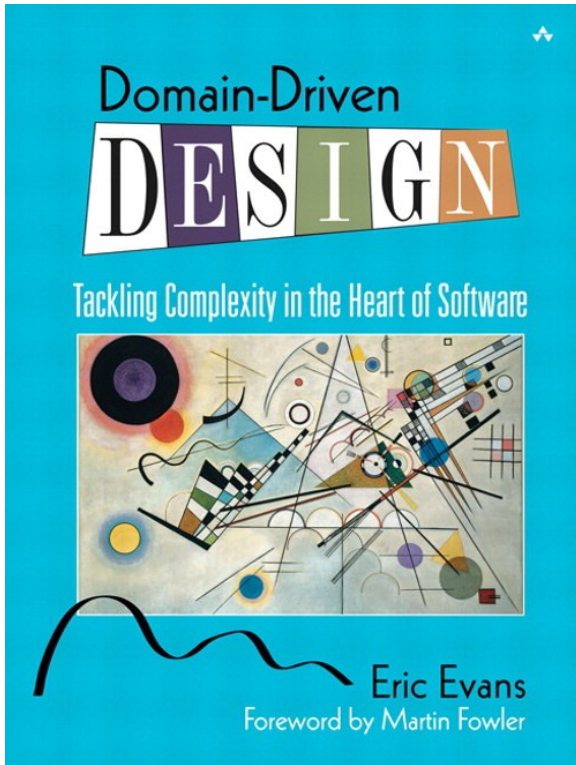
Some Indicators for Poor Software Quality (extracted from CISQ report)



Domain Driven Design (DDD)



Domain Driven Design (DDD) – Terminology



Strategic Design
Tactical Design

Core Subdomain
Supporting Subdomain
Generic Subdomain

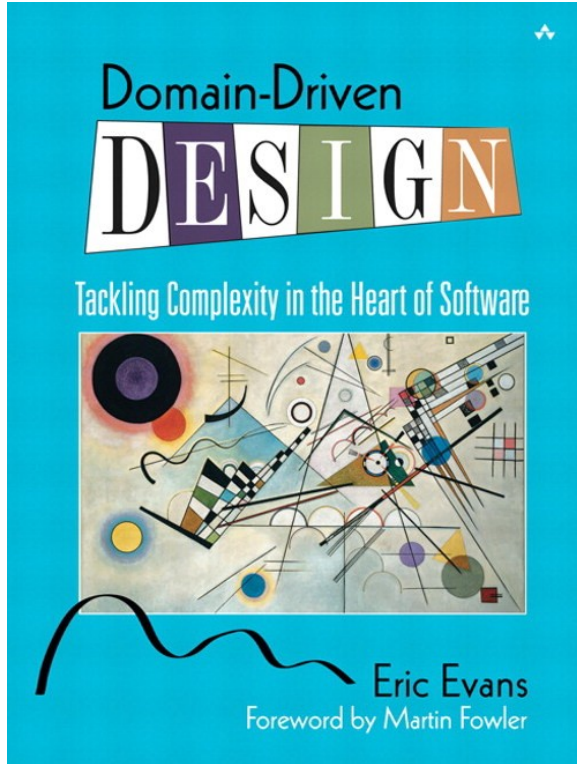
Context Maps
Anti-Corruption Layer
Shared Kernel
Open Host Service
Separate Ways
Partnership
Customer-Supplier
Conformist

Problem Space
Solution Space

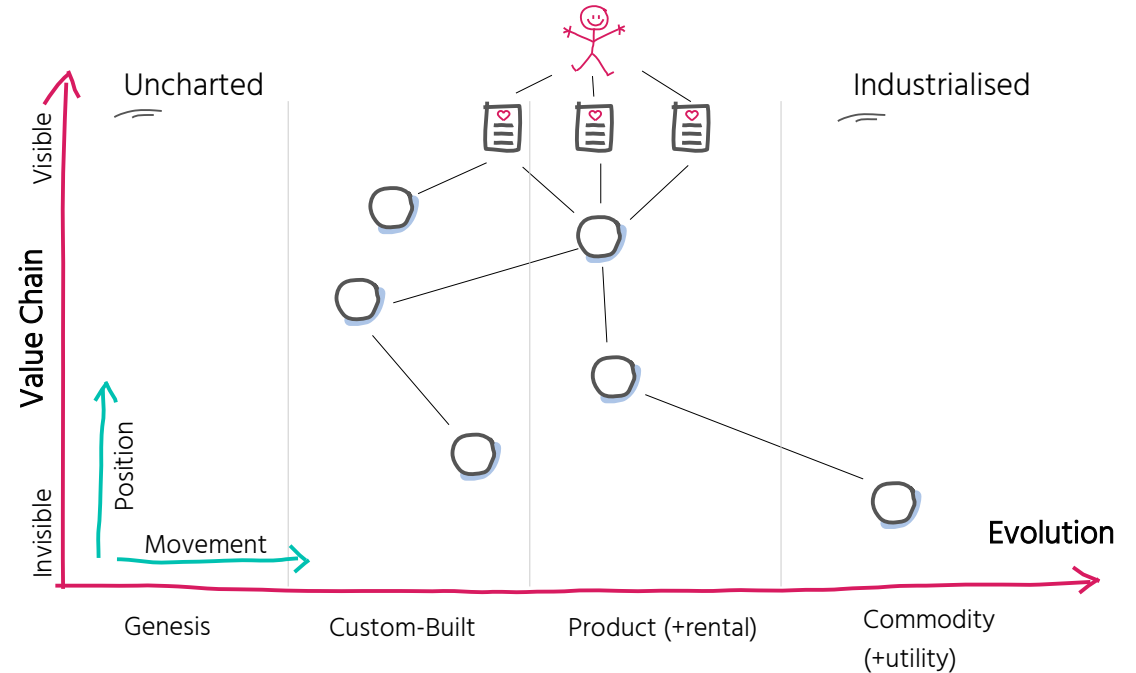
Bounded Context
Ubiquitous Language

Domain Model
Entity
Value Object
Aggregate
Repository
Factory
Application Service
Domain Service
Domain Event

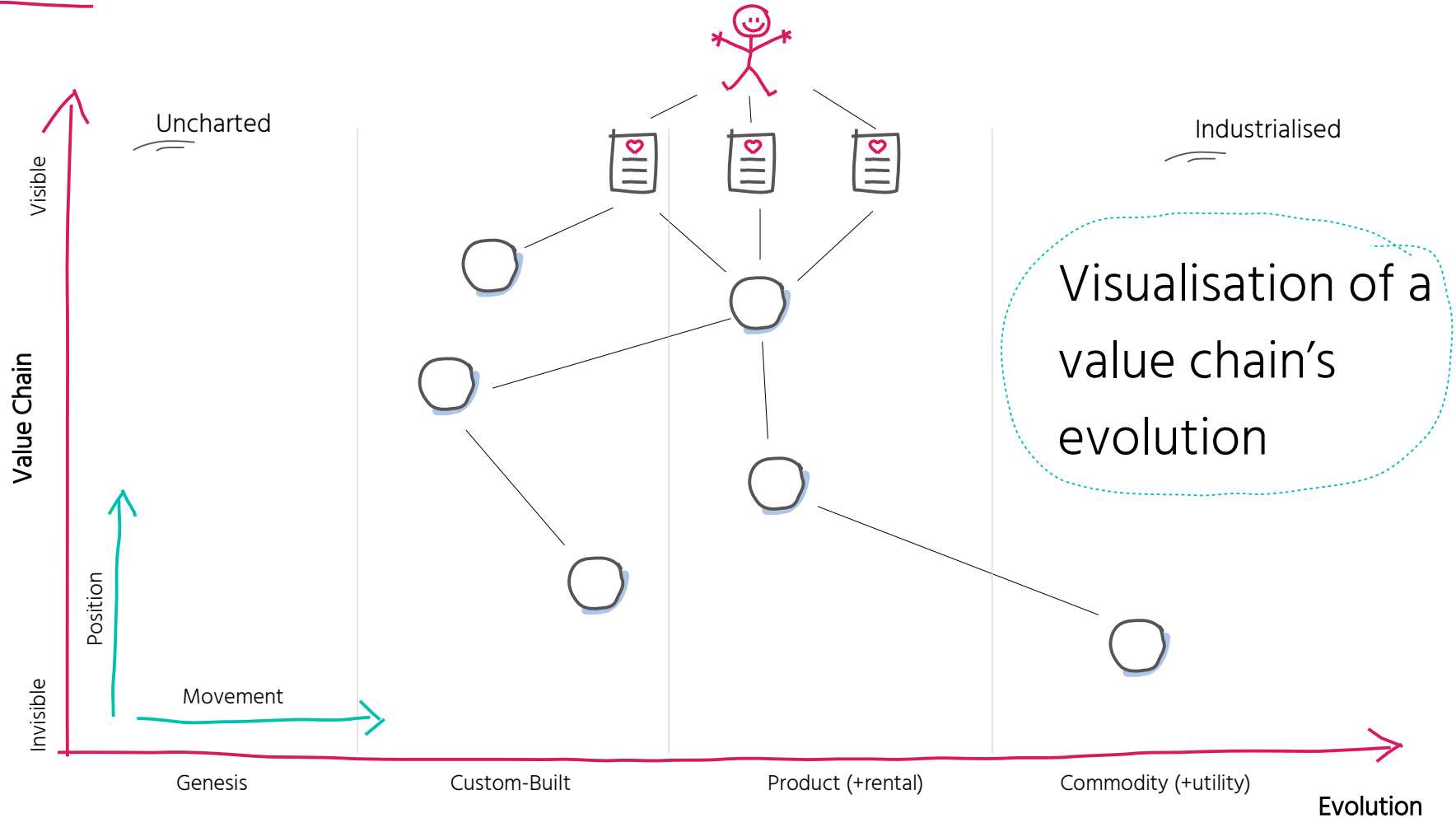
DDD & Wardley Maps



&

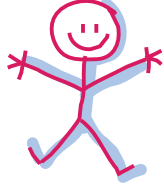


Wardley Maps BY SIMON WARDLEY



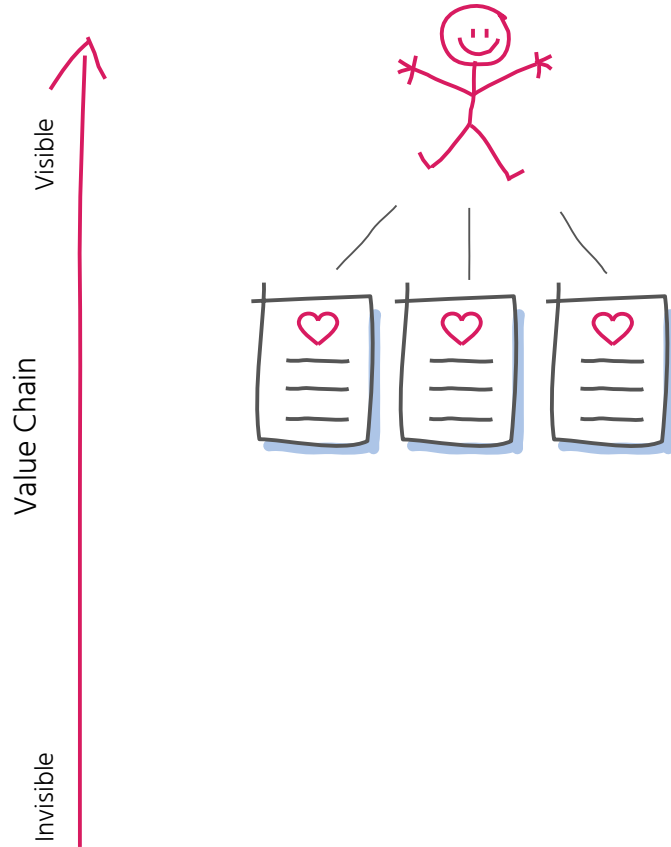
Wardley Maps – VALUE CHAIN

Visible
Value Chain
Invisible



Who are your users?

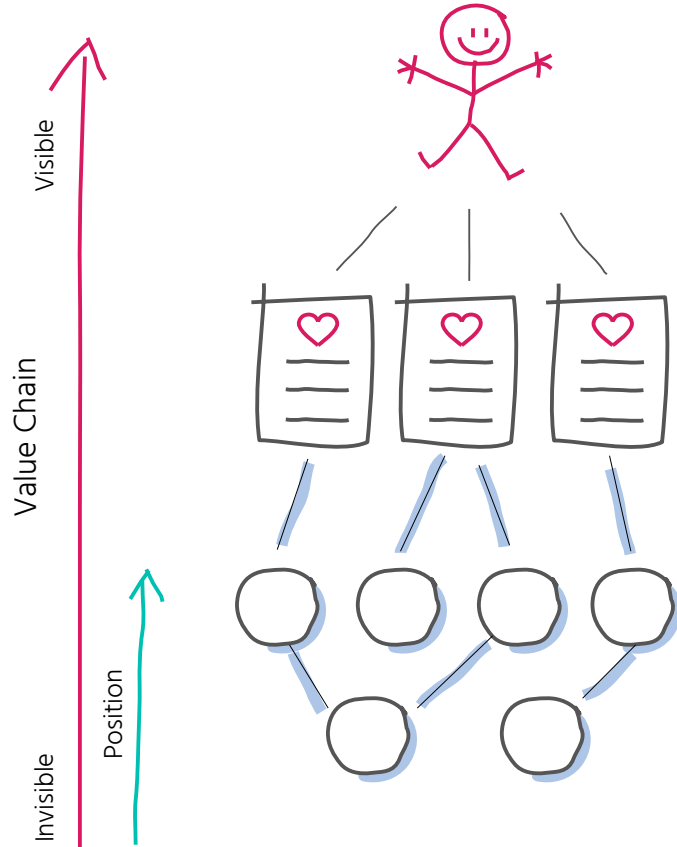
Wardley Maps – VALUE CHAIN



Who are your users?

What are your users' needs?

Wardley Maps – VALUE CHAIN

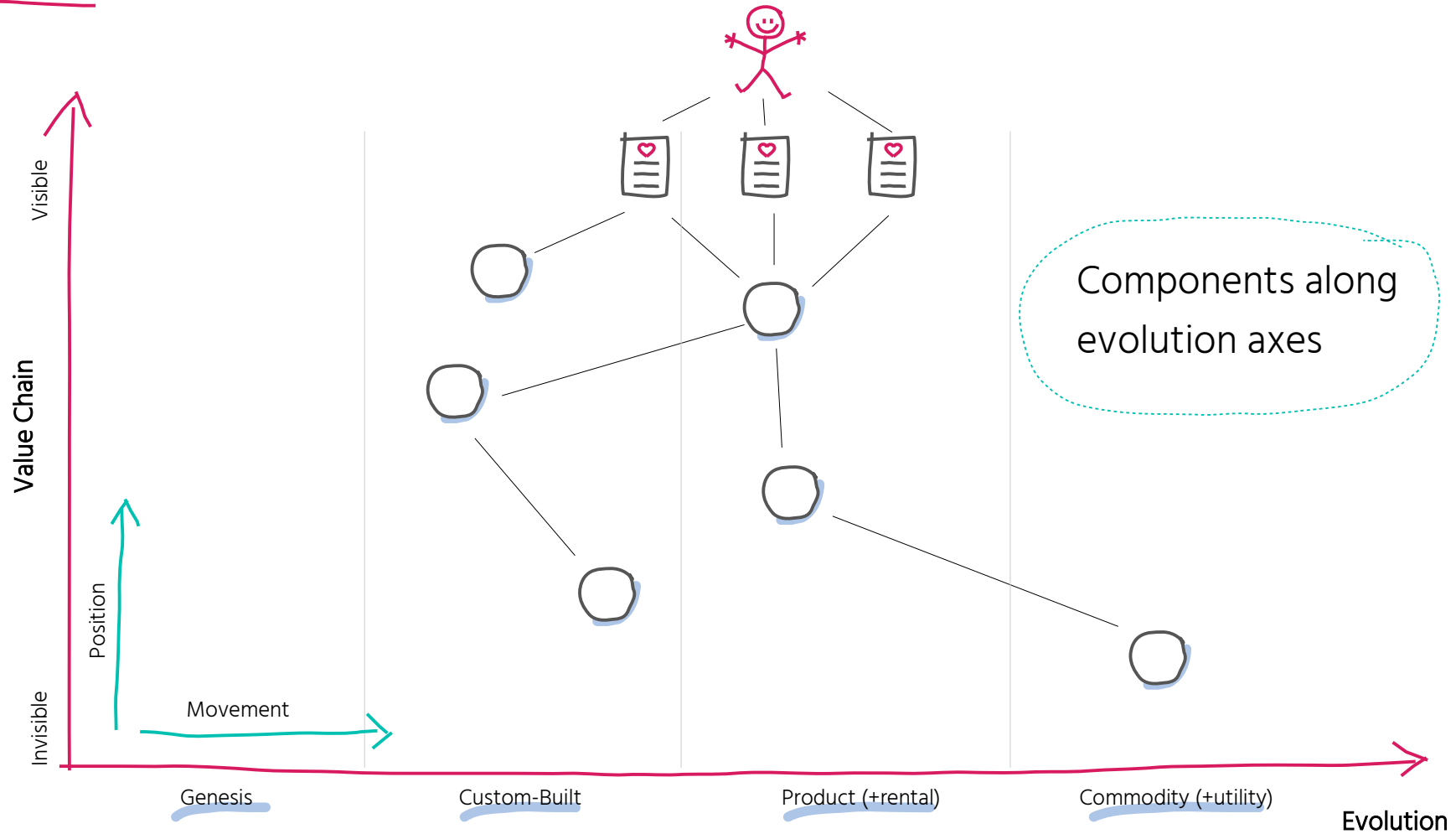


Who are your users?

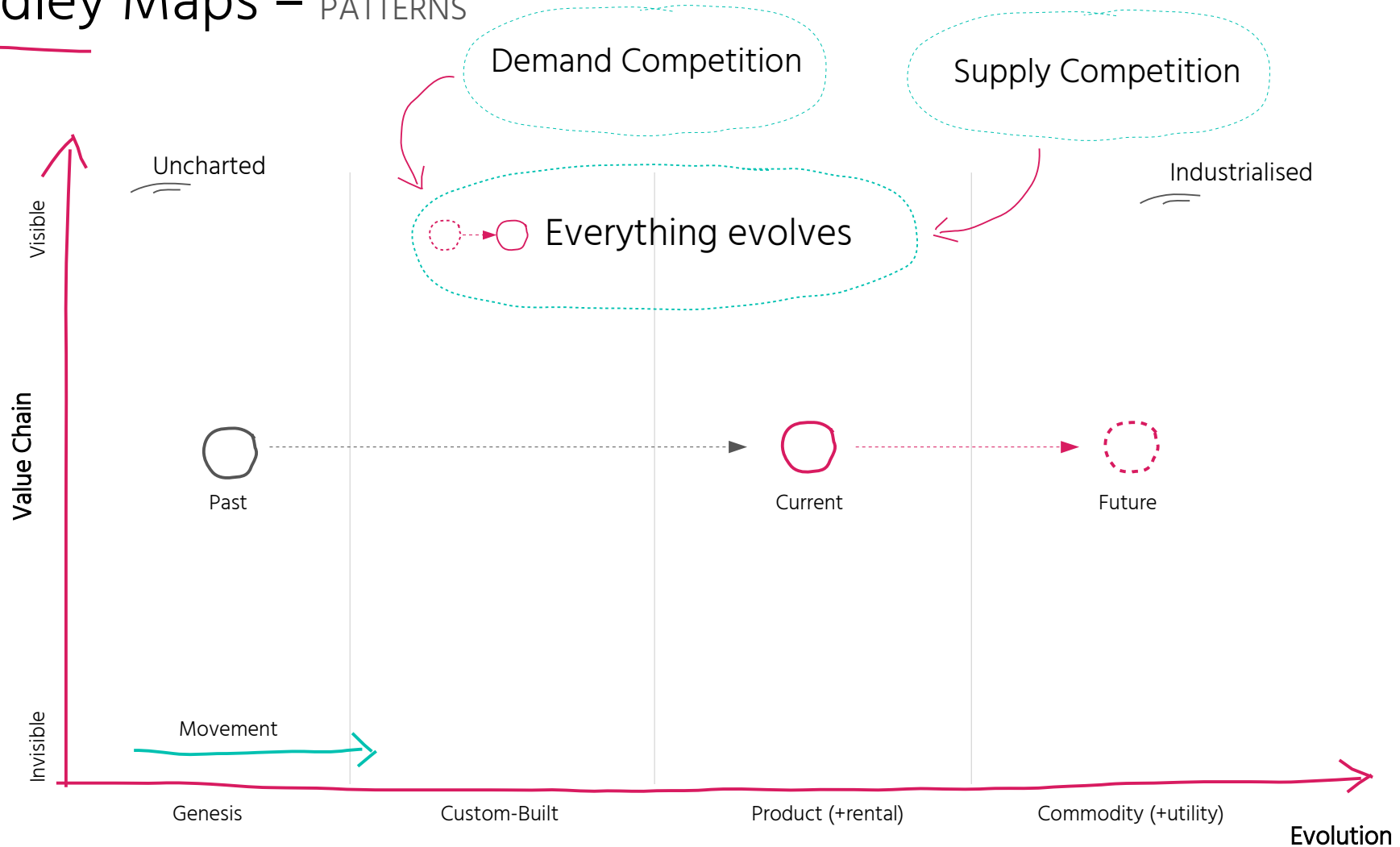
What are your users' needs?

What are the components/activities to fulfill your users' needs incl. dependencies?

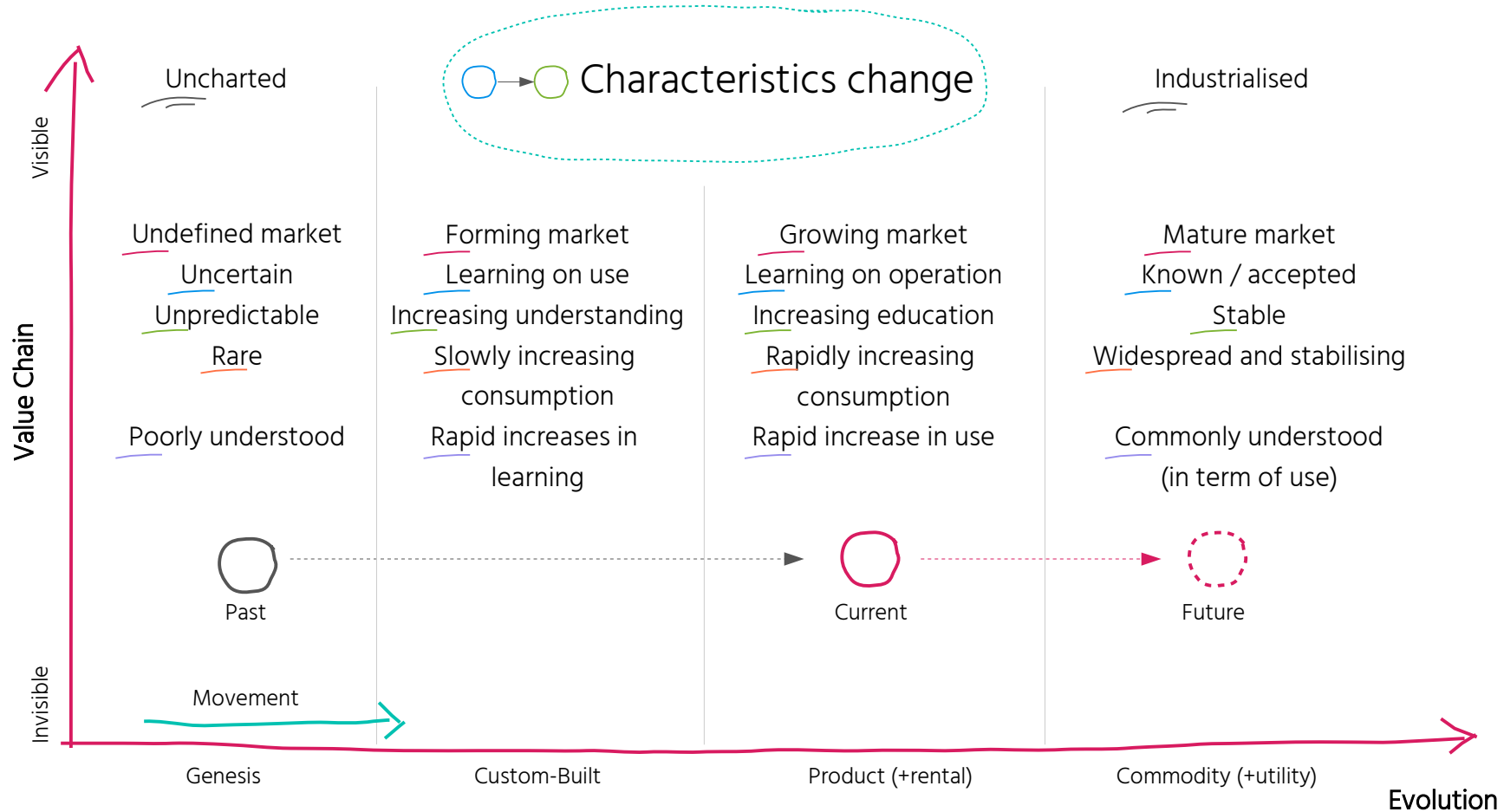
Wardley Maps – LANDSCAPE



Wardley Maps – PATTERNS

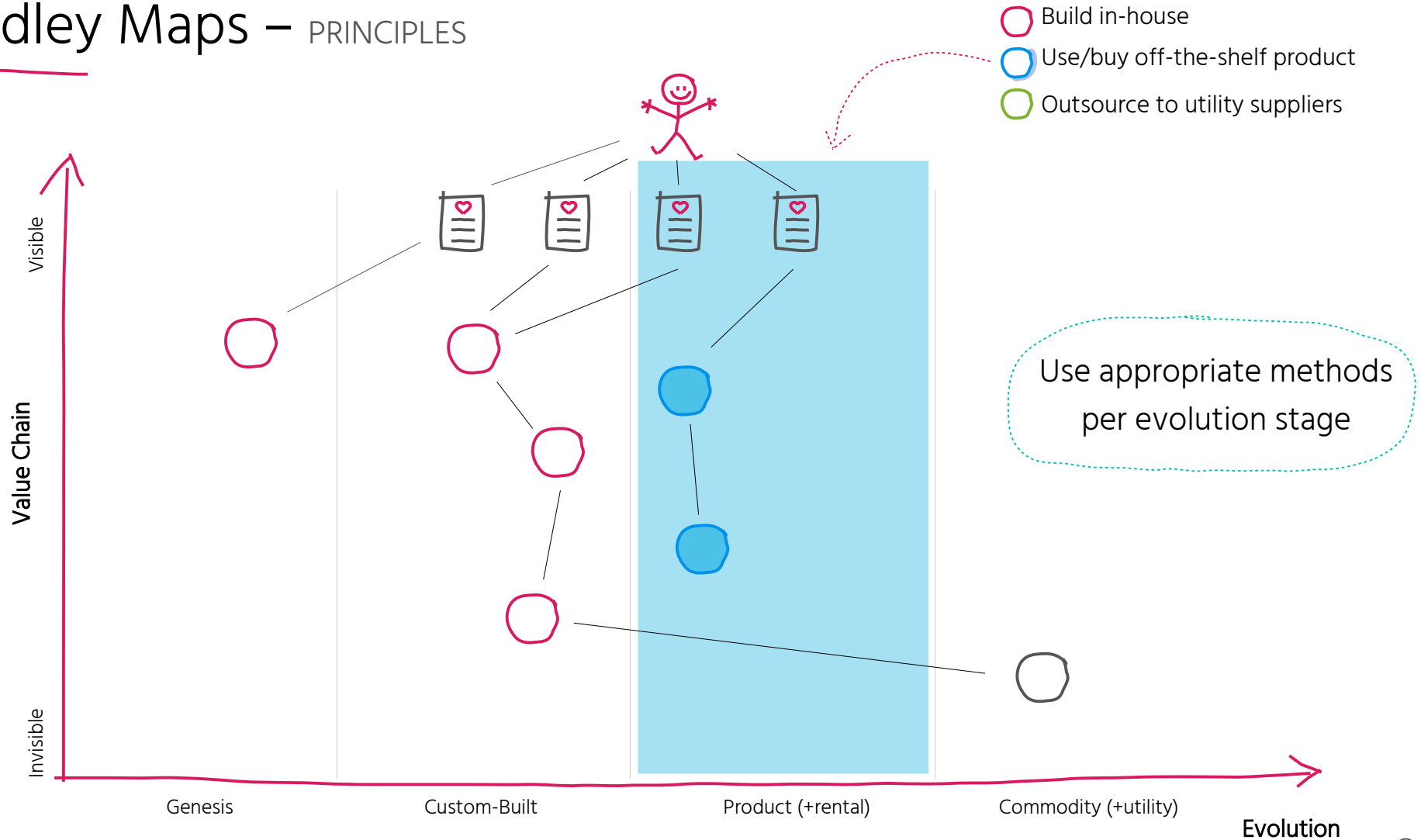


Wardley Maps – PATTERNS

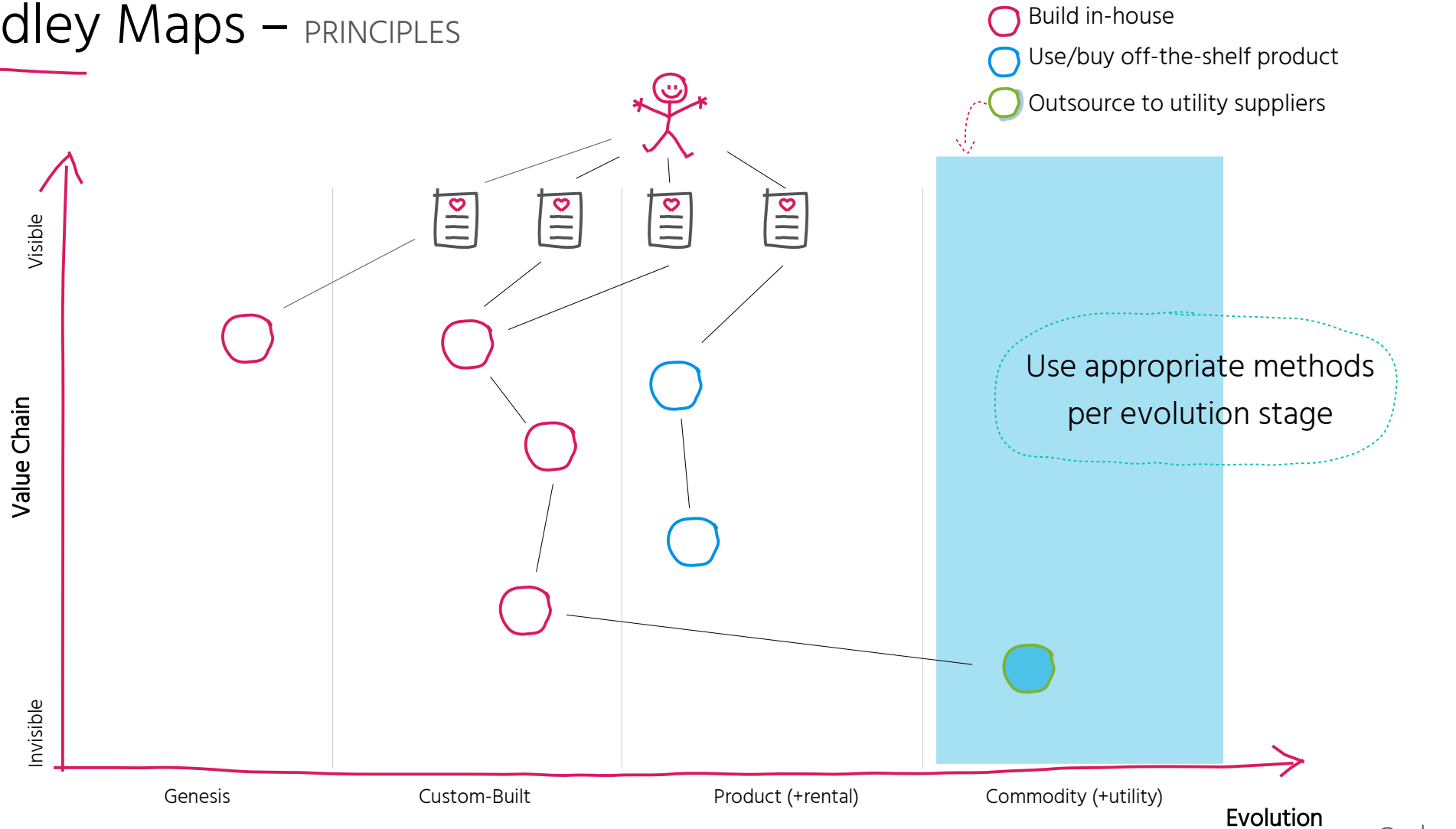




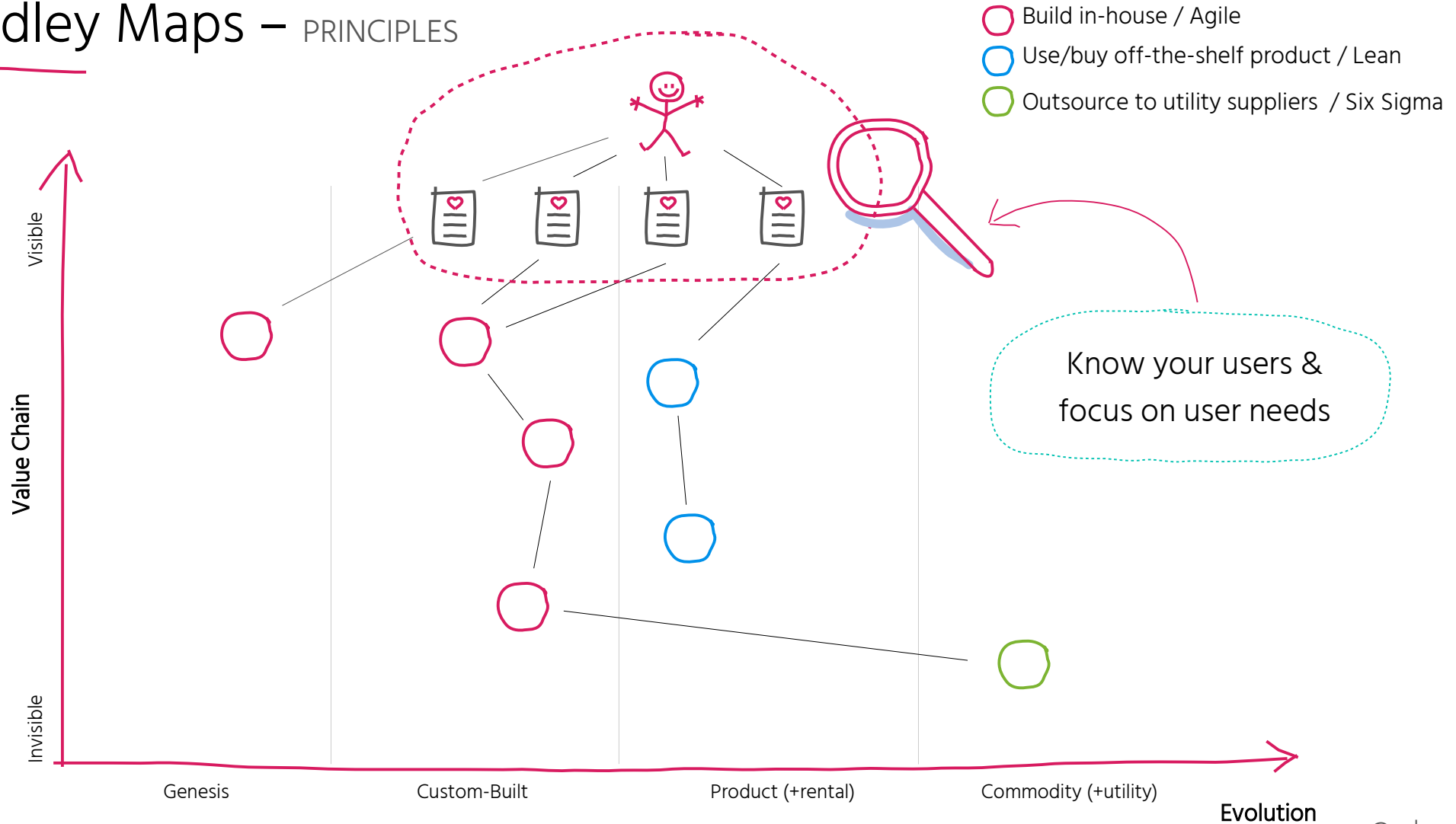
Wardley Maps – PRINCIPLES



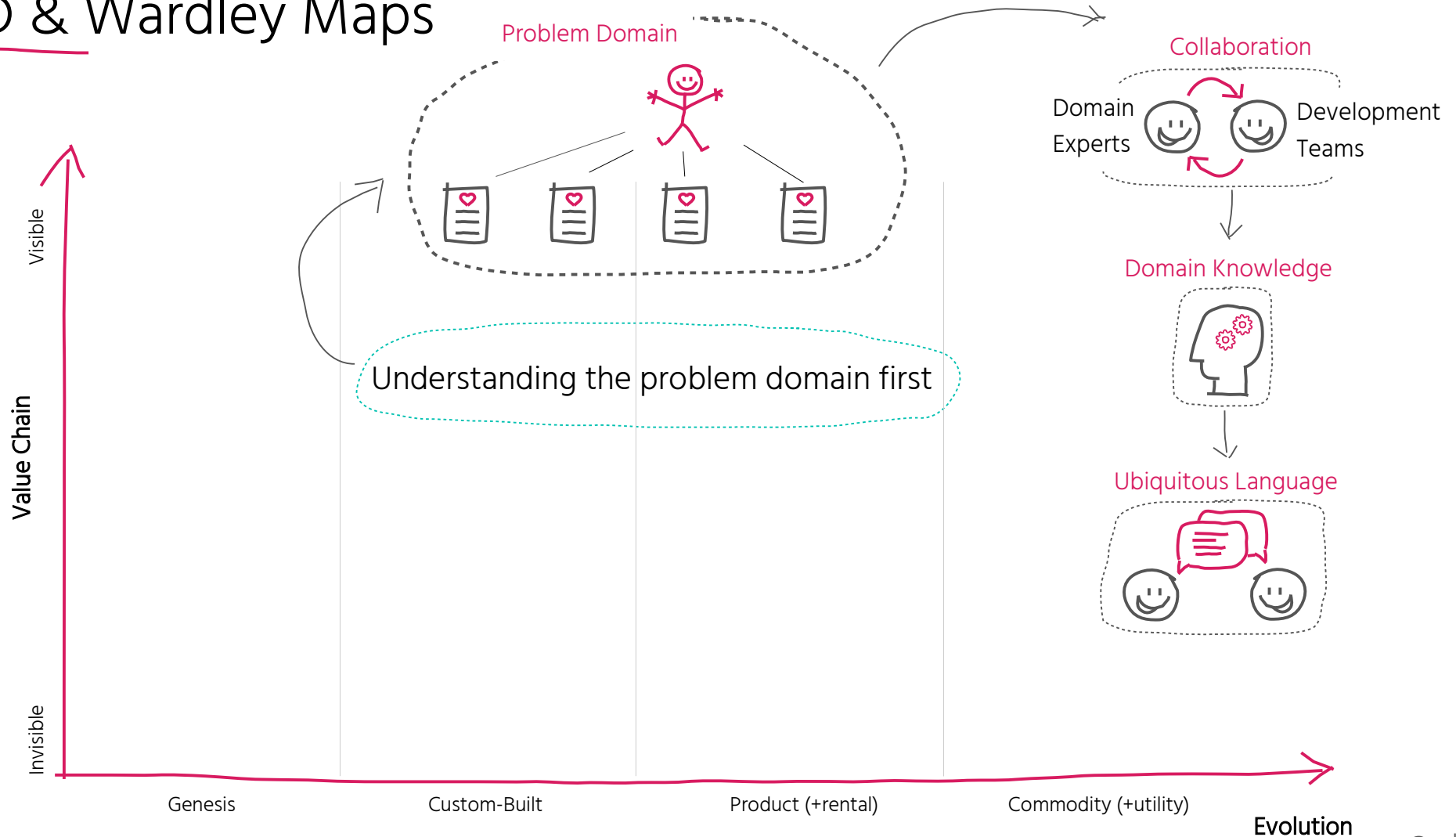
Wardley Maps – PRINCIPLES



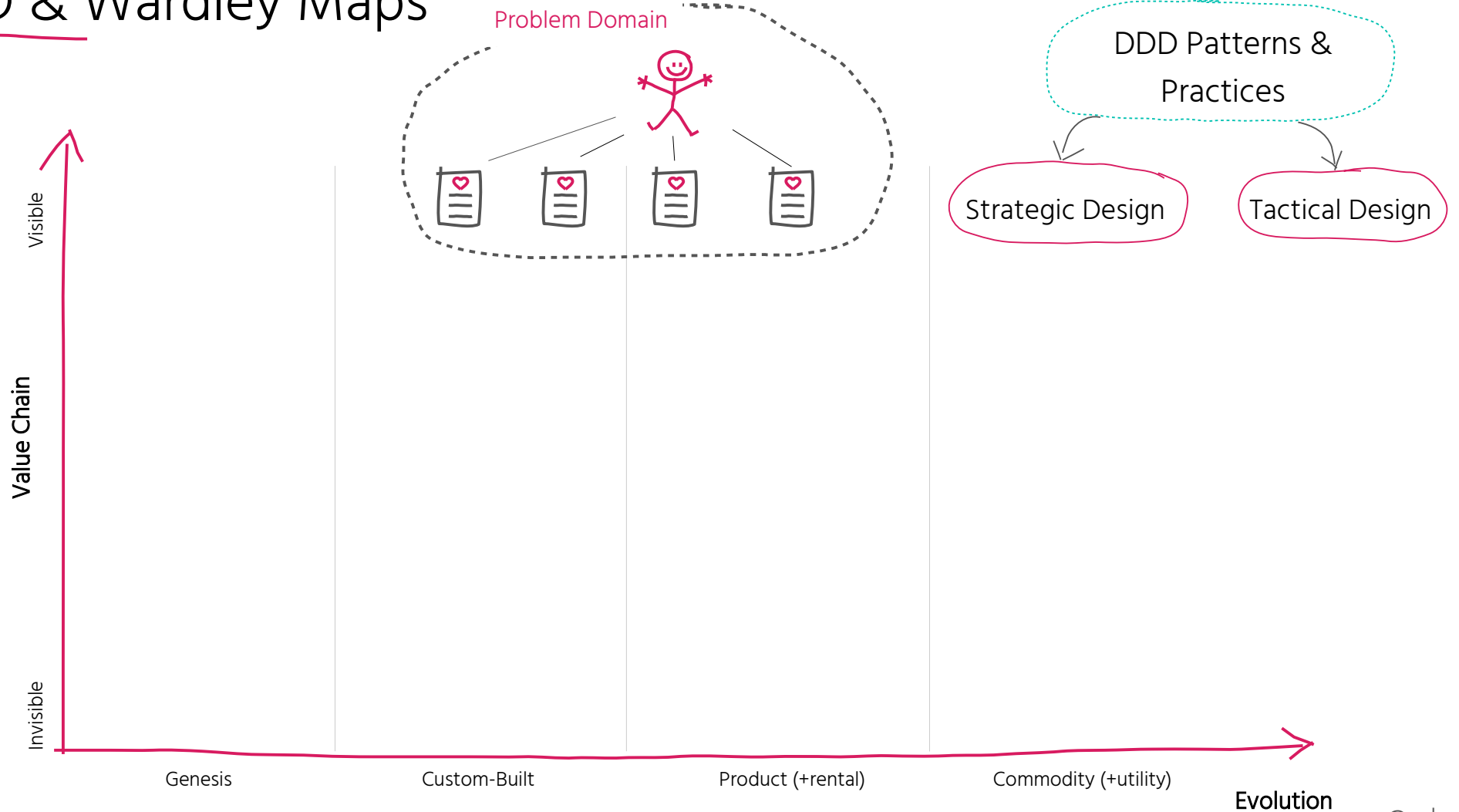
Wardley Maps – PRINCIPLES



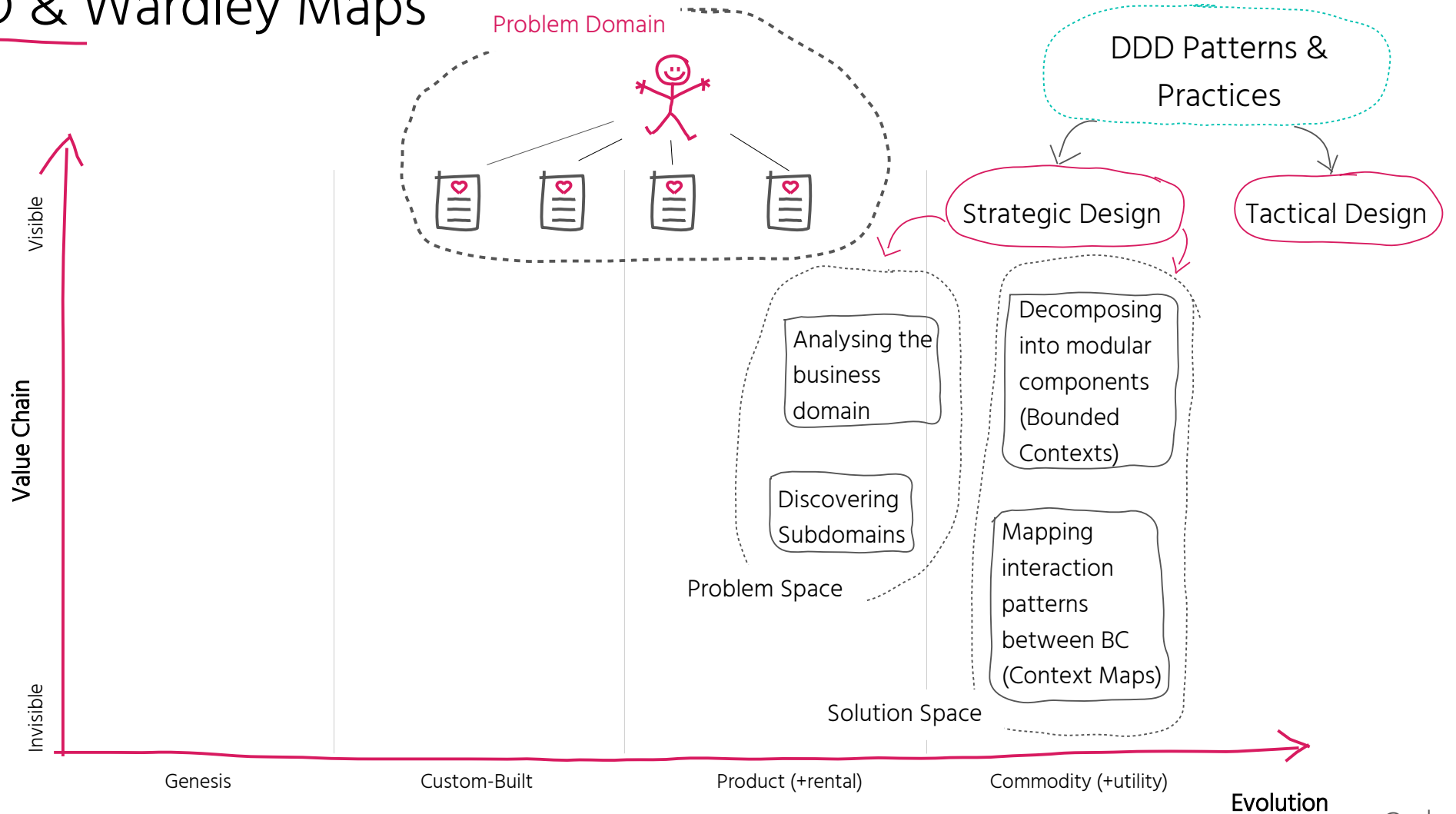
DDD & Wardley Maps



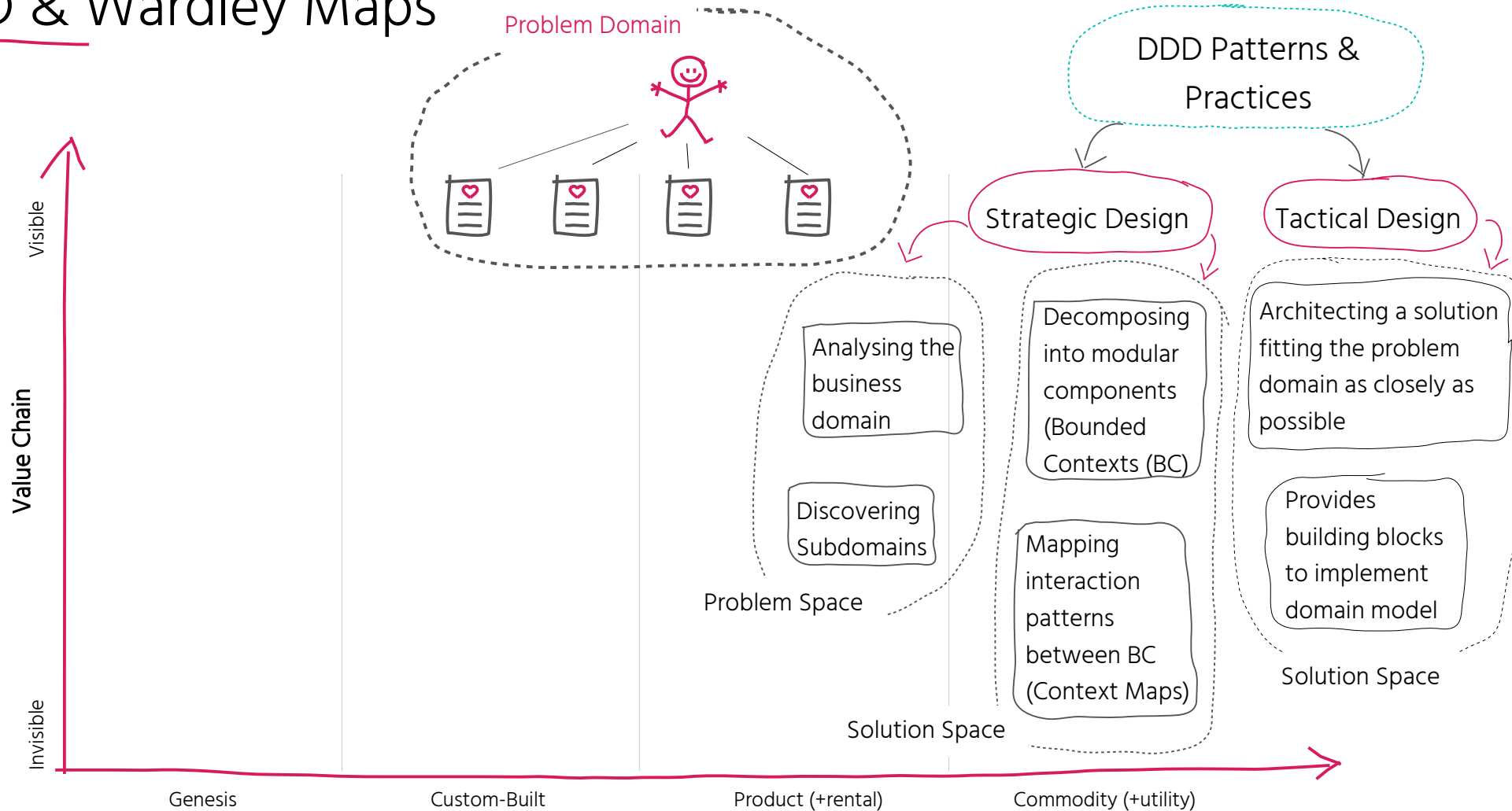
DDD & Wardley Maps



DDD & Wardley Maps

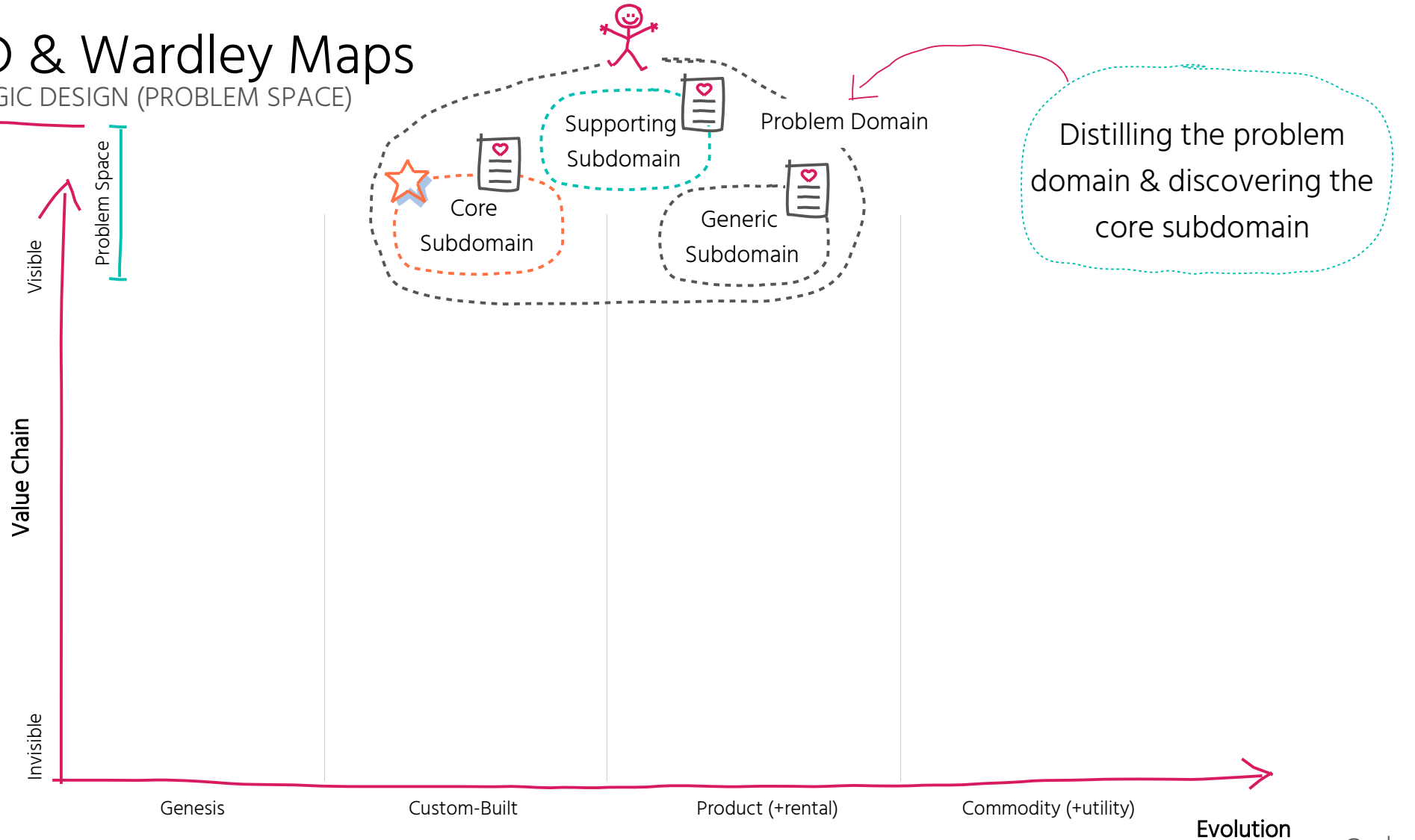


DDD & Wardley Maps



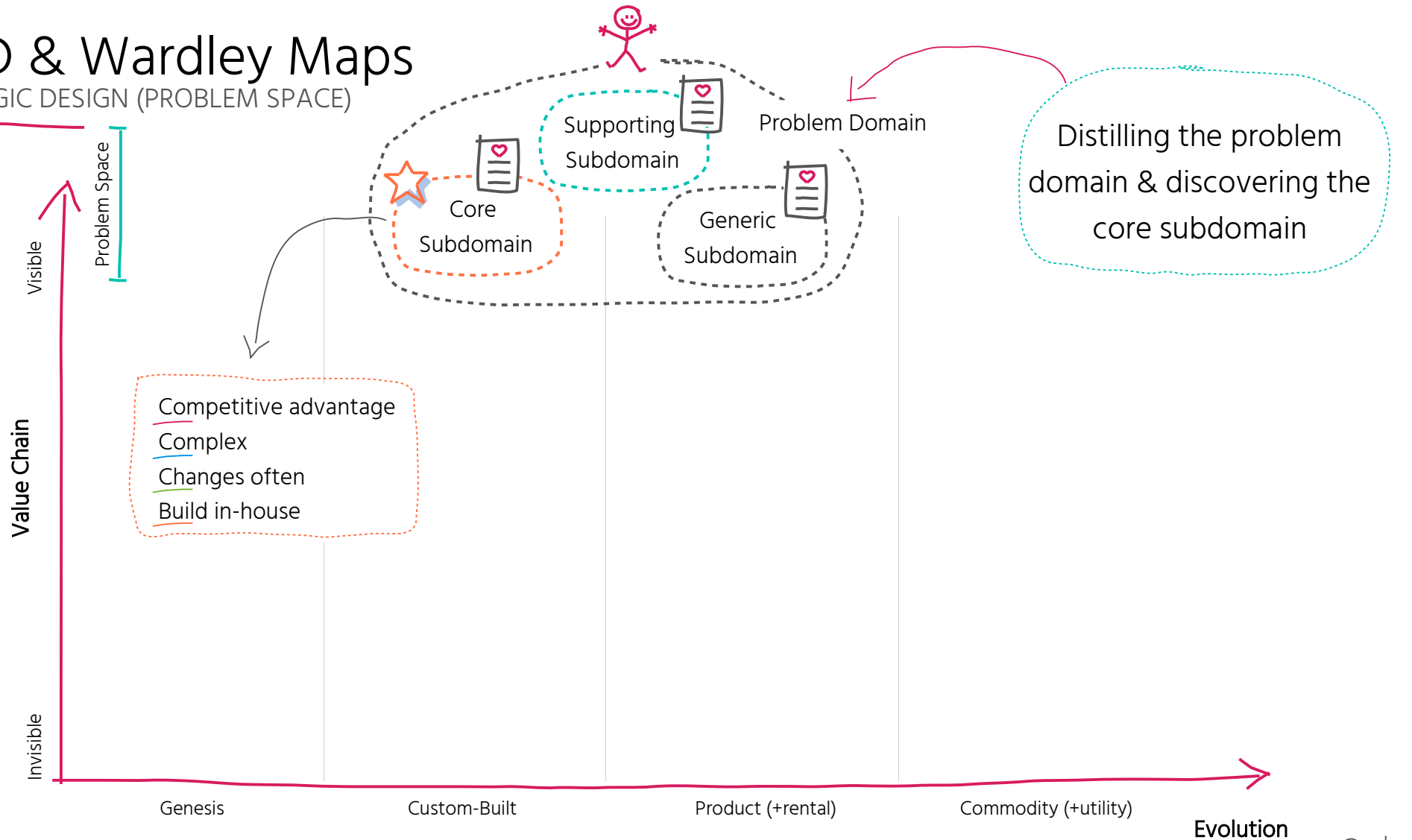
DDD & Wardley Maps

STRATEGIC DESIGN (PROBLEM SPACE)



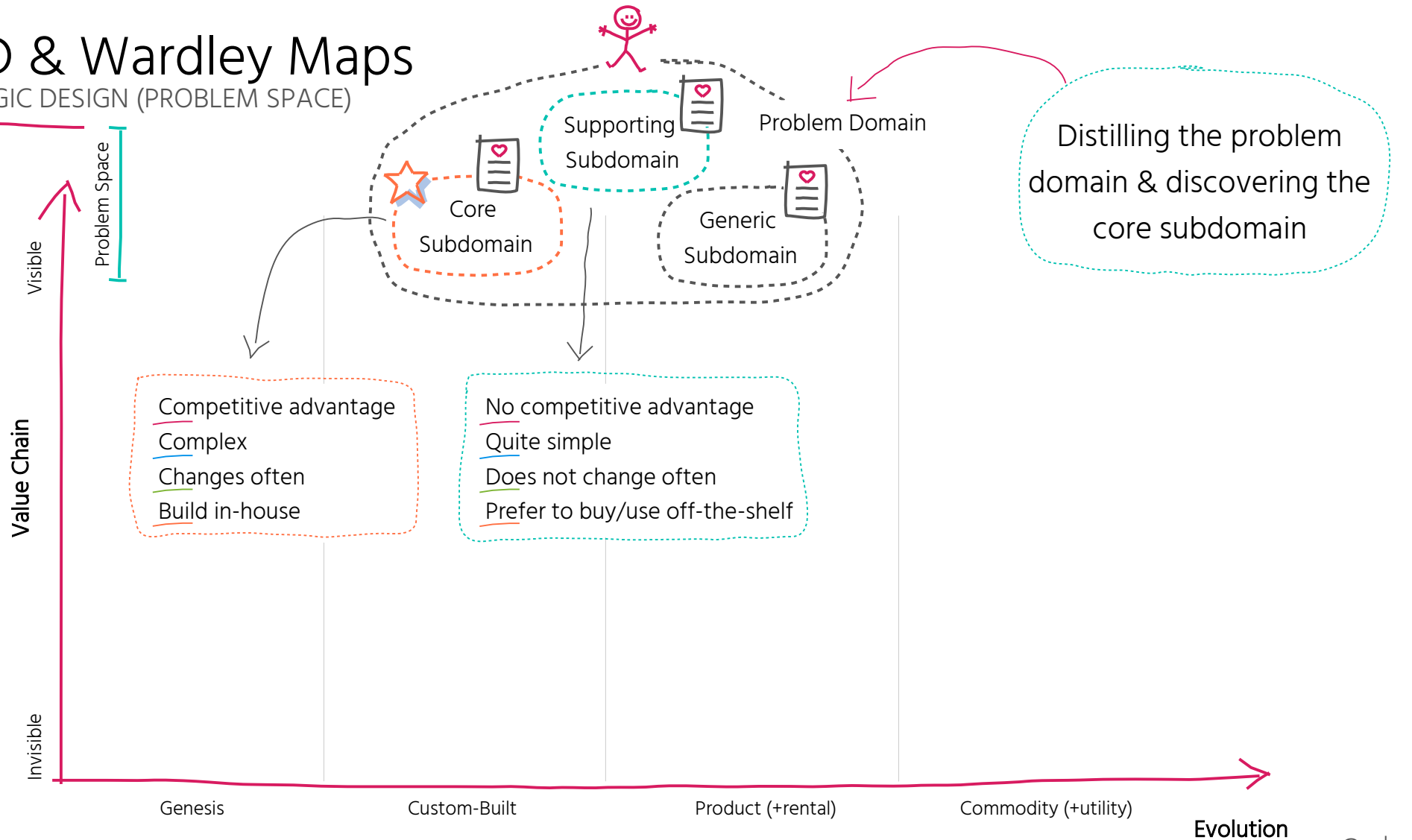
DDD & Wardley Maps

STRATEGIC DESIGN (PROBLEM SPACE)



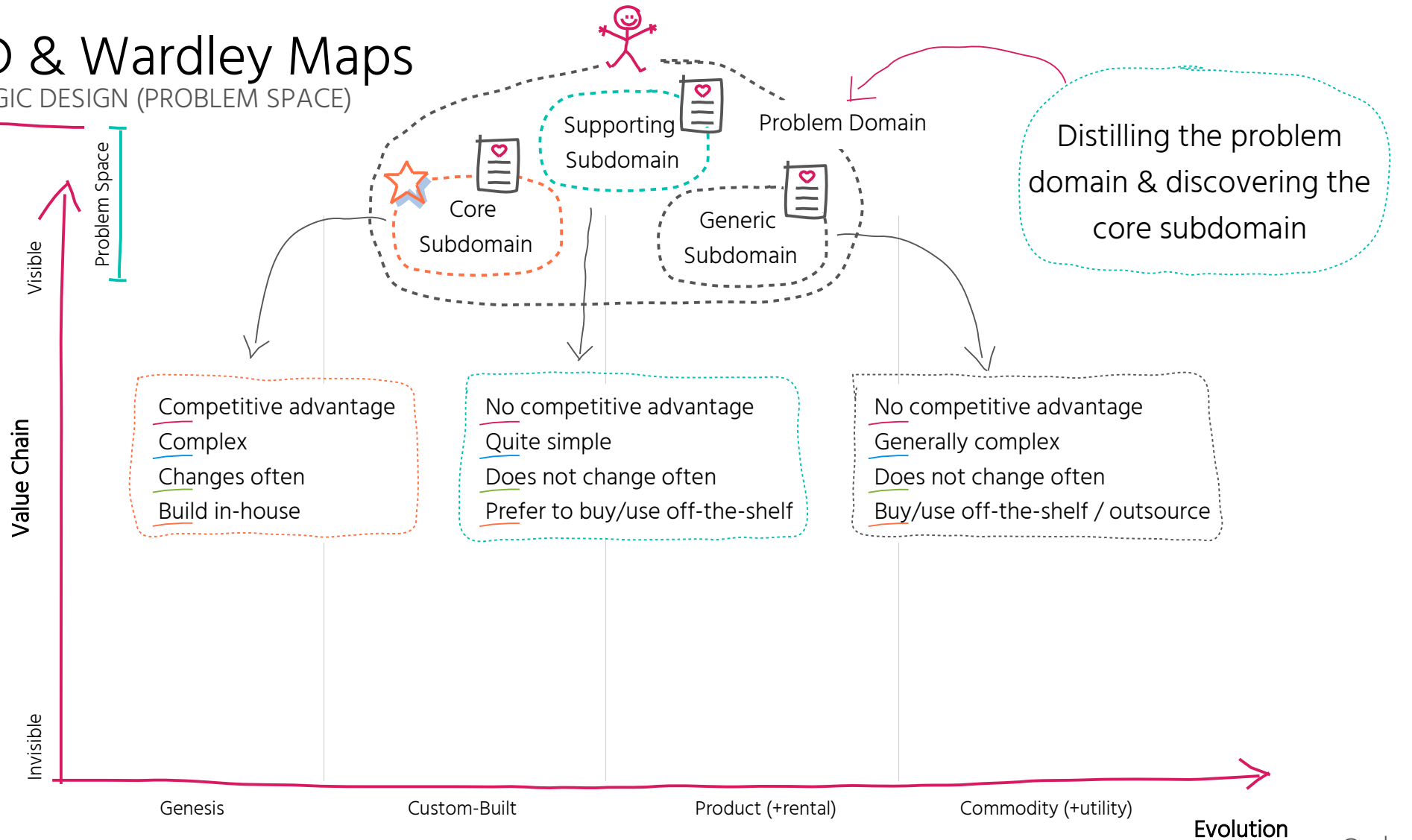
DDD & Wardley Maps

STRATEGIC DESIGN (PROBLEM SPACE)



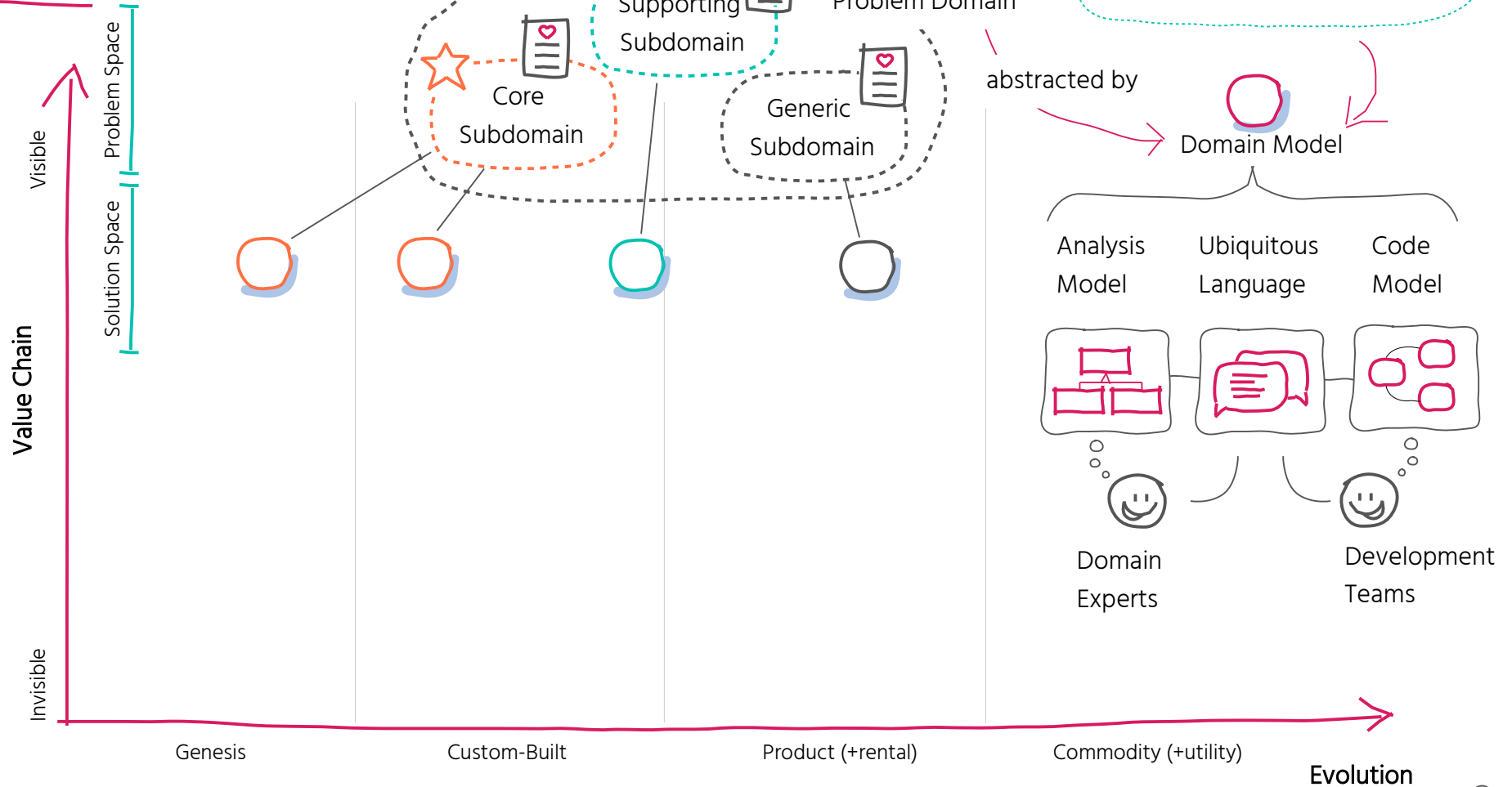
DDD & Wardley Maps

STRATEGIC DESIGN (PROBLEM SPACE)



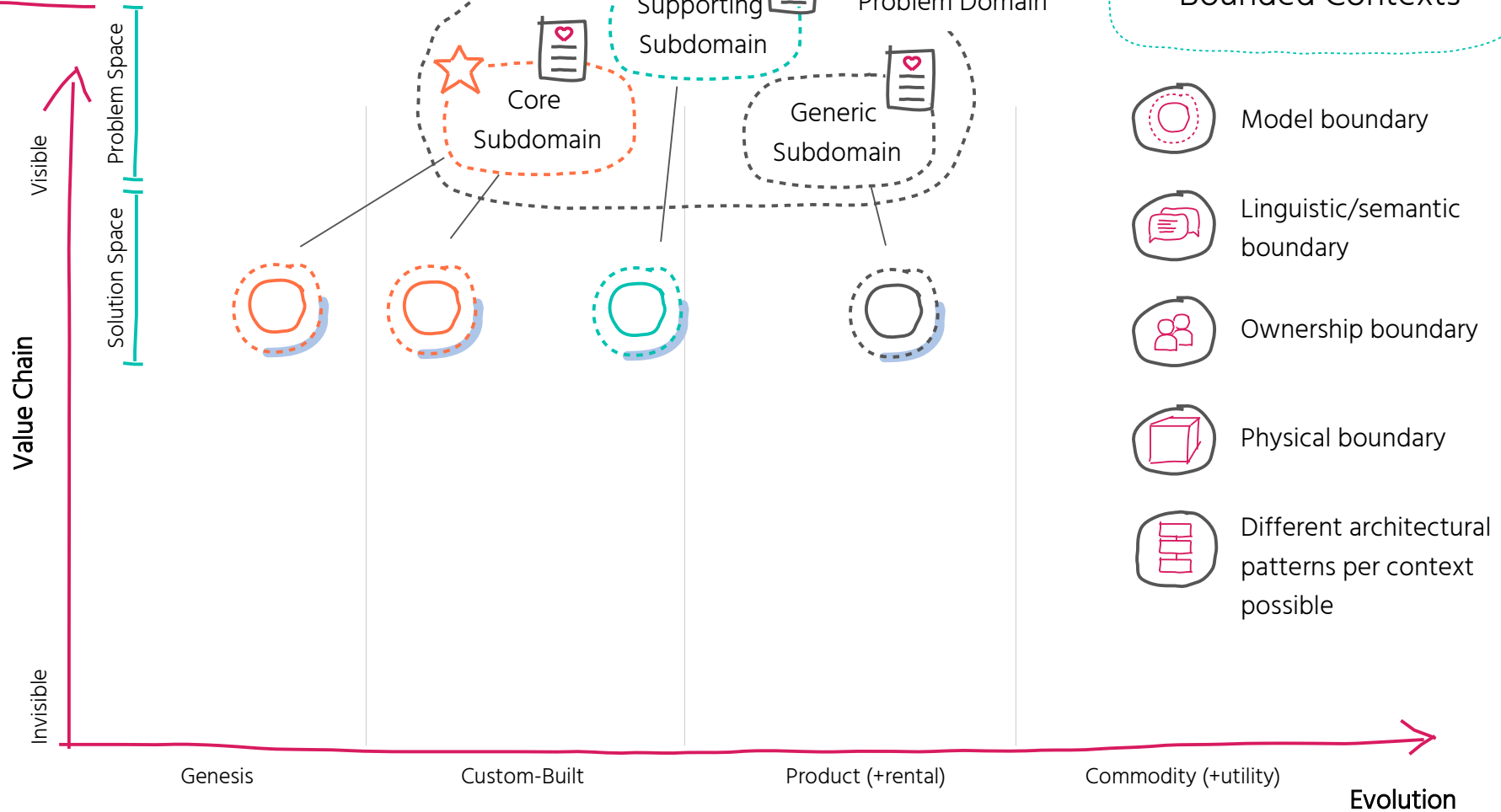
DDD & Wardley Maps

STRATEGIC DESIGN (SOLUTION SPACE)



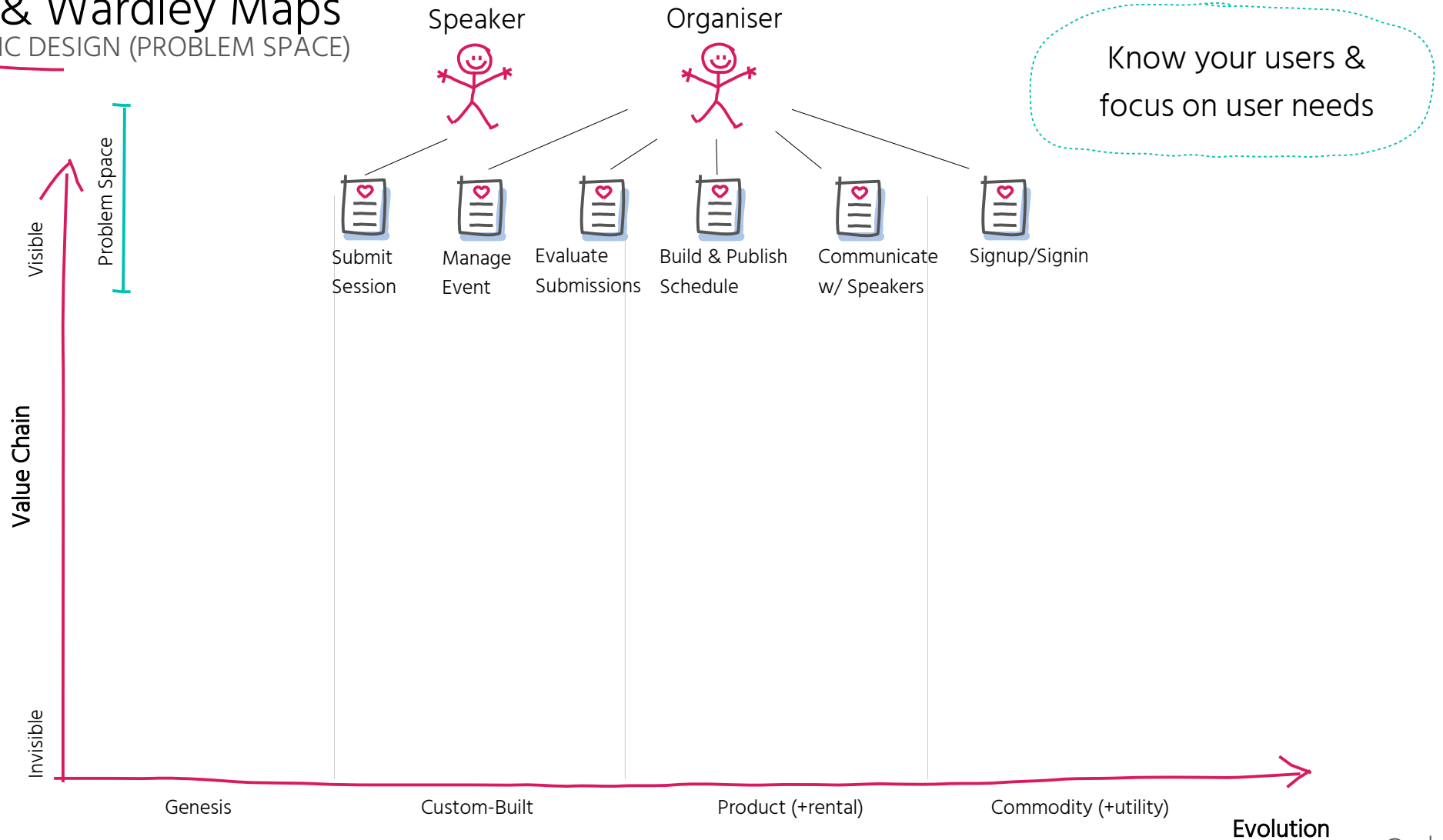
DDD & Wardley Maps

STRATEGIC DESIGN (SOLUTION SPACE)



DDD & Wardley Maps

STRATEGIC DESIGN (PROBLEM SPACE)

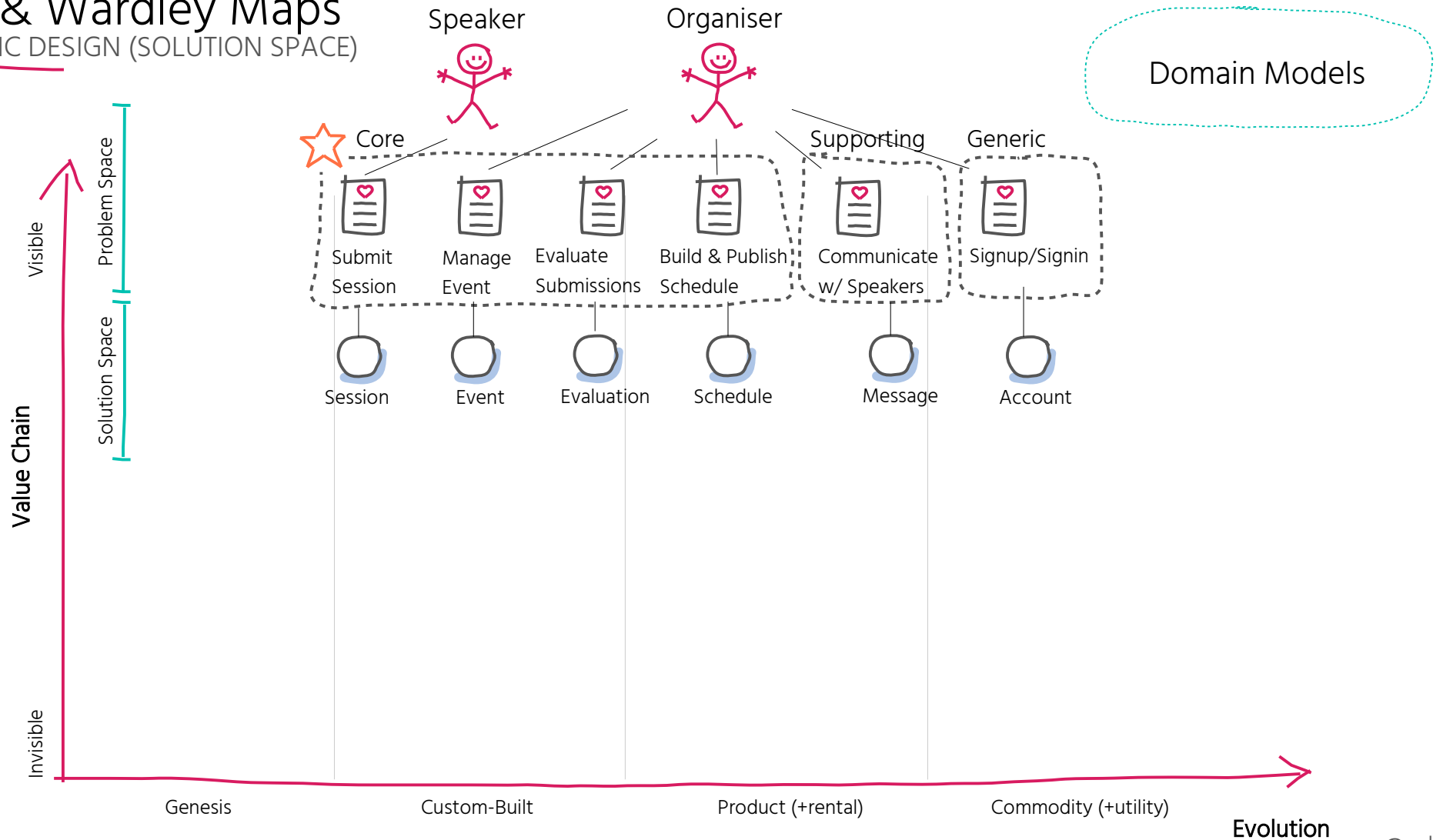


STRATEGIC DESIGN (PROBLEM SPACE)



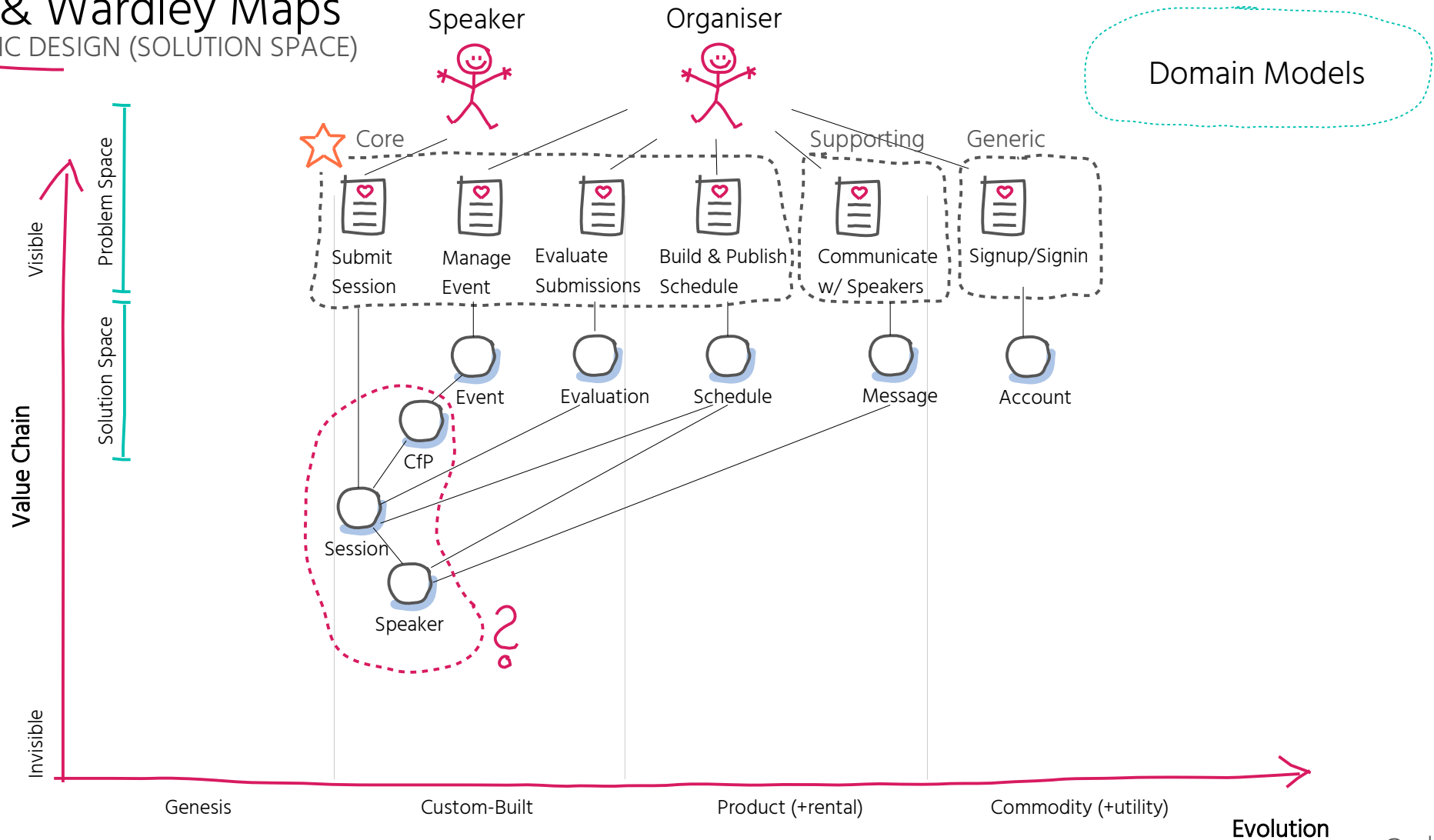
DDD & Wardley Maps

STRATEGIC DESIGN (SOLUTION SPACE)



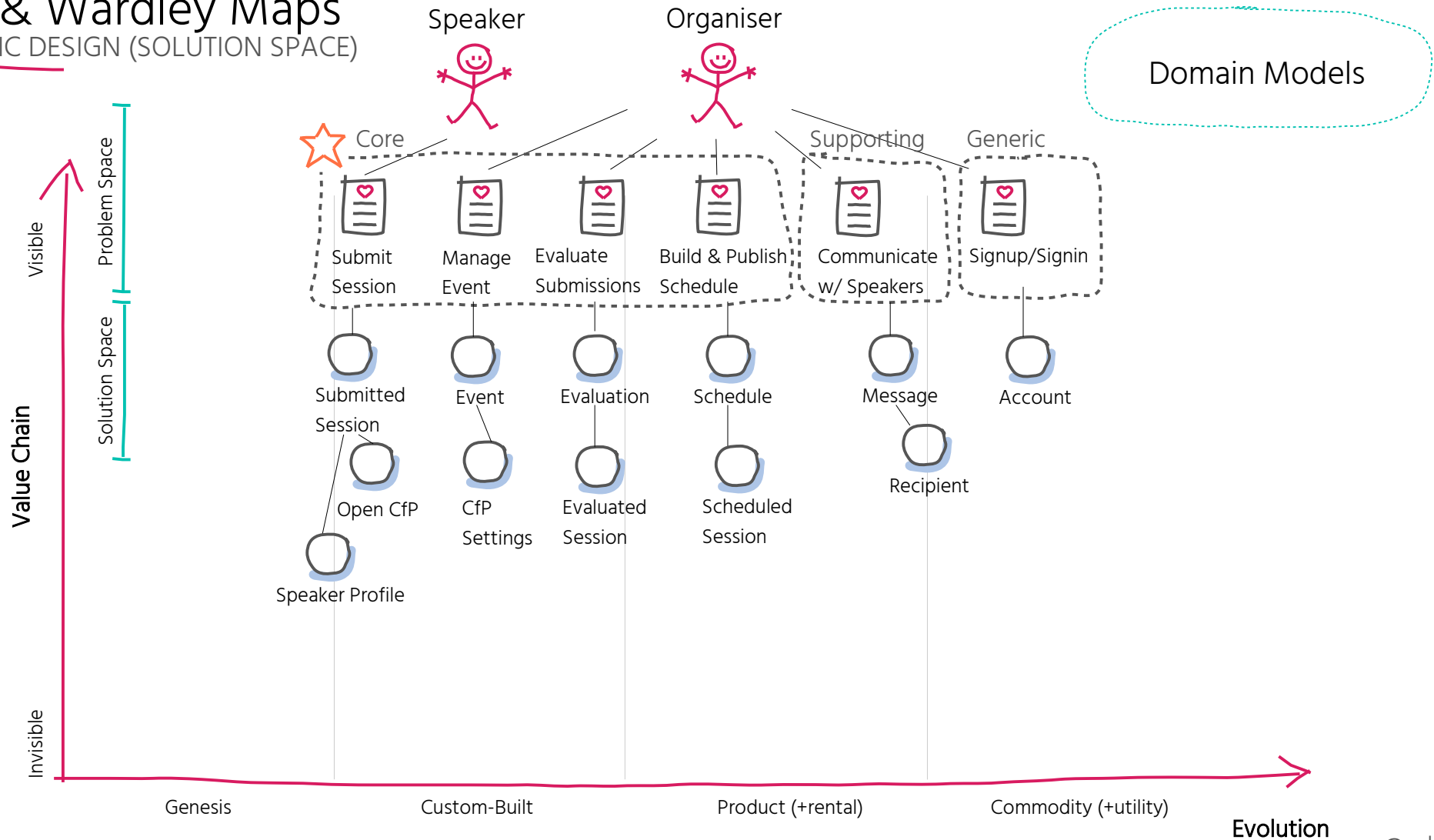
DDD & Wardley Maps

STRATEGIC DESIGN (SOLUTION SPACE)



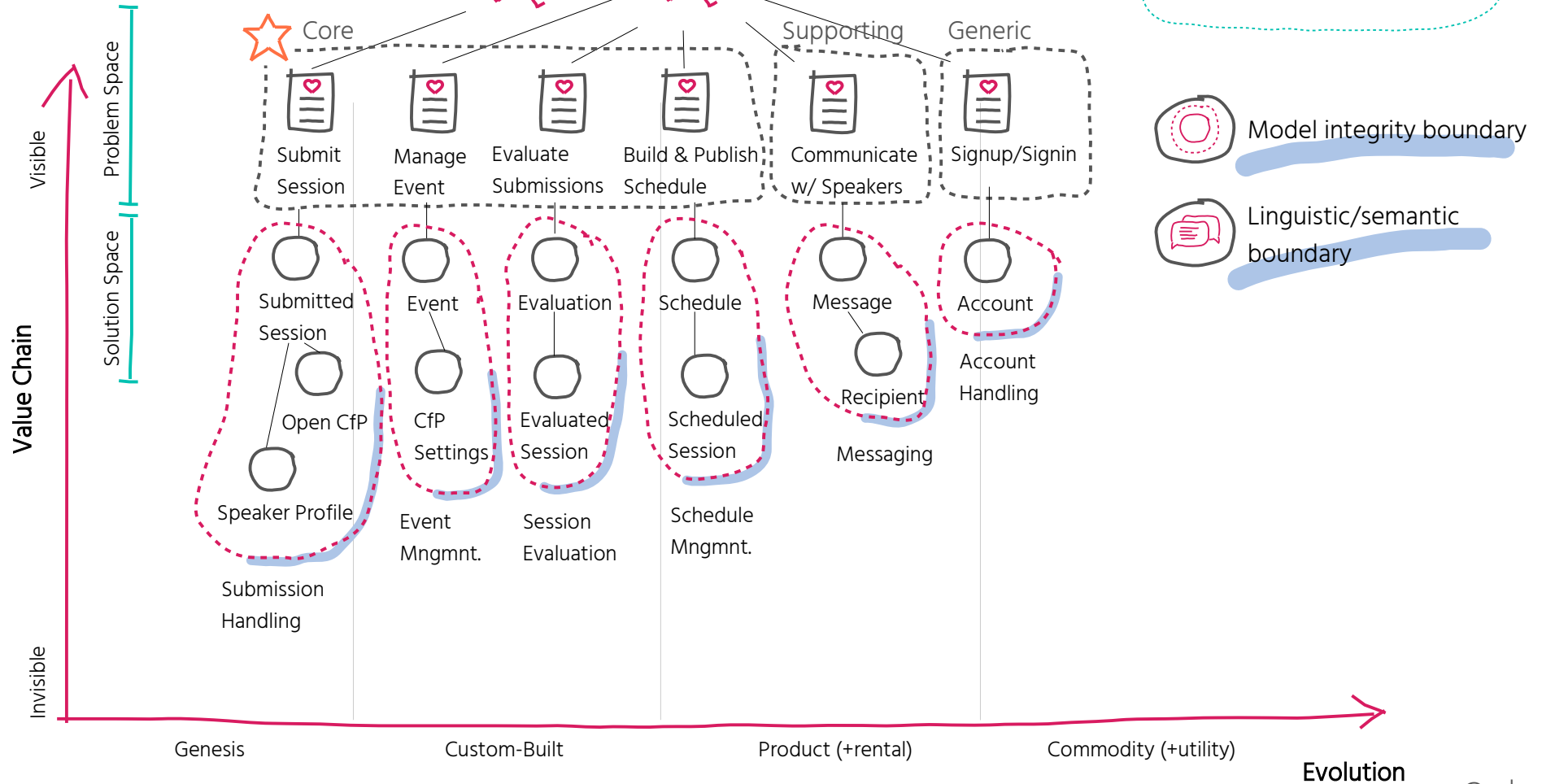
DDD & Wardley Maps

STRATEGIC DESIGN (SOLUTION SPACE)



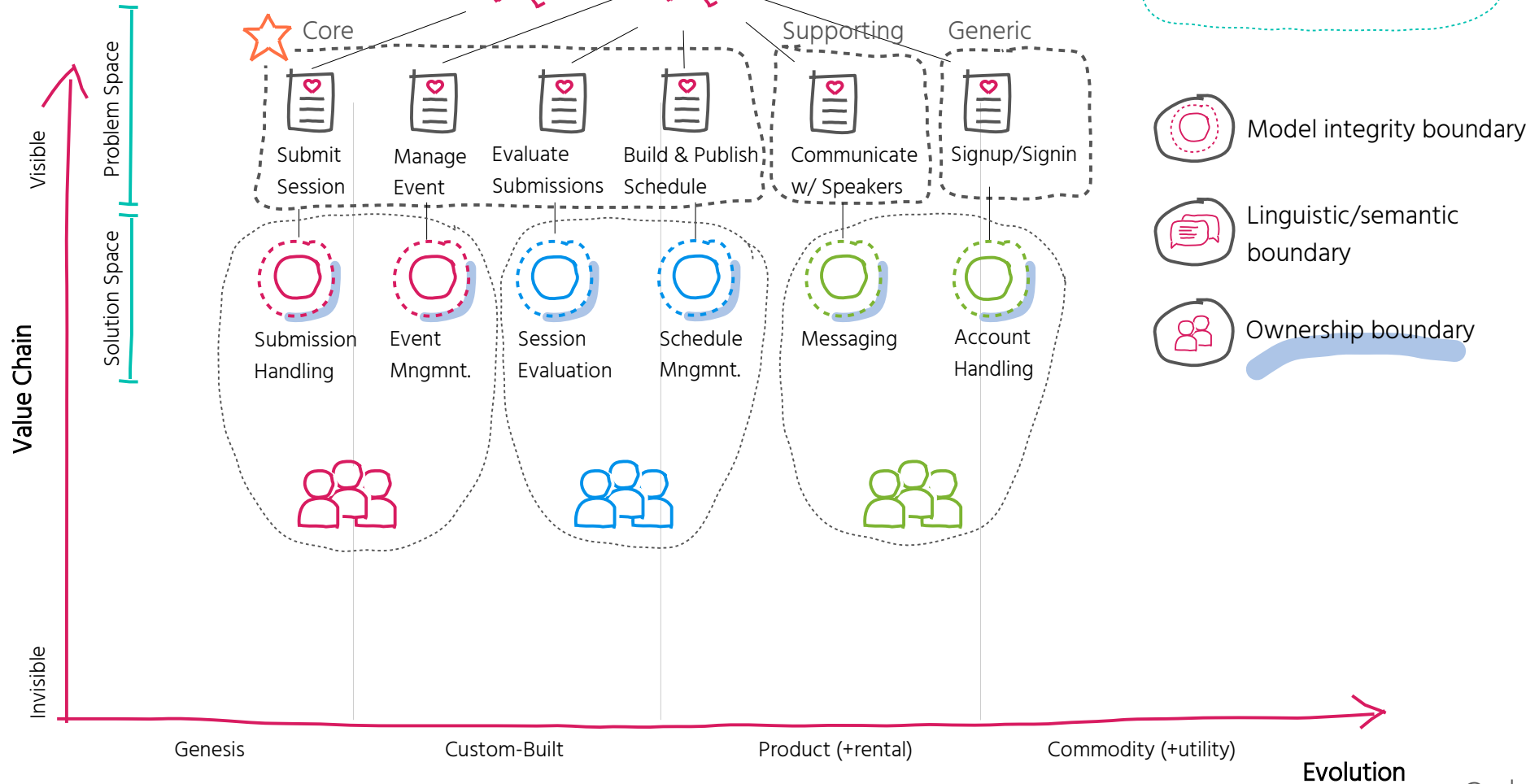
DDD & Wardley Maps

STRATEGIC DESIGN (SOLUTION SPACE)



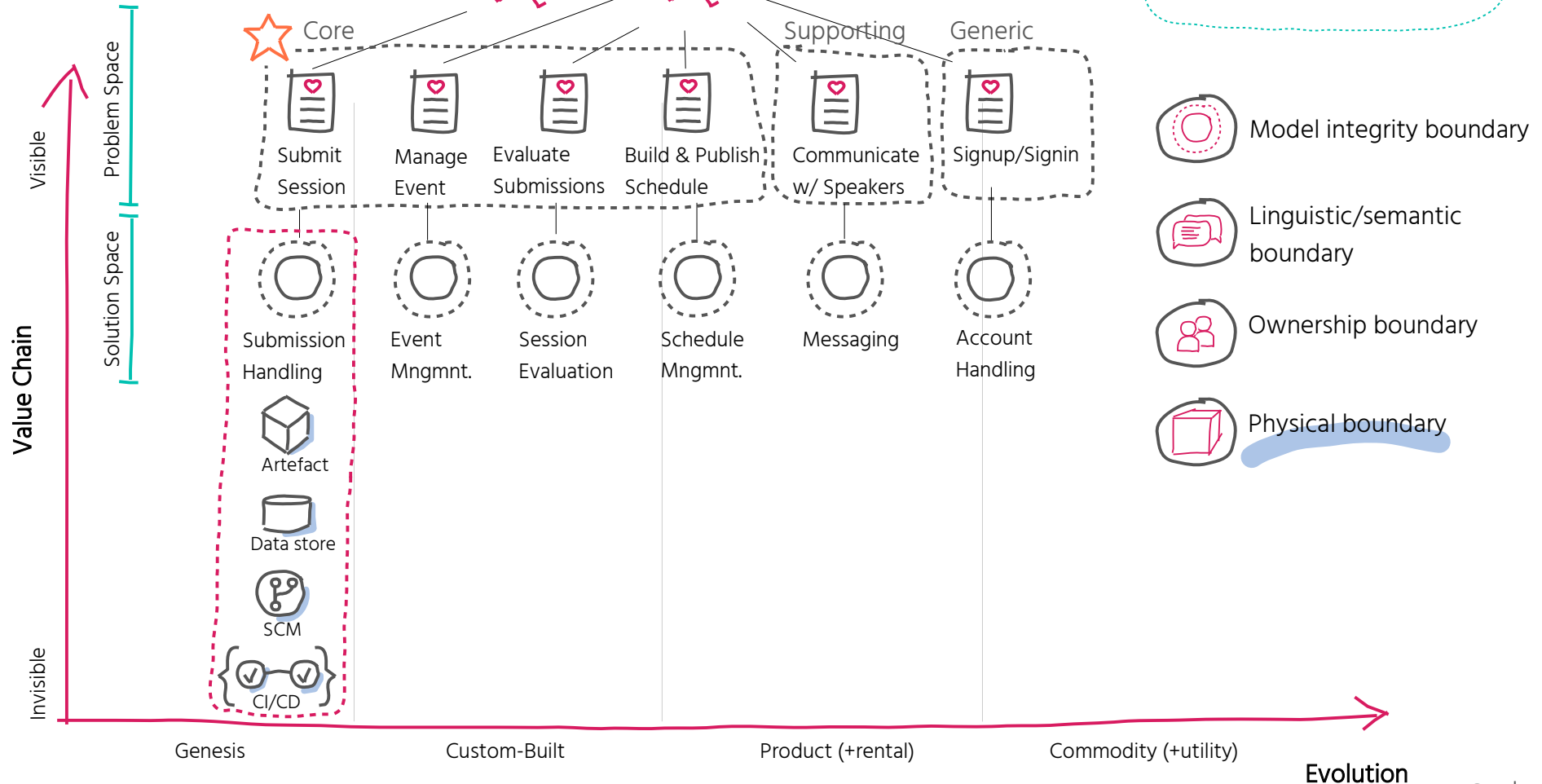
DDD & Wardley Maps

STRATEGIC DESIGN (SOLUTION SPACE)



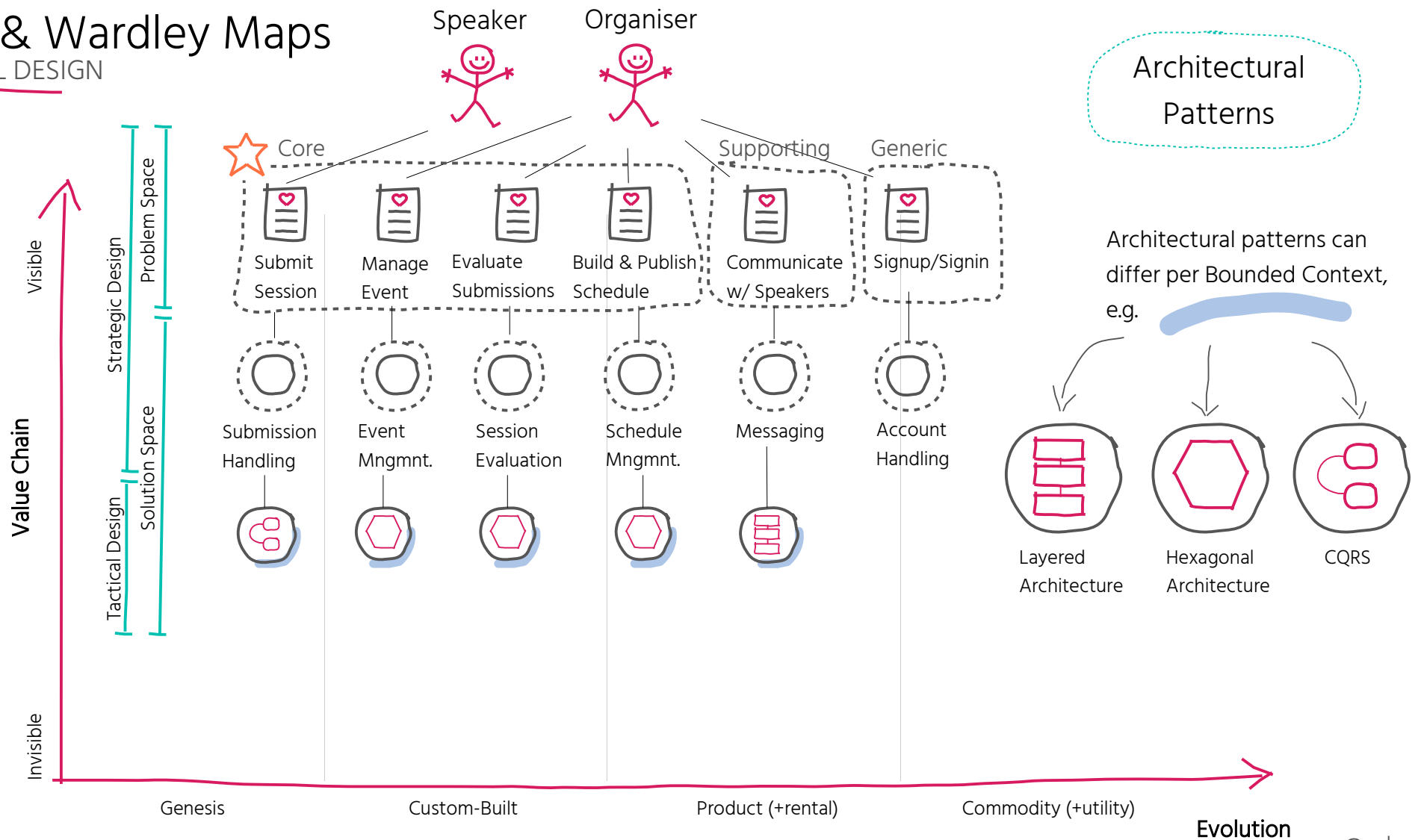
DDD & Wardley Maps

STRATEGIC DESIGN (SOLUTION SPACE)

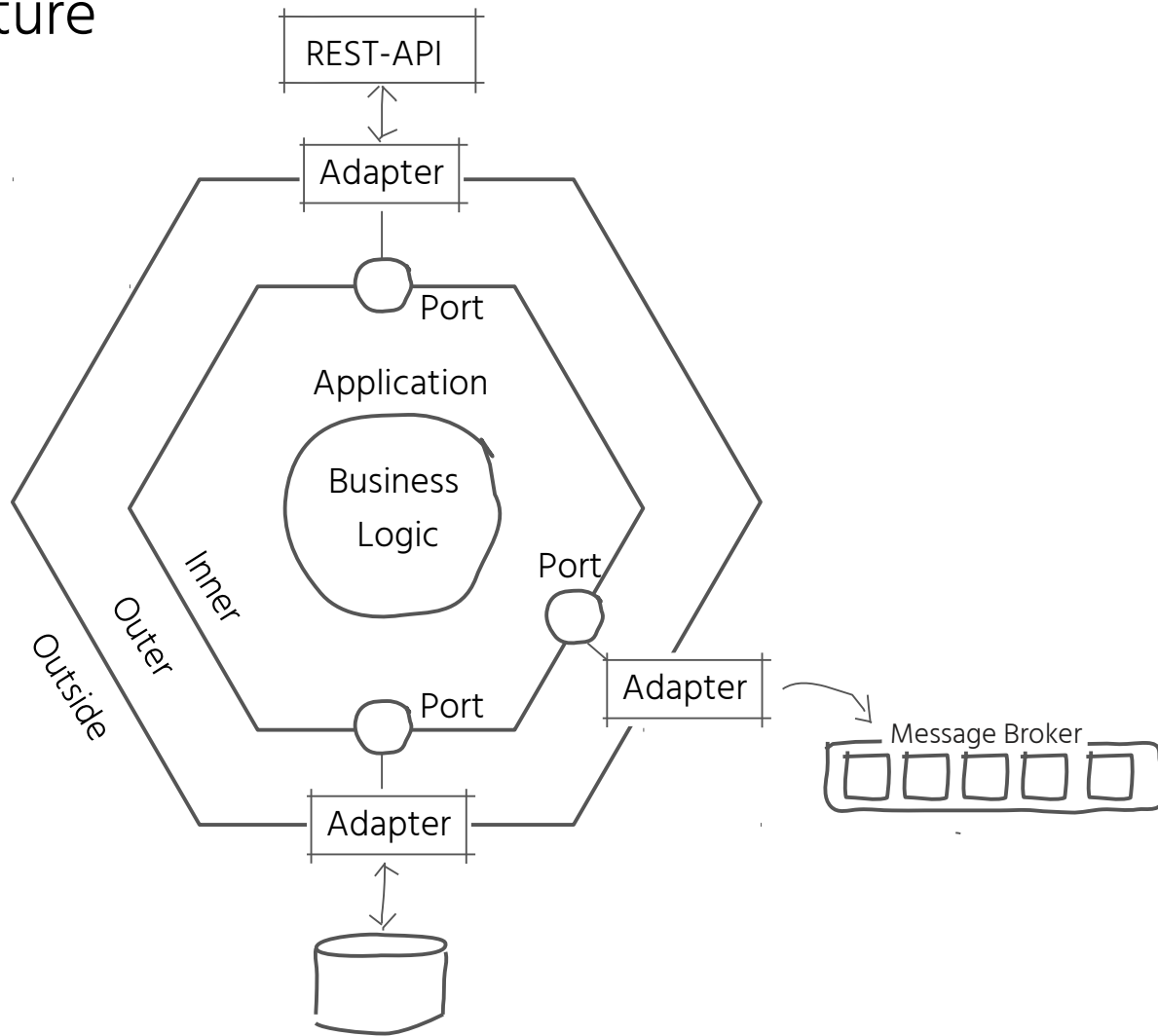


DDD & Wardley Maps

TACTICAL DESIGN

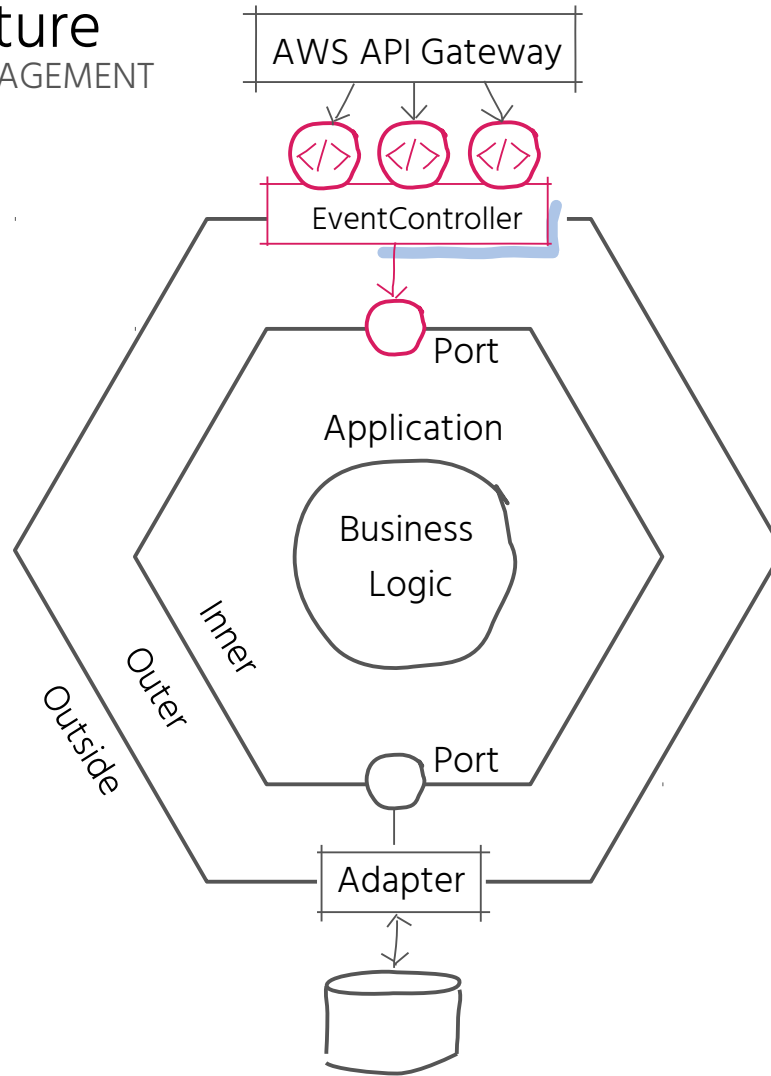


Hexagonal Architecture

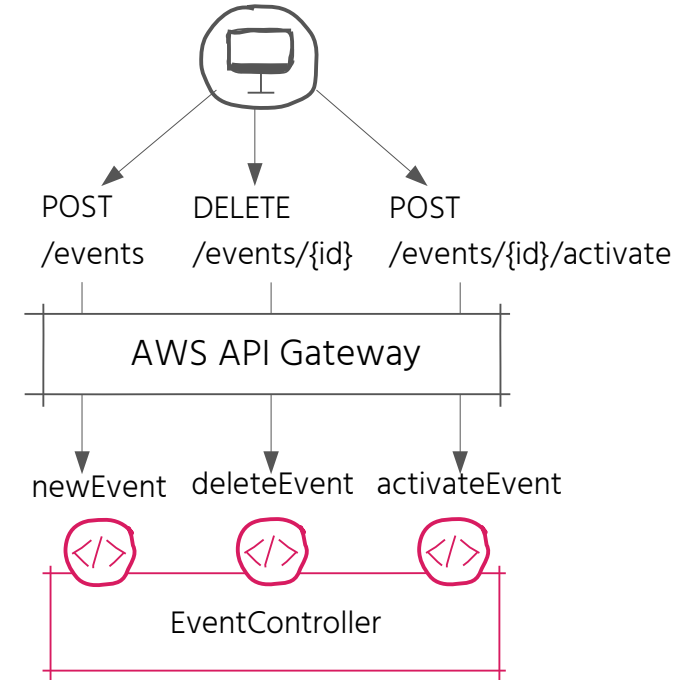


Hexagonal Architecture

BOUNDED CONTEXT: EVENT MANAGEMENT



REST-API with
AWS API-Gateway and
AWS Lambda



Hexagonal Architecture

BOUNDED CONTEXT: EVENT MANAGEMENT

REST-API
Adapter

Port

```
export class EventsController {
```

```
  private readonly eventsService: EventApplicationService;
```

```
  public constructor(eventsService: EventApplicationService) {  
    this.eventsService = eventsService;  
  }
```

Lambda
Function

```
  public activateEvent: Handler = async (event: APIGatewayEvent, context: Context, callback: Callback) => {  
    if (!event.pathParameters) {  
      return callback(undefined, failure({ status: "error", error: "no event id specified" }));  
    }  
    if (!event.requestContext.authorizer) {  
      return callback(undefined, failure({ status: "error", error: "no authorized user specified" }));  
    }  
    try {
```

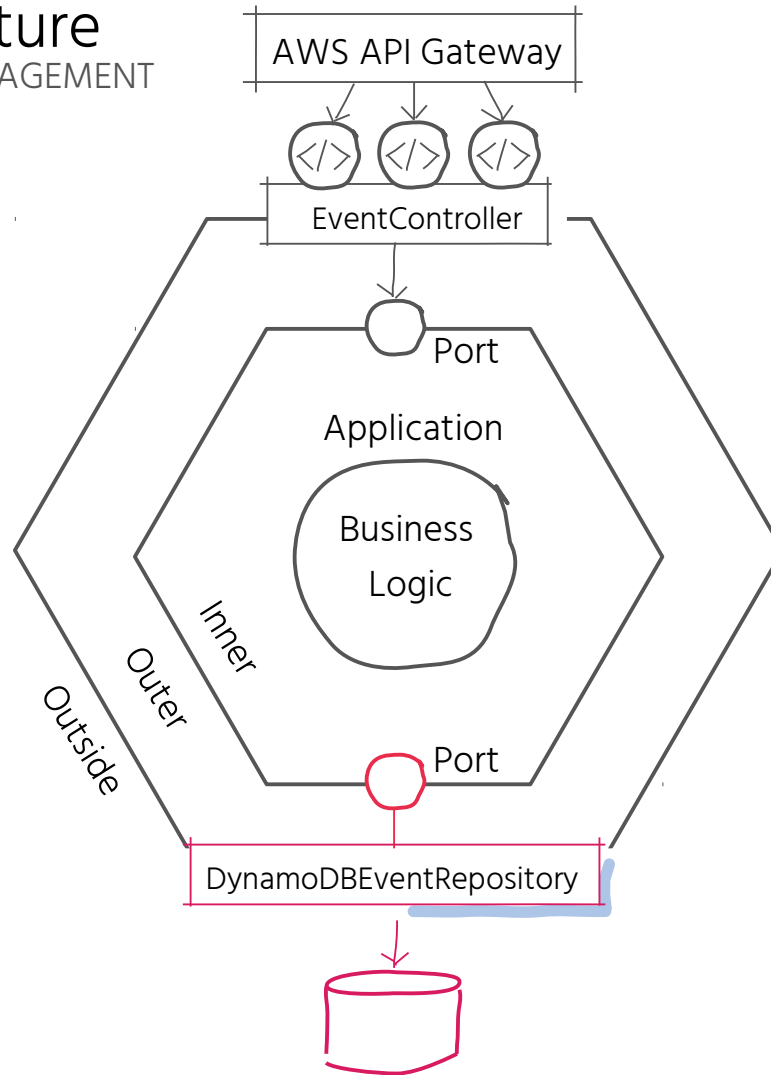
```
      const eventId = new EventId(event.pathParameters.id);  
      const userId = new UserId(event.requestContext.authorizer.claims['cognito:username']);  
  
      await this.eventsService.activateEvent(eventId, userId);  
  
      callback(undefined, success({status: "ok"}));  
    } catch(e) {  
      return callback(undefined, failure({ status: "error", error: e }));  
    }  
  };
```

Lambda
Function

```
  public newEvent: Handler = async (event: APIGatewayEvent, context: Context, callback: Callback) => {  
    // ...  
  }
```


Hexagonal Architecture

BOUNDED CONTEXT: EVENT MANAGEMENT



Data Storage with
AWS DynamoDB

Hexagonal Architecture

BOUNDED CONTEXT: EVENT MANAGEMENT

```
export default class DynamoDBEventRepository implements EventRepository {  
  
    private static TABLE_NAME: string = "events";  
    private readonly dynamoDbClient: AWS.DynamoDB.DocumentClient;  
  
    constructor() {  
        this.dynamoDbClient = new AWS.DynamoDB.DocumentClient();  
    }  
  
    public async eventId(id: EventId): Promise<Event|null> {  
        const params : DocumentClient.GetItemInput = {  
            TableName: DynamoDBEventRepository.TABLE_NAME,  
            Key: {  
                eventId: id  
            }  
        };  
        const result: DocumentClient.GetItemOutput = await this.dynamoDbClient.get(params).promise();  
        const item: AttributeMap|undefined = result.Item;  
        if (item) {  
            const id = new EventId(item.eventId);  
            const name = Name.create(item.name);  
            const description = Description.create(item.description);  
            const period = Period.create(item.period.startDate, item.period.endDate);  
  
            return Event.create(id, name, item.eventStatus, period, description);  
        }  
        return null;  
    }  
  
    public saveEvent(event: Event) {  
        const params : DocumentClient.PutItemInput = {  
            // ...  
        };  
  
        return this.dynamoDbClient.put(params).promise();  
    }  
}
```

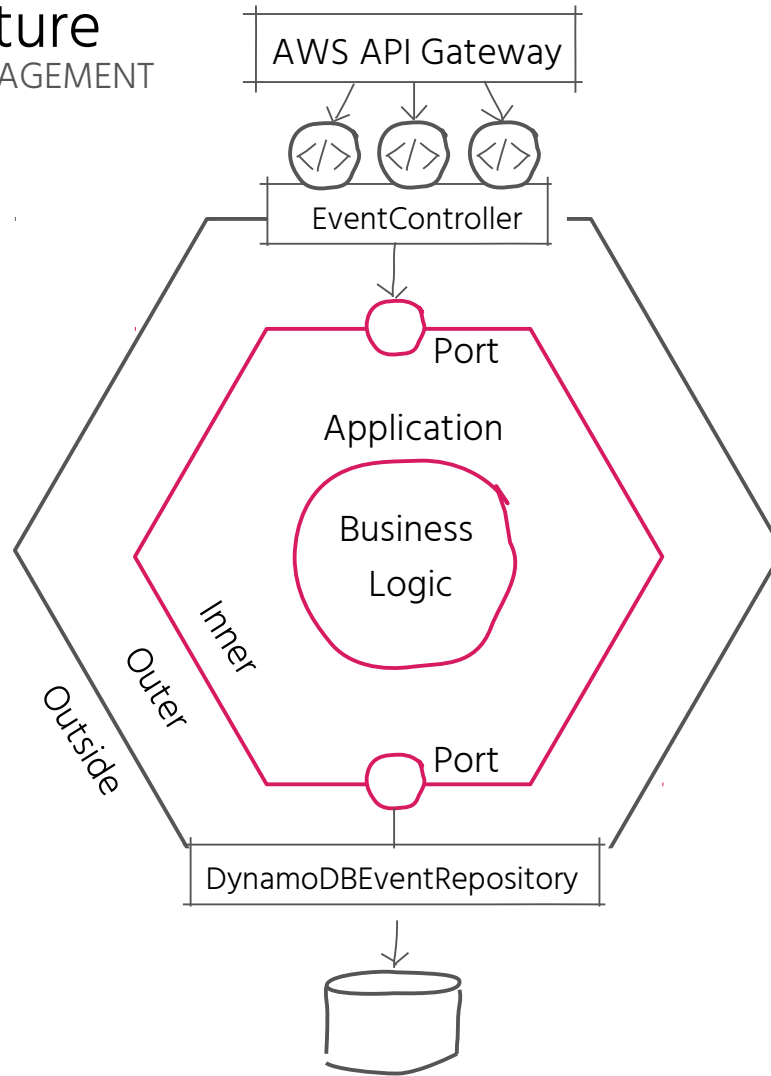
Port

DynamoDB Client

Database Adapter

Hexagonal Architecture

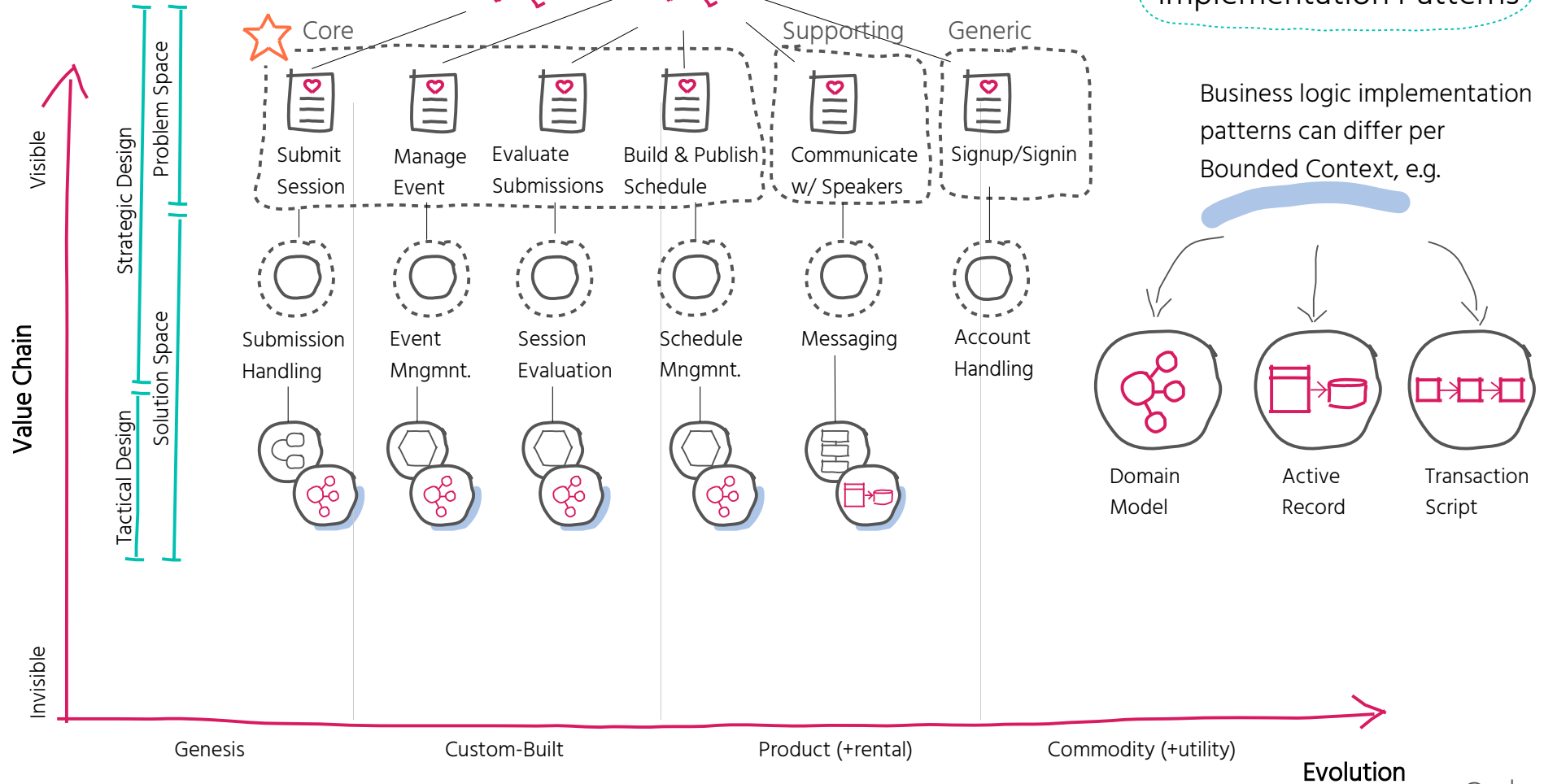
BOUNDED CONTEXT: EVENT MANAGEMENT



Business Logic
Implementation

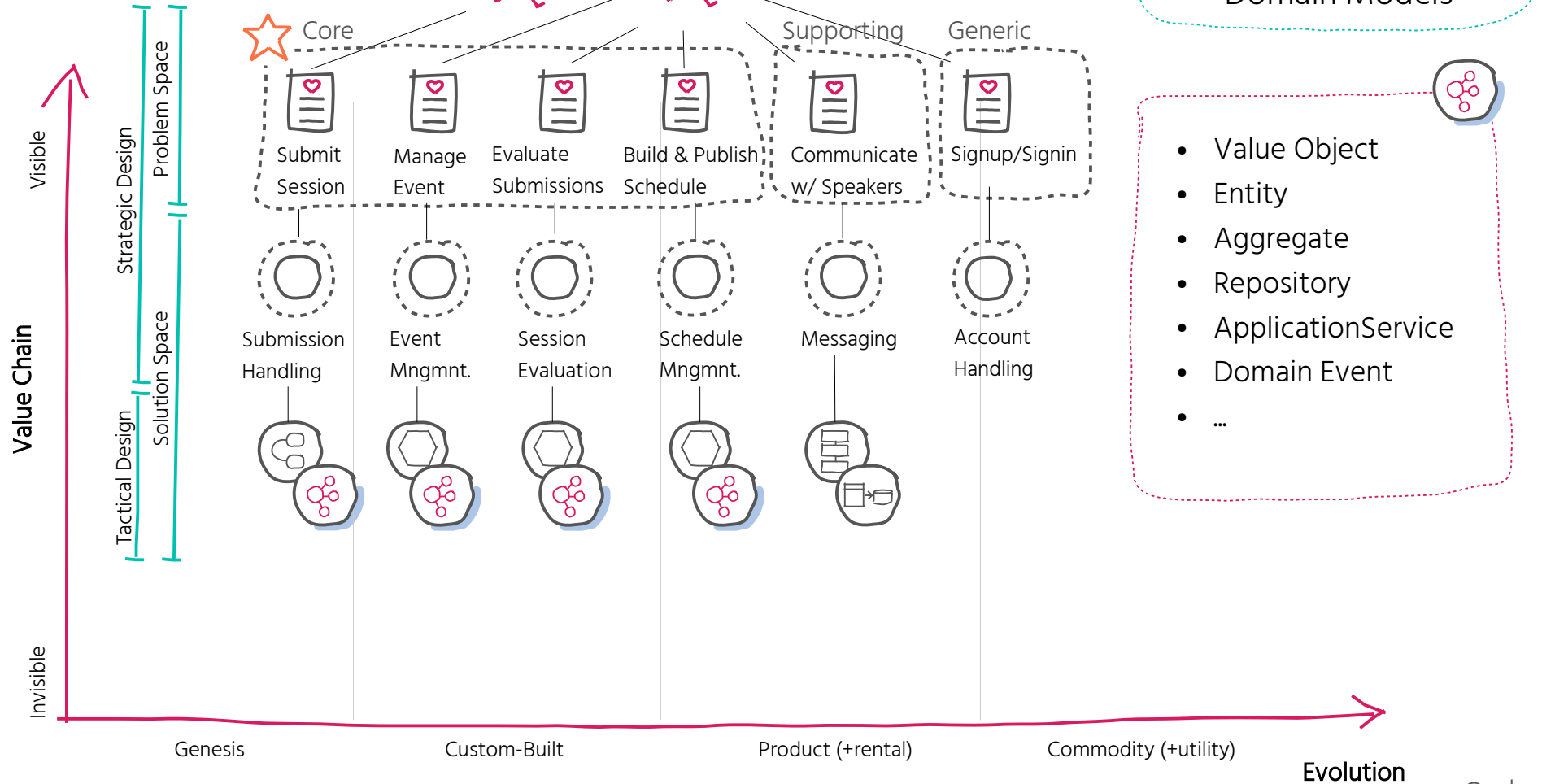
DDD & Wardley Maps

TACTICAL DESIGN



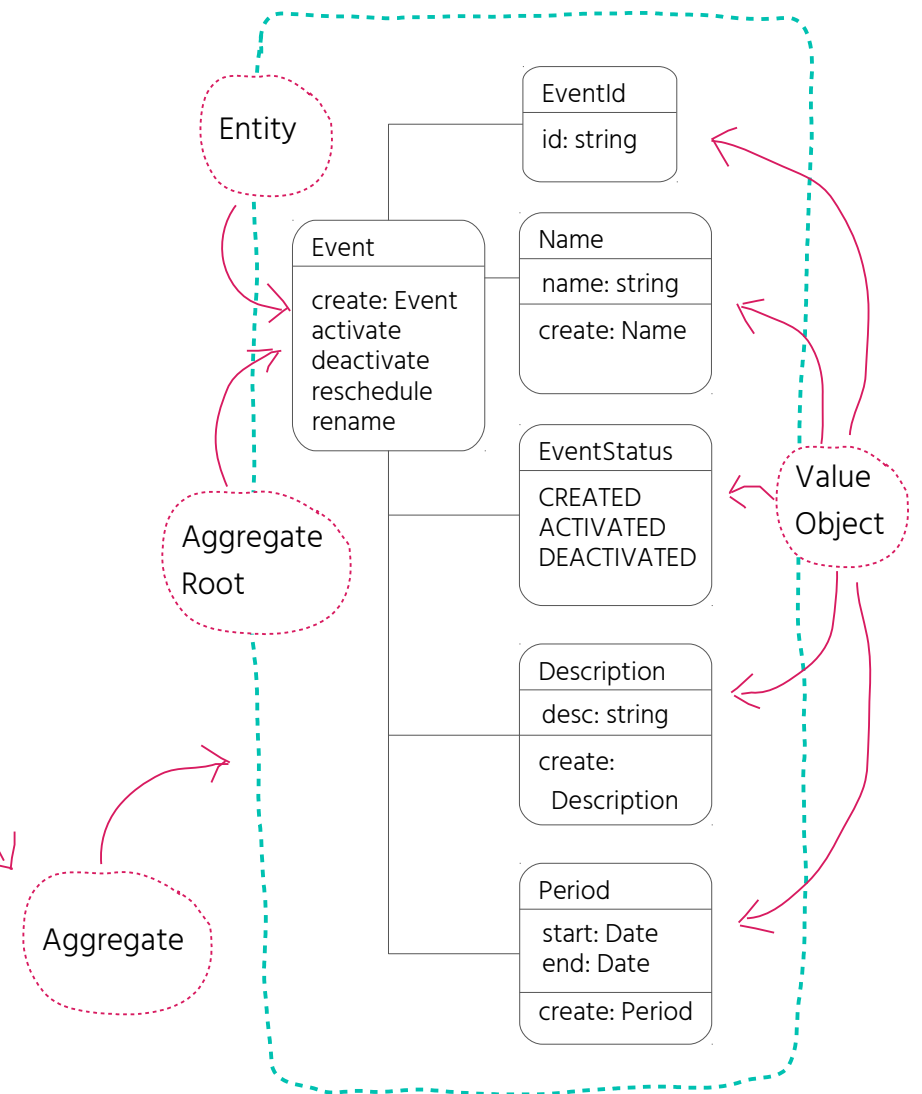
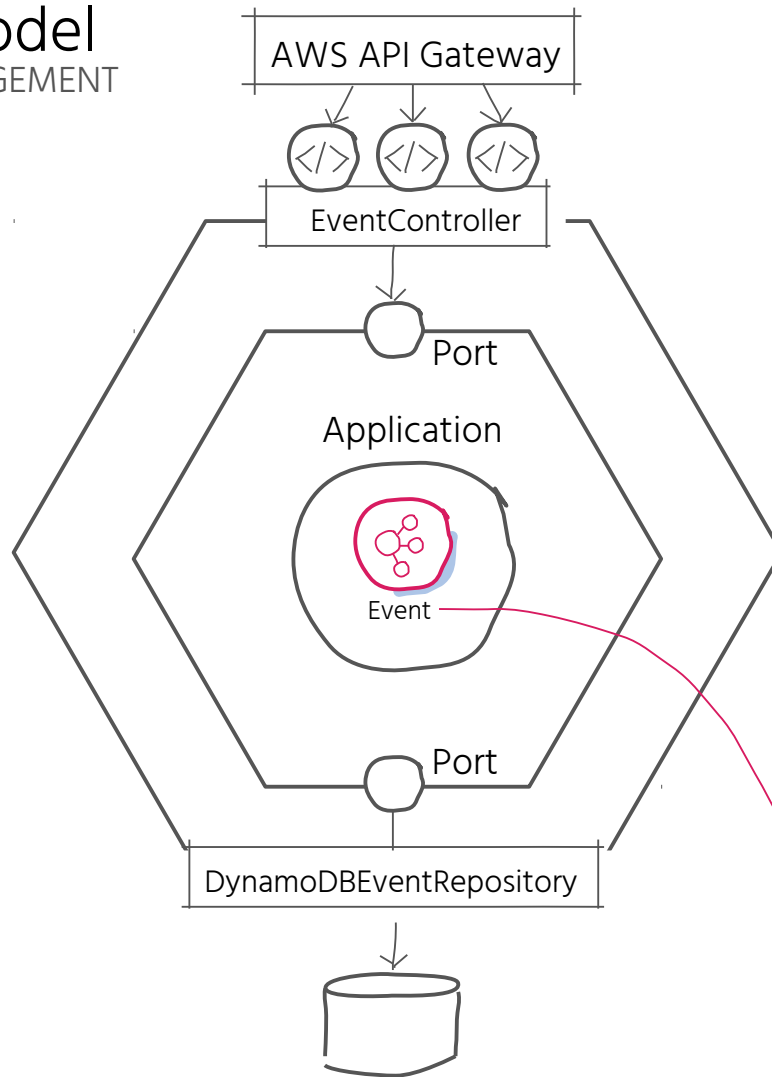
DDD & Wardley Maps

TACTICAL DESIGN



Domain Model

BC: EVENT MANAGEMENT

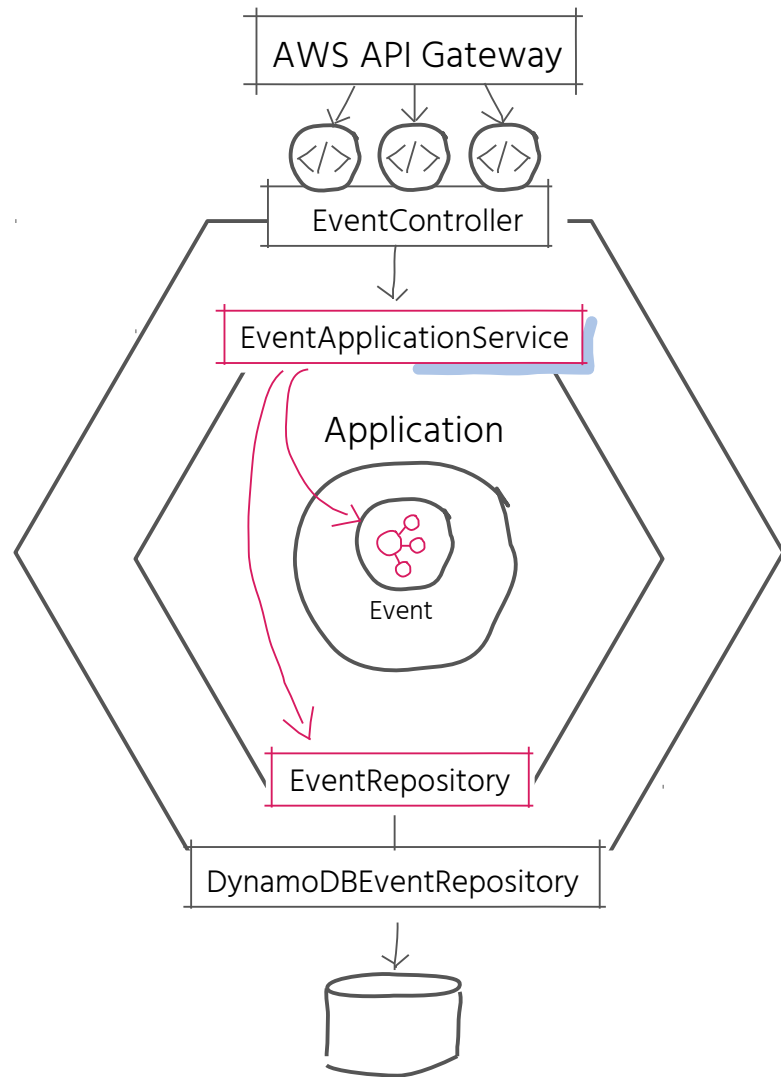


```
export default class Event {  
  readonly id: EventId;  
  name: Name;  
  description?: Description;  
  status: EventStatus;  
  period: Period;  
  
  private constructor(id: EventId, name: Name, status: EventStatus, period: Period, description?: Description) {  
    this.id = id;  
    this.name = name;  
    this.description = description;  
    this.status = status;  
    this.period = period;  
  }  
  
  public activate() {  
    if (this.status === EventStatus.CLOSED) {  
      throw new Error("You cannot activate a closed event");  
    }  
    if (this.status === EventStatus.ACTIVATED) {  
      throw new Error("This event has already been activated");  
    }  
    this.status = EventStatus.ACTIVATED;  
  }  
  
  public rename(name: Name) {  
    if (!name) {  
      throw new Error("You cannot rename the event to an empty name");  
    }  
    this.name = name;  
  }  
  
  // ... //  
}
```

Aggregate

Domain Model

BC: EVENT MANAGEMENT



Domain Model

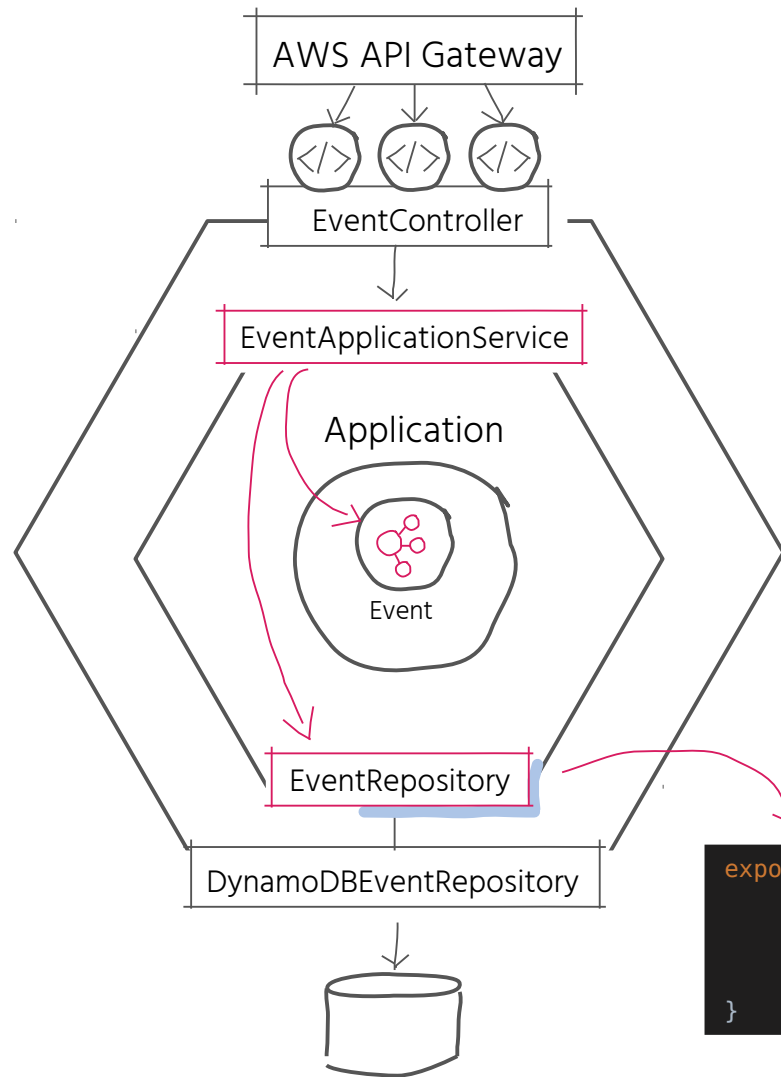
EVENT MANAGEMENT

ApplicationService

```
export default class EventApplicationService {  
  private readonly eventRepository: EventRepository;  
  
  constructor(eventRepository: EventRepository) {  
    this.eventRepository = eventRepository;  
  }  
  
  public async activateEvent(id: EventId) {  
    const event = await this.eventRepository.eventOfId(id);  
    if (!event) {  
      throw new Error("Could not deactivate event with id " + id + ", since event does not exist.");  
    }  
    event.activate();  
    await this.eventRepository.saveEvent(event);  
  }  
  
  // ... //  
}
```

Domain Model

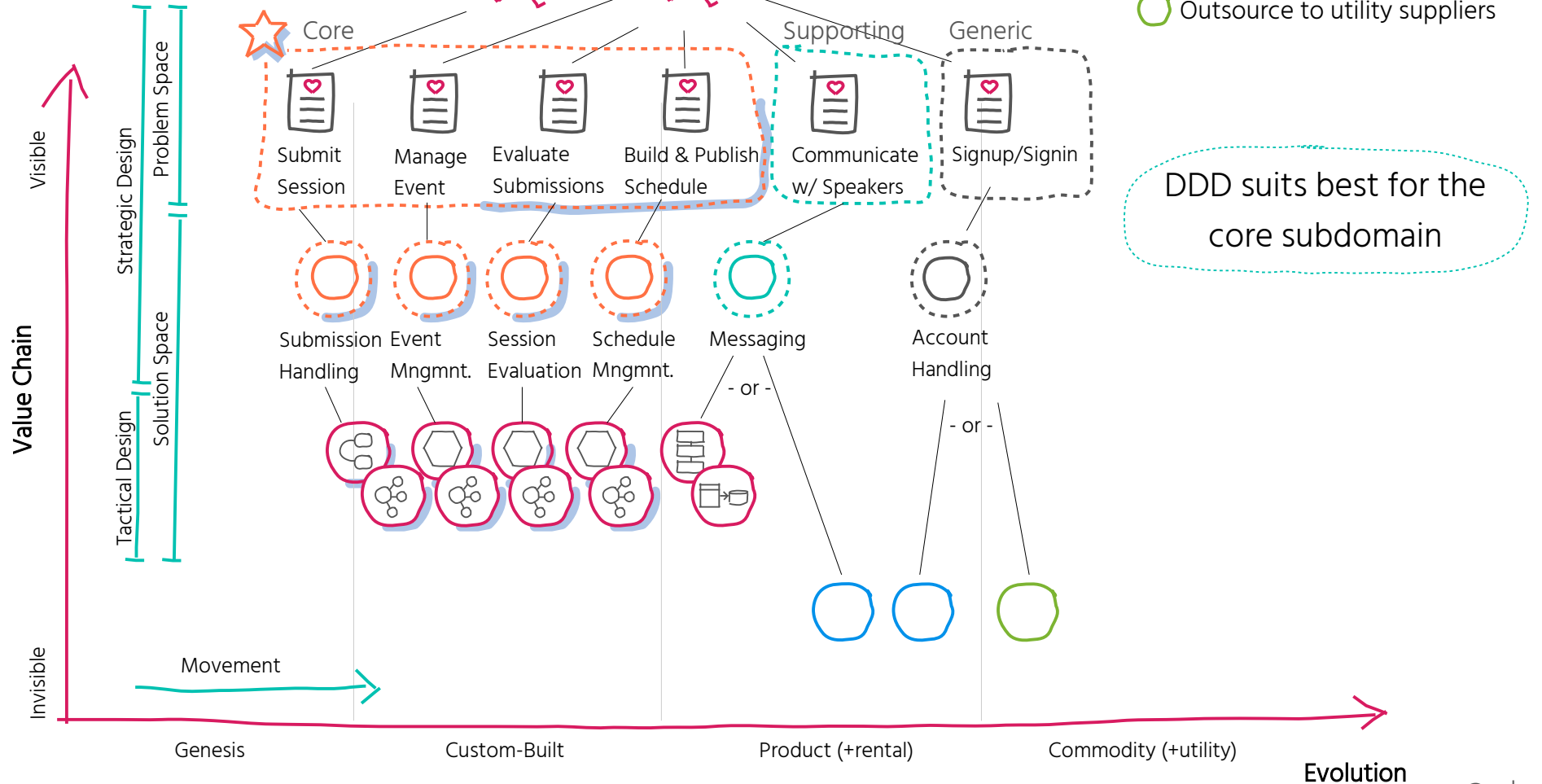
EVENT MANAGEMENT



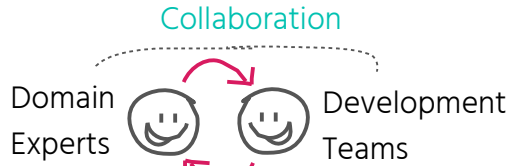
```
export default interface EventRepository {  
  saveEvent(event: Event): void;  
  eventOfId(id: EventId): Promise<Event|null>;  
  // ... //  
}
```

DDD & Wardley Maps

TACTICAL DESIGN



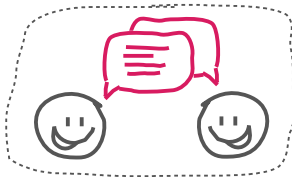
DDD helps with ...



Domain Knowledge



Ubiquitous Language



Gaining Domain Knowledge



Core
Subdomain

Discovering the
Core Subdomain



Business
Domain



Needs



Strategy

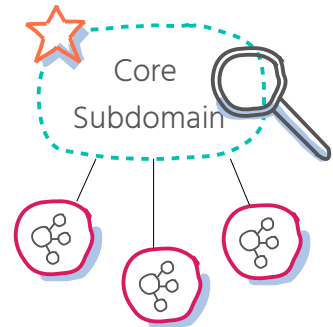


Better Software Design

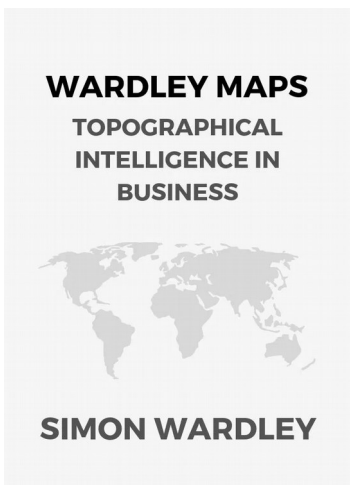
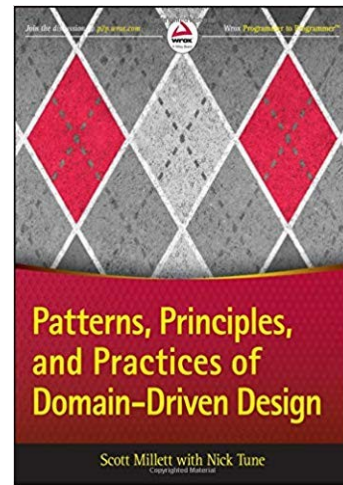
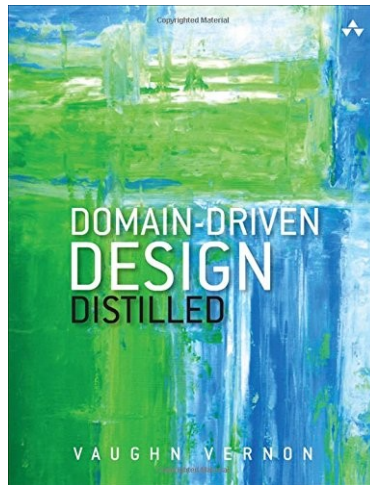
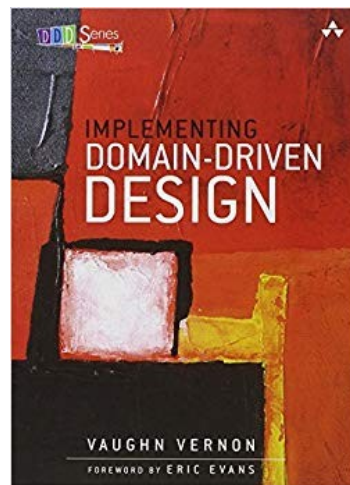
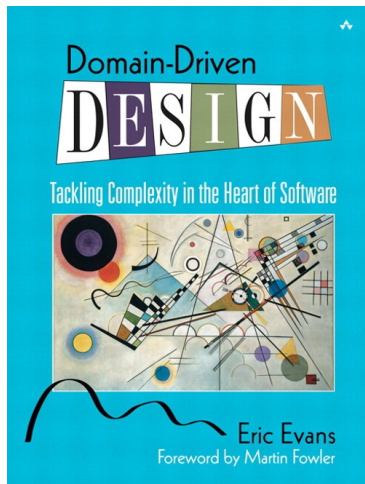
Aligning Software Design
to Business Domain

But ...

Do not apply DDD
everywhere!
Focus on your core!



Some References



<https://learnwardleymapping.com/>

<https://medium.com/wardleymaps>

<https://miro.com/blog/wardley-maps-whiteboard-canvas/>

<https://community.wardleymaps.com/>