(Six) Developer Best Practices

Looking beyond the hype

Markus Völter
(voelter@acm.org)

About

Markus Voelter
Independent Consultant

About

Markus Voelter
Independent Consultant

Software Architecture
DSLs & MDSD
Product Lines

About

Markus Voelter
Independent Consultant

http://www.voelter.de
voelter@acm.org
skype: schogglad

1 Introduction


Frustrated by
all the hype?


If so
this presentation
is for you.


Otherwise
you should leave ☺

People
often
talk
about
technologies
instead of
core concepts.

Technology-discussions create a lot of accidental complexity.


The essence
is hidden behind
tech gobbledygook


Concepts change much
More slowly than the
Hype-driven techno
marketing

This presentation:

This Introduction

This presentation:

2

Principles Quick Tour

This presentation:

Three Case Studies

This presentation:

Three Case Studies

3 SOA

This presentation:

Three Case Studies

3 SOA

4 Concurrency

This presentation:

Three Case Studies

3 SOA

4 Concurrency

5 DSLs and Stuff

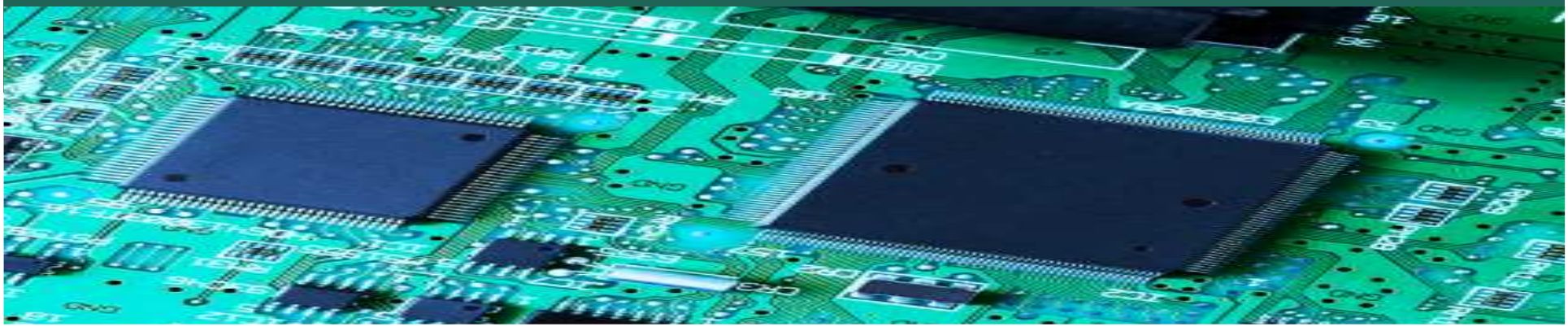
This presentation:

6

Recap at the End

Principles Quick Tour

Modularize



Encapsulate

Partitioning

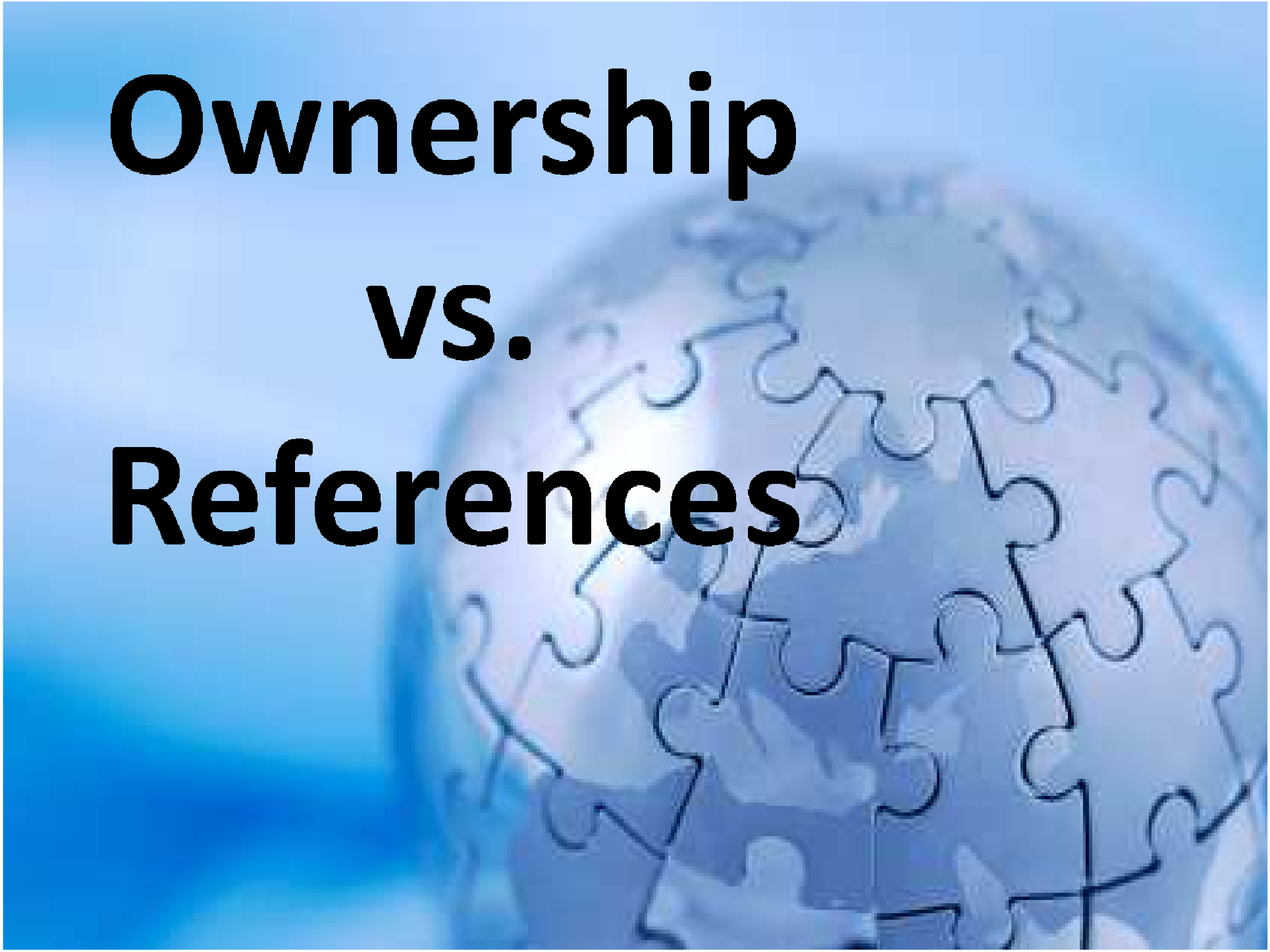
Contracts

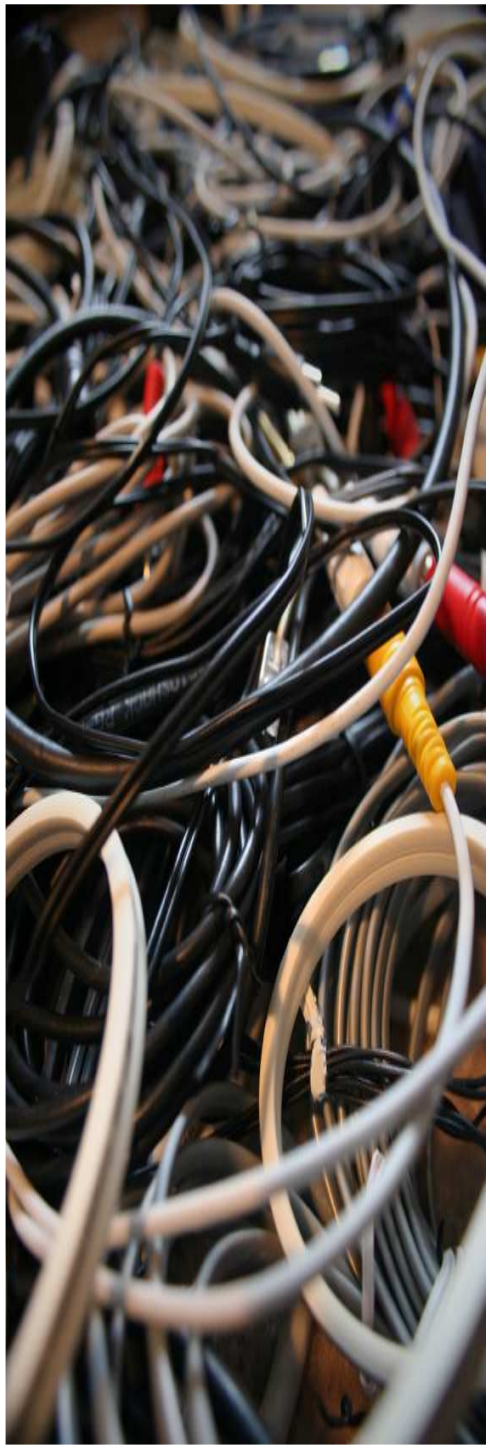
Decoupling

Indirection

Discovery

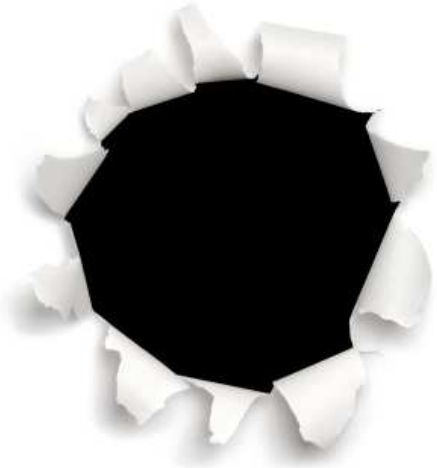
Ownership vs. References



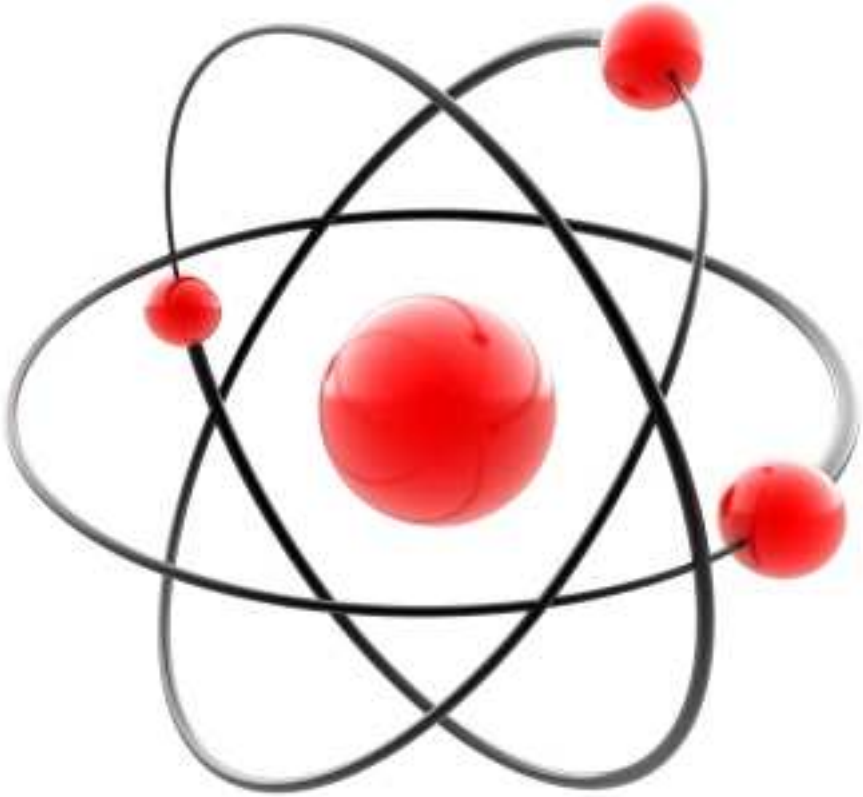


Handle Crosscuts

Go Down



Isolate



Atomicity

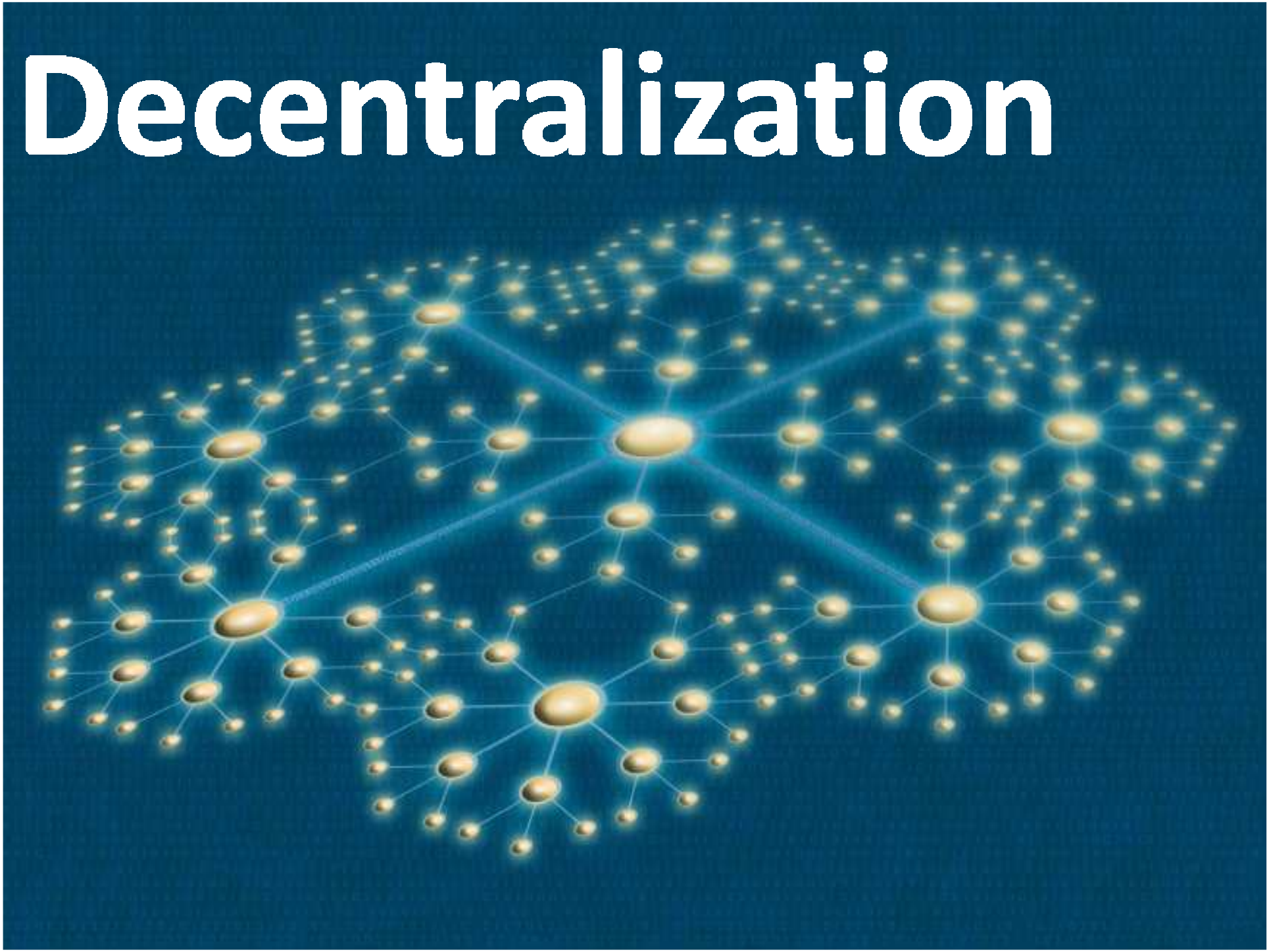
Isolate Technology

Parametrization

Staging

Simplicity

Decentralization




Open/Closed
World

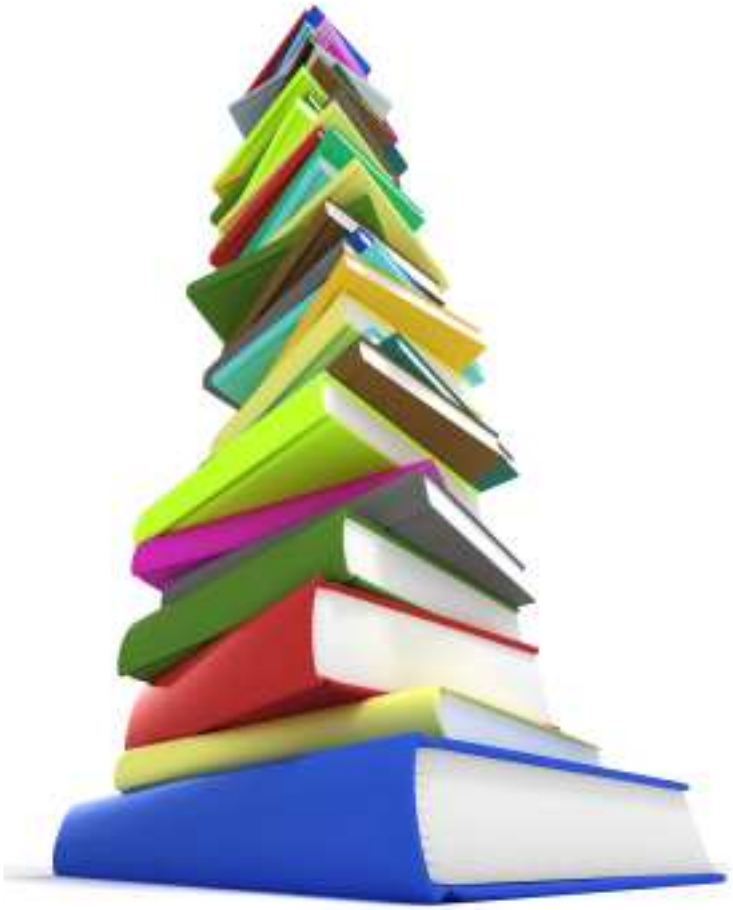
Distribution vs. Local Caching

Compensation

Deferred Consistency

Bootstrapping

Standard Library



Platforms

Containers

Orthogonality

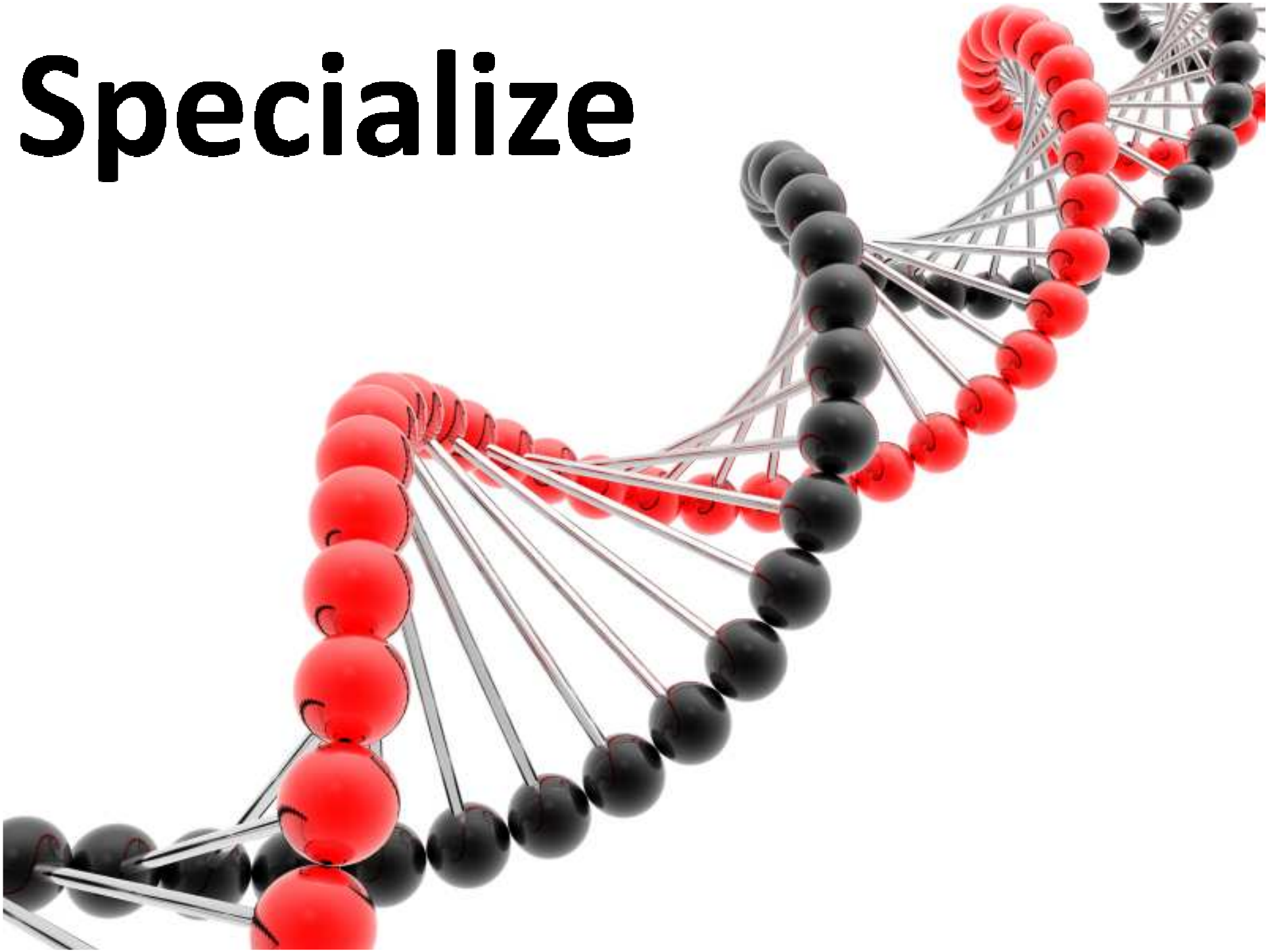
Identity

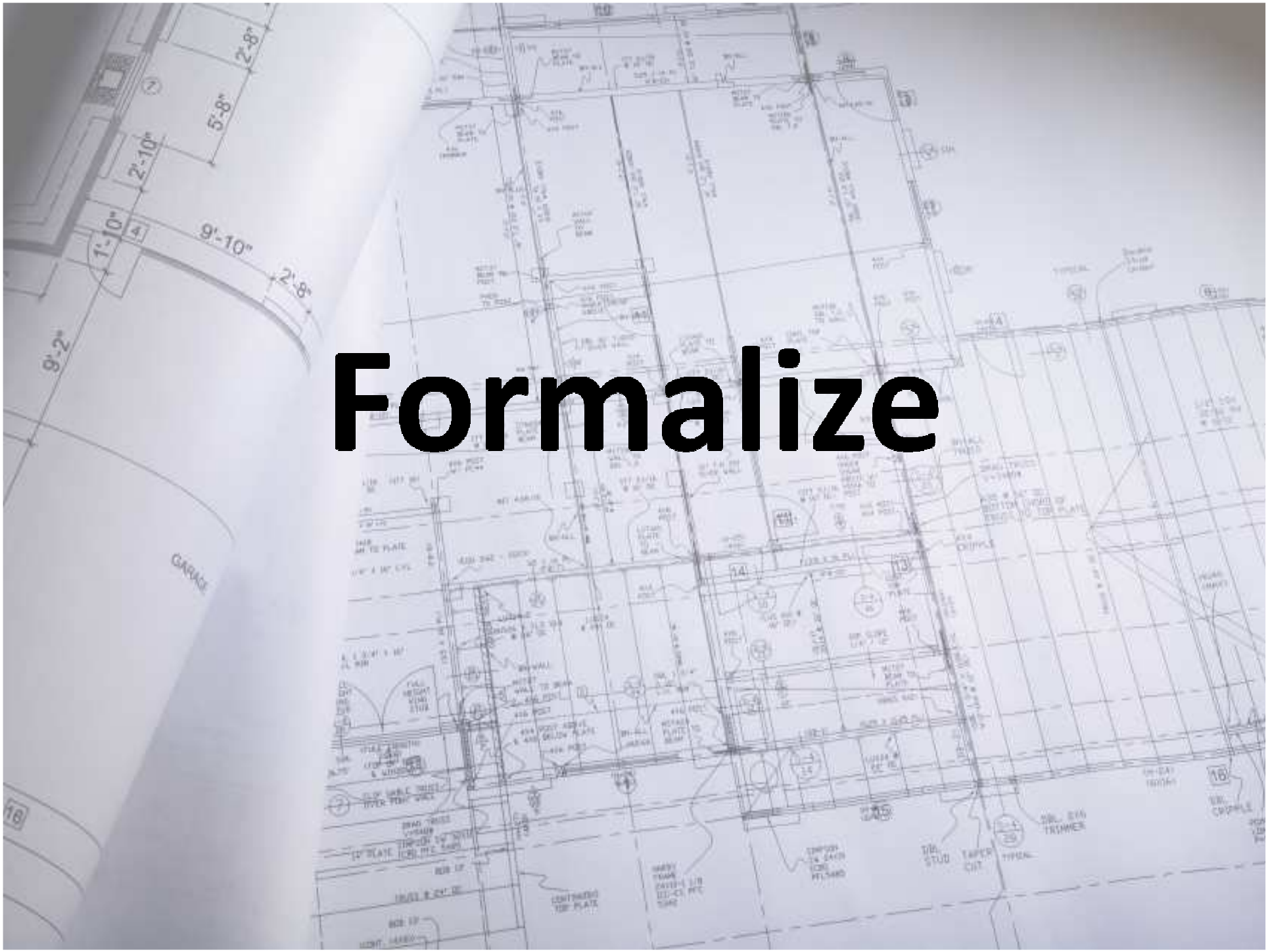

Abstraction

Types & Instances

Hierarchical Decomposition

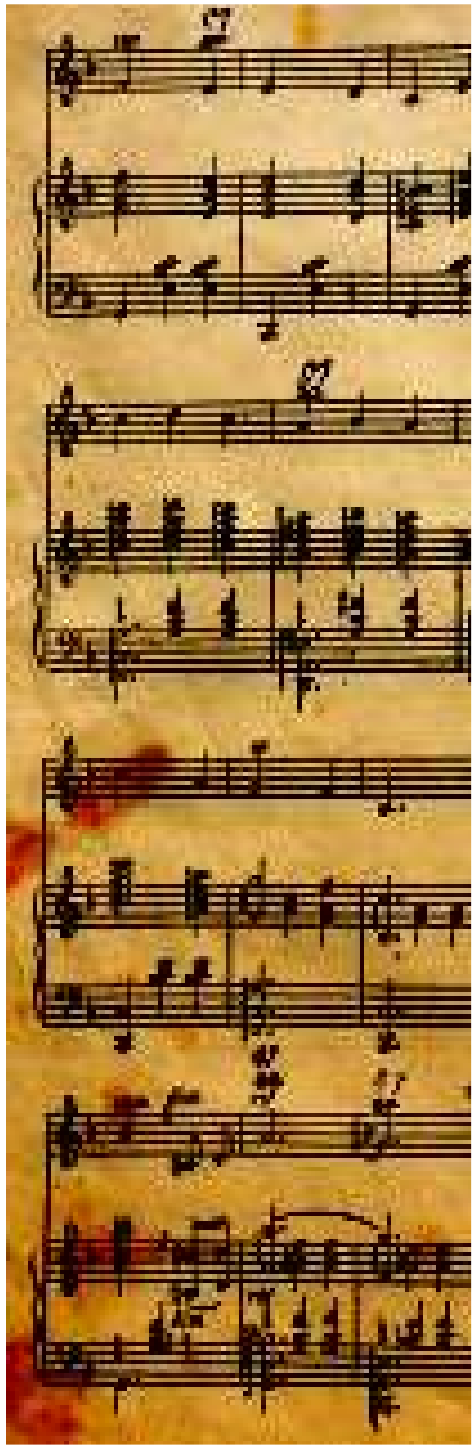
Specialize




Formalize


Viewpoints

Notation




Go Meta

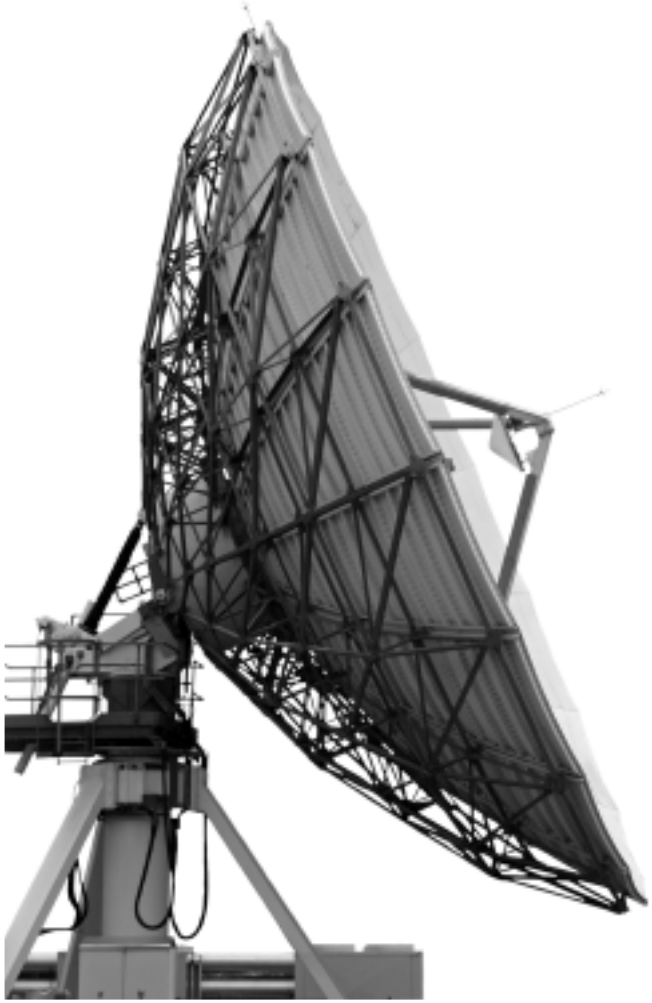
Reflection

Evolution/ Self Modification

Translate

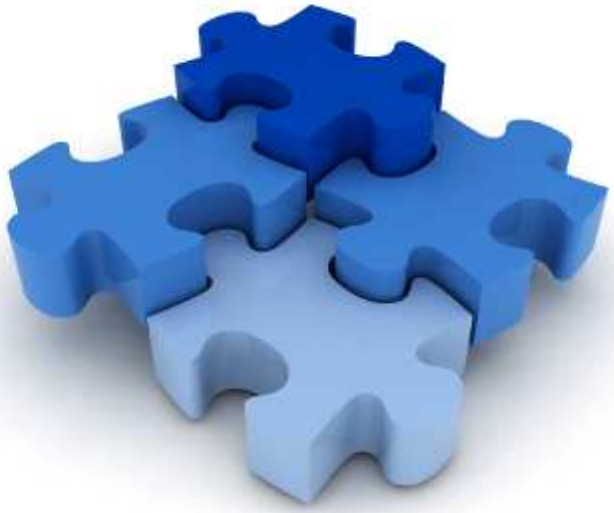
Interpret

Tracking



Automate

Protocols




Make
Transparent

Make Explicit

Lazy vs. Eager


30
Limit
Freedom

Declaration

Implementation

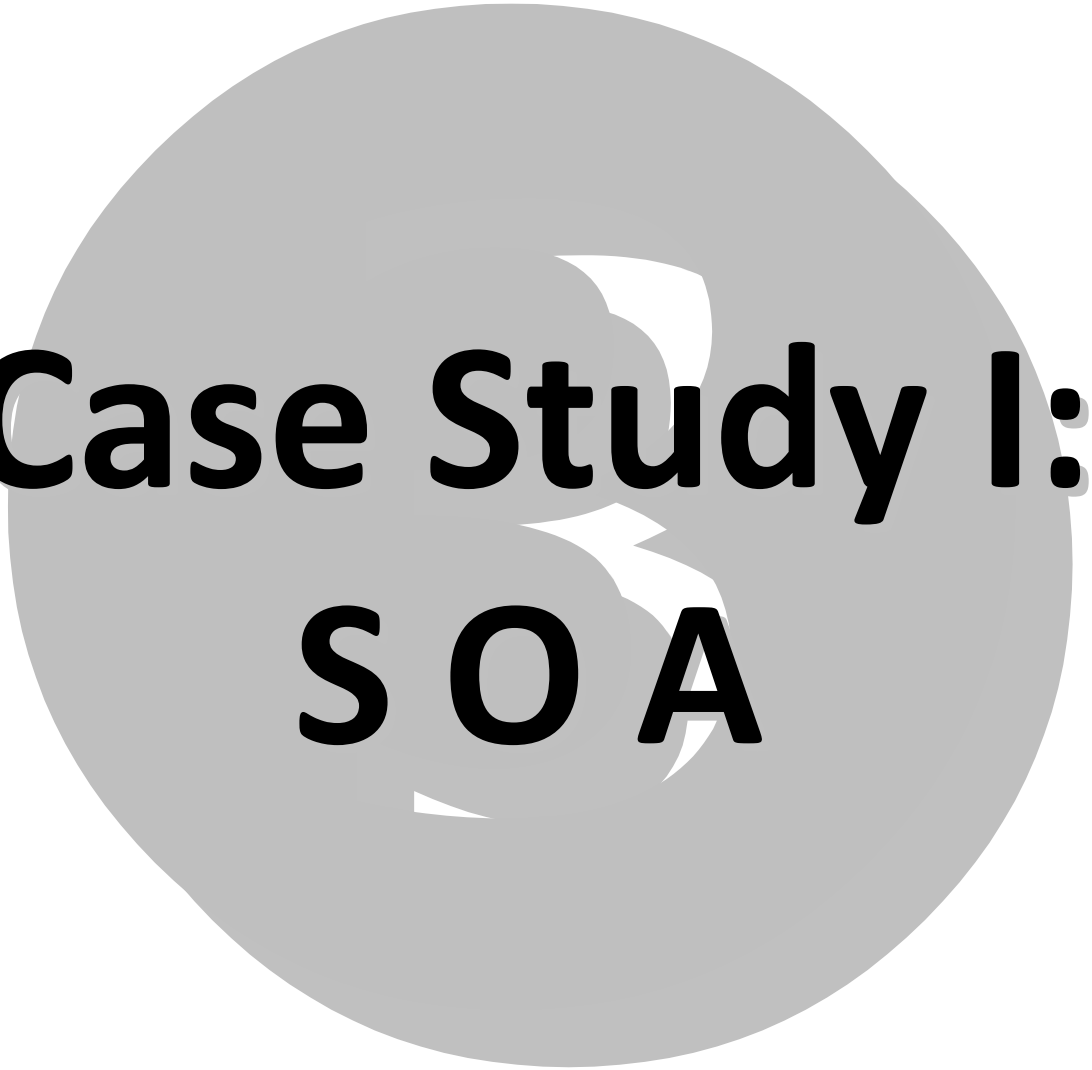
Don‘t Overspecify

Avoid Sideeffects


Iterate

Capture Best Practices

Measure



Case Study I: S O A


REST
XML
SLA
SCA
Business
JBI
Web Services
UDDI
HTTP
Driven
JMS

DEFINITIO

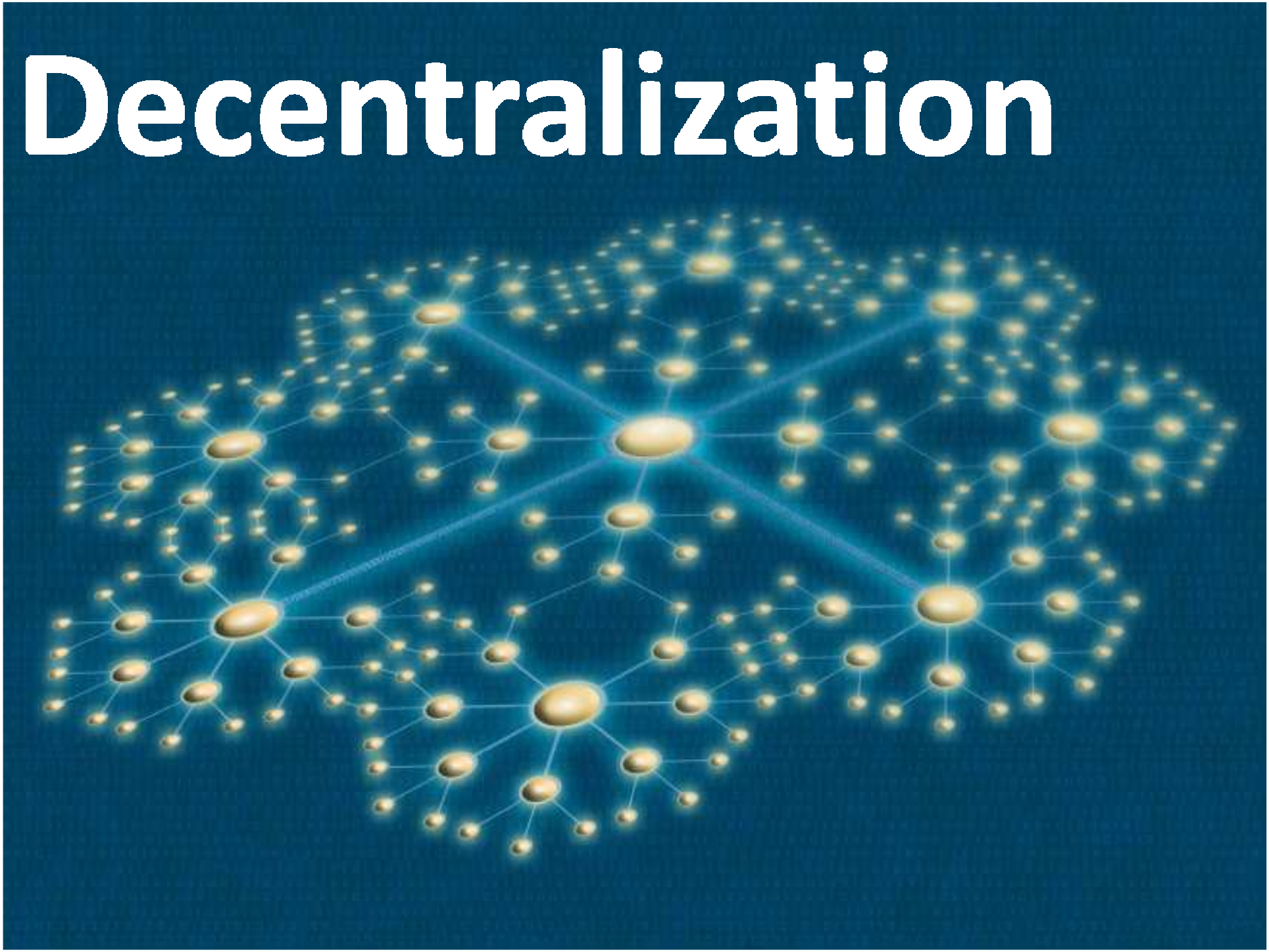
A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

OASIS

Distributed Responsibility

”Business Agility“

Decentralization



Decentralization

Service Ownership

Decentralization

Service Ownership

Technical Decisions

Decentralization

Service Ownership

Technical Decisions

Operation

Operate

Independently

Evolve

Independently

Shut Down

Independently

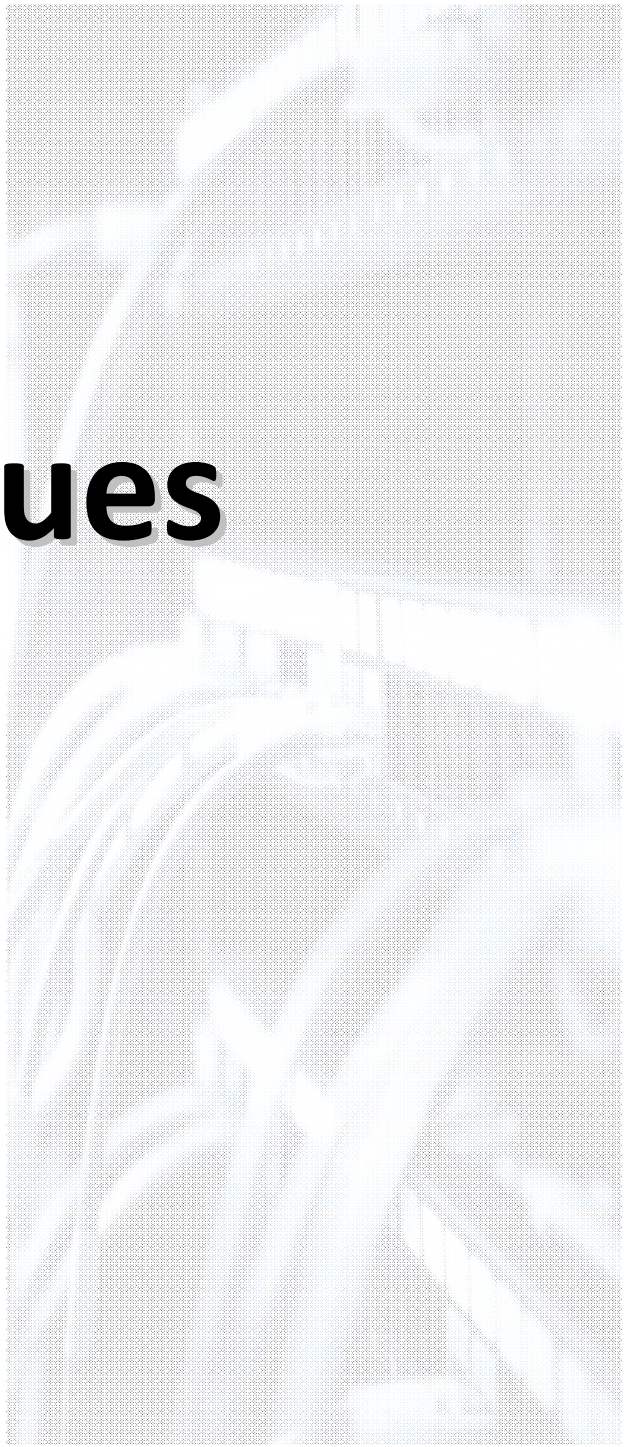
Restart

Independently

Decoupling

Decoupling

Message Queues



Decoupling

Message Queues

No rigid Data Structs

Decoupling

Message Queues
No rigid Data Structs
ESB (convert data)

Decoupling

Message Queues
No rigid Data Structs
ESB (convert data)
Compensating Tx

Deferred Consistency

Batch jobs

Deferred

Consistency

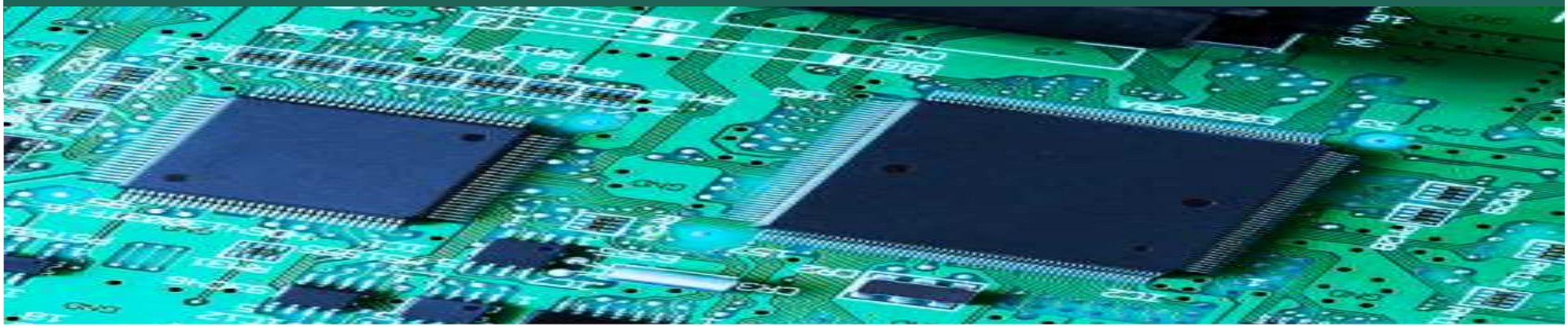
Batch jobs

Delayed replication

Deferred consistency


Batch jobs
Delayed replication
Session death
Deferred
consistency

Modularize



Modularize

Break Down

Modularize

Break Down

Clear Responsibilities

Modularize

Break Down

Clear Responsibilities

CRC

Modularize

Break Down

Clear Responsibilities

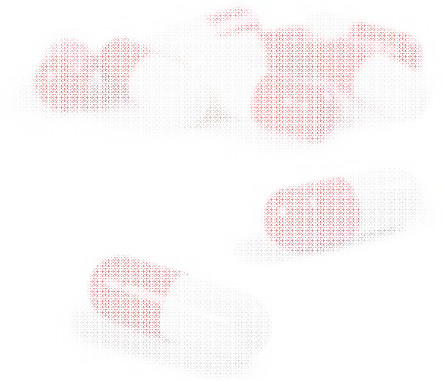
CRC

Reuse

Encapsulate

Encapsulate

Keep Impl Private



Indirection

Indirection

Logical Names

Indirection

Logical Names

ESB (conversion)

Indirection

Logical Names
ESB (conversion)
Proxies (monitoring)

Discovery

Discovery

Lookup based on Props

Discovery

Lookup based on Props

Load Balancing

… still keep things **reliable**

Contracts

SLA Contracts

ConSLActs

Orchestration

SLA

Contracts

Orchestration

Interfaces (DBC)

Interoperab
ility

Maintainab

ility

Platforms

Platforms

ESBs

JEE

SCA

Isolate Technology

Isolate

Technology

WSDL

SDO

SCA

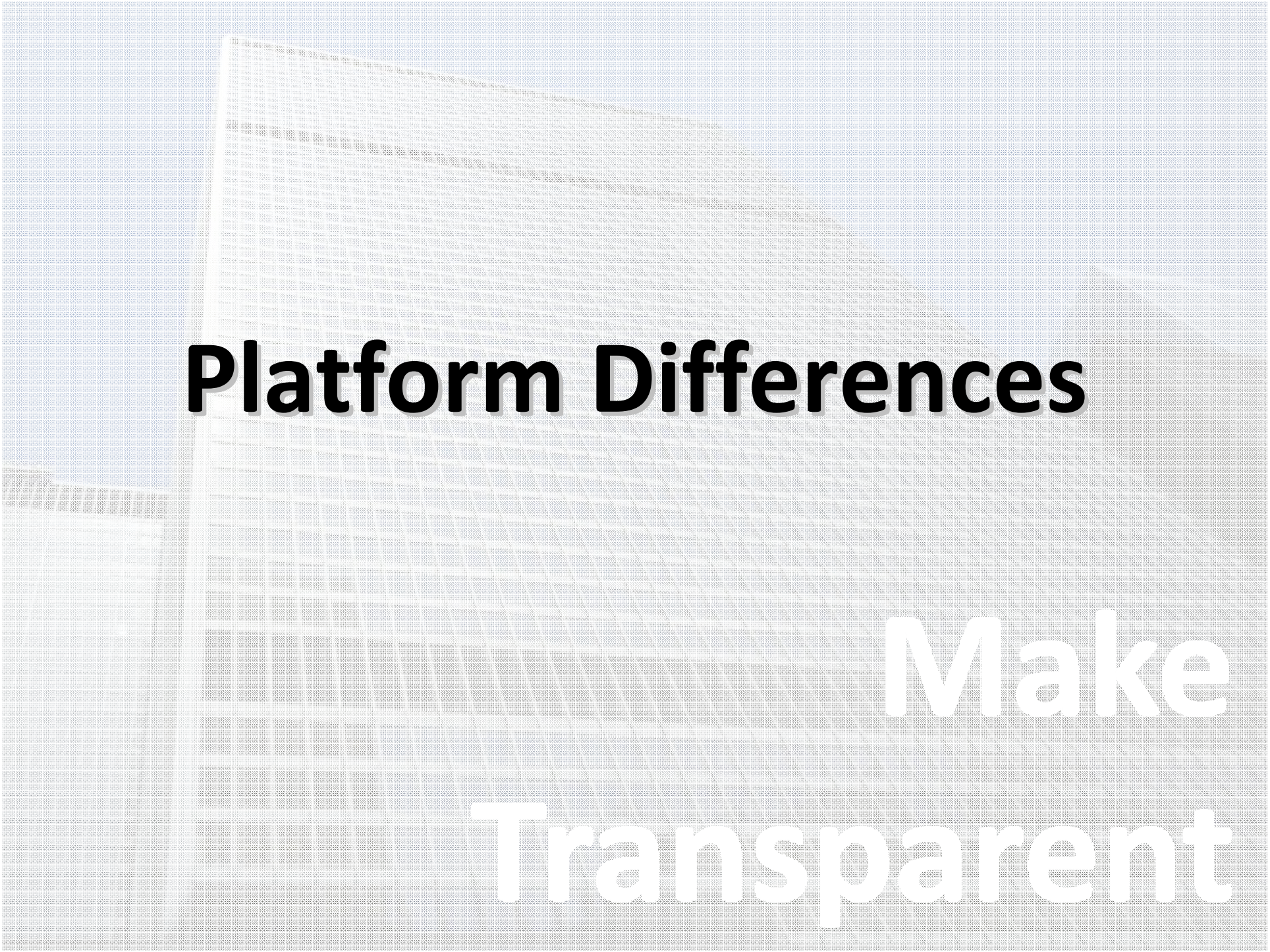
MDSD

Developers should not have to care about

different platforms


Make
Transparent

Platform Differences



Platform Differences

Data Conversion

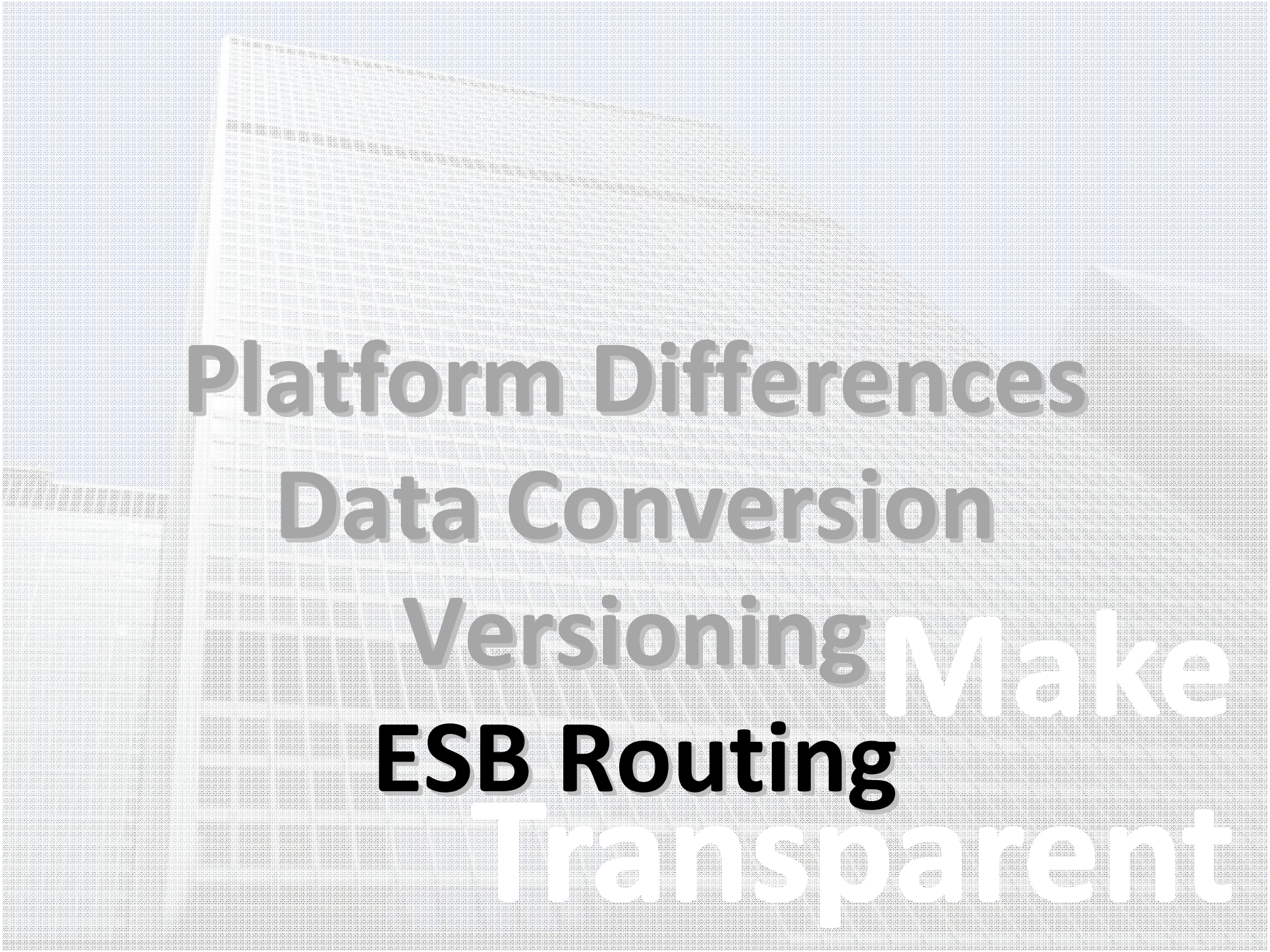
Make Transparent

Platform Differences

Data Conversion

Versioning

Make Transparent


Platform Differences
Data Conversion
Versioning
Make
ESB Routing
Transparent

Developers need to be aware of the

distributed nature

of the overall system.

Make Explicit

Service Calls

Make Explicit

Service Calls Potential Distribution

Service Calls

Make Explicit

Potential Distribution

Ownership Transfer

Keep the system running smoothly...

Measure

Service Use (GC, Billing)

Measure

Service Use (GC, Billing)

Load (Scaling)

Measure

Service Use (GC, Billing)

Load (Scaling)

Performance

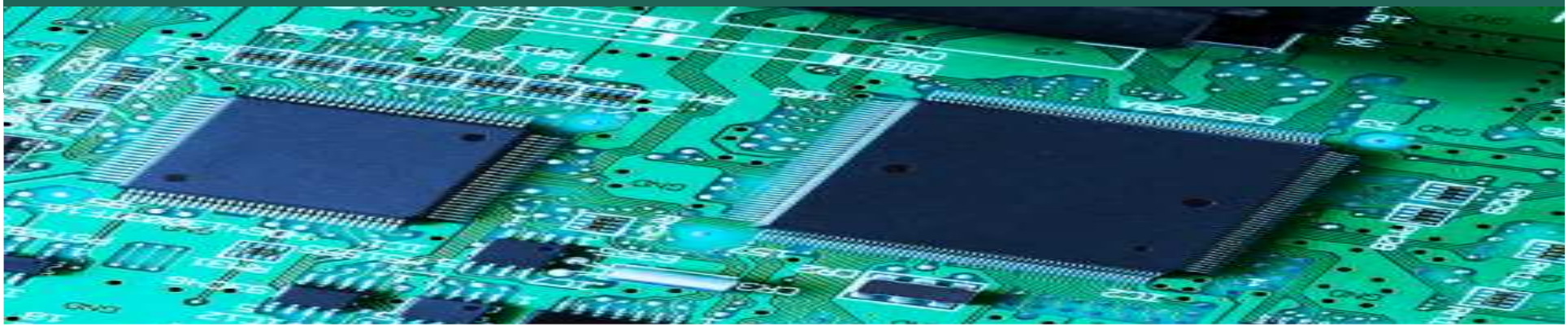
Measure

Case Study II: Concurrency

IMPORTANT!

Traditional Approach

Modularize



Modularize

More manageable concurrent Units „Smaller Problems“

Encapsulate

Encapsulate

Encapsulate inside module – don‘t expose!

„Thread-safe interface“

Platforms

Platforms

Servlets

Platforms

Servlets

EJB

Platforms

Servlets

EJB

COM+ (Apartments)

… and finally

API

Contracts

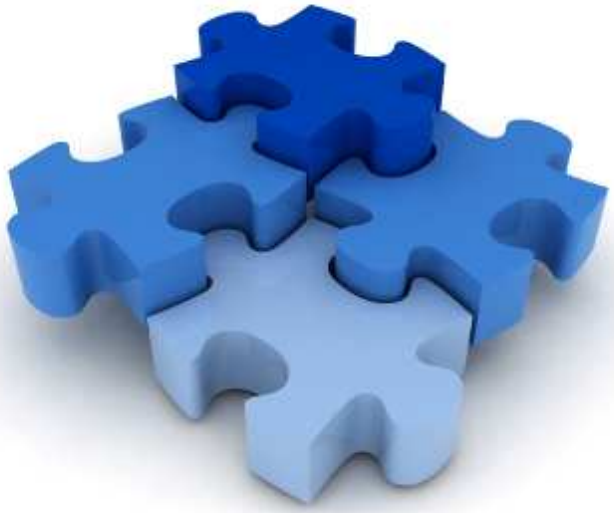
Thread safety

Thread safety

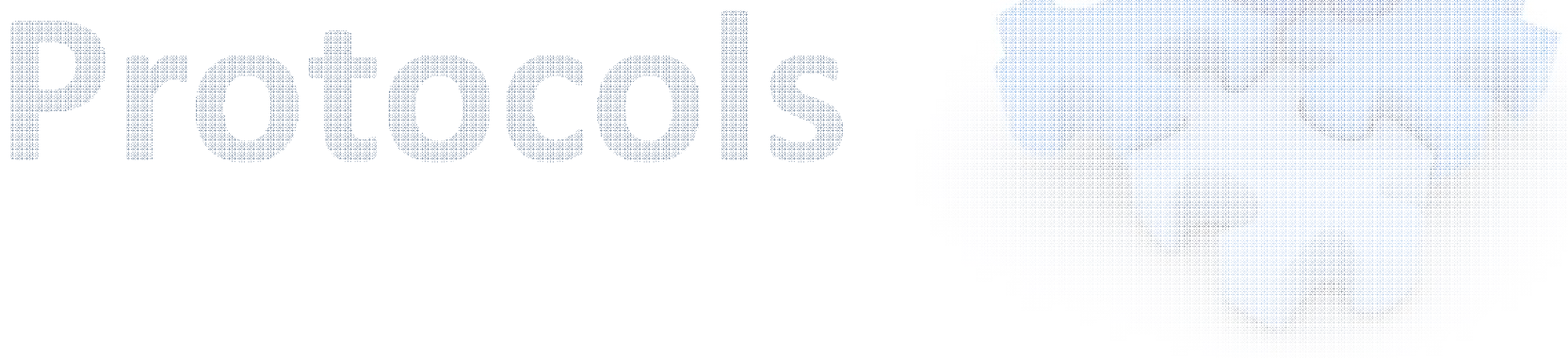
Contracts

Locking APIs

Protocols



Rely on user to „do the right thing“



Rely on user to
„do the right thing“

Protocols

Acquire/Release

Rely on user to
„do the right thing“

Protocols

Acquire/Release

Wait/Notify

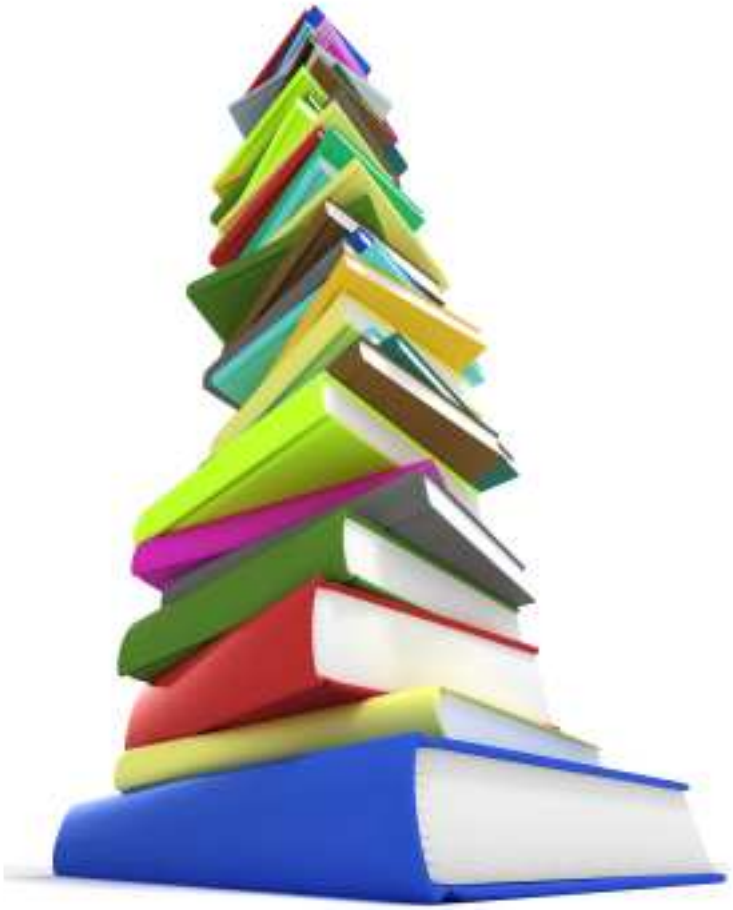
Build more

powerful

abstractions

from basics.

Standard Library



Standard

Semaphores

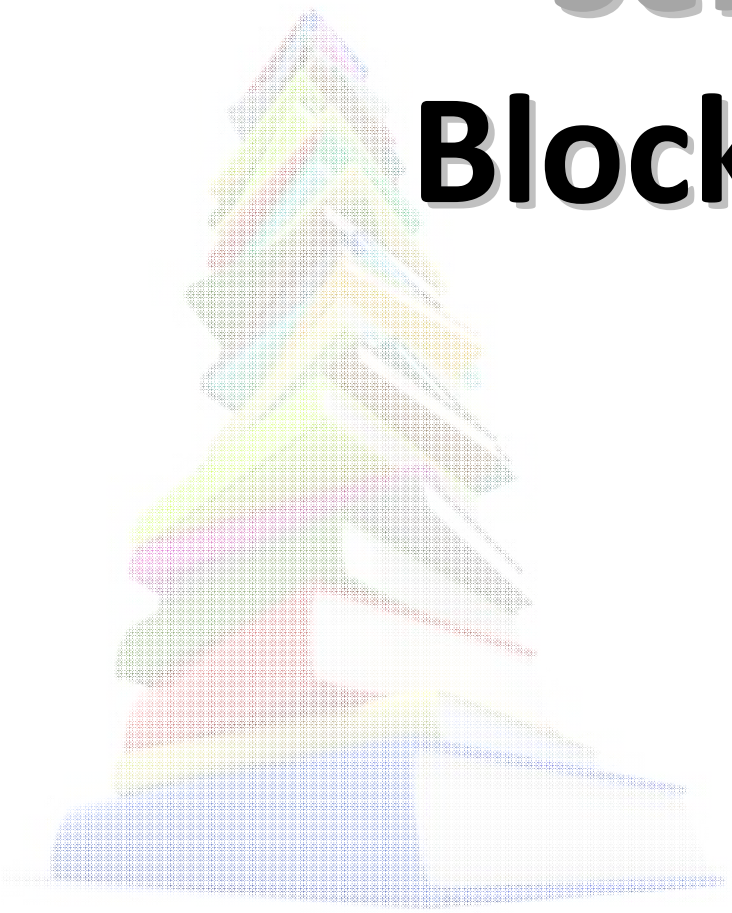
Library

Standard

Semaphores

Library

Blocking Queues

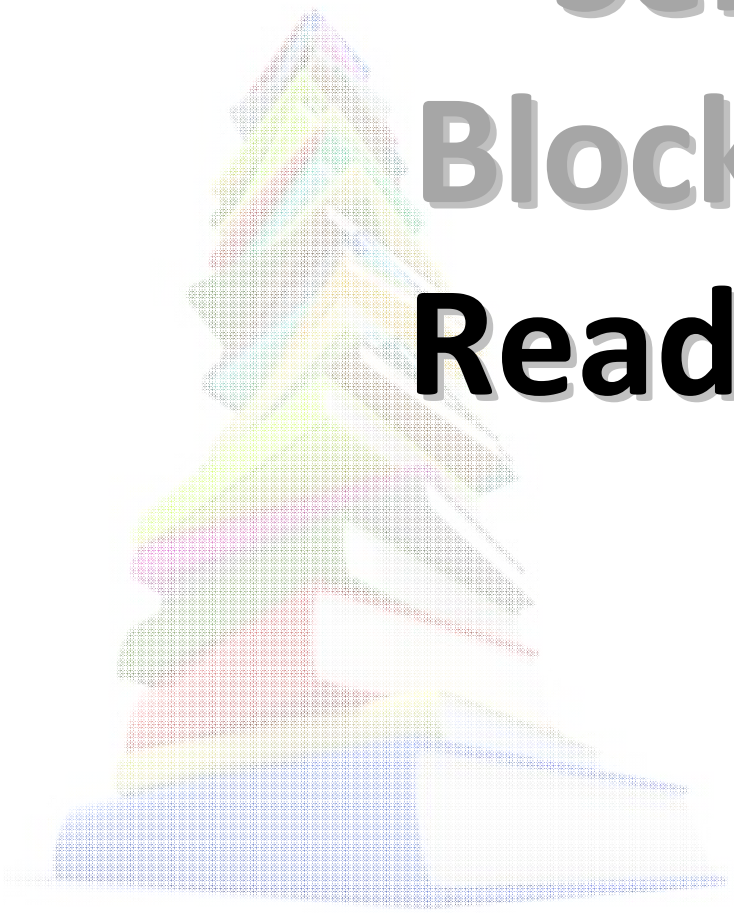


Standard Library

Semaphores

Blocking Queues

Read/Write Lock



Standard Library

Semaphores

Blocking Queues

Read/Write Lock

Monitors

Bootstrapping

Blocking Queue uses Monitors

Blocking Queue uses Monitors uses Semaphores

More recent trends

Declaration

Implementation

Tx Memory
(I want this atomic, don‘t care about locks)

Don‘t Overspecify

Tx Memory (what, not how)

Don't

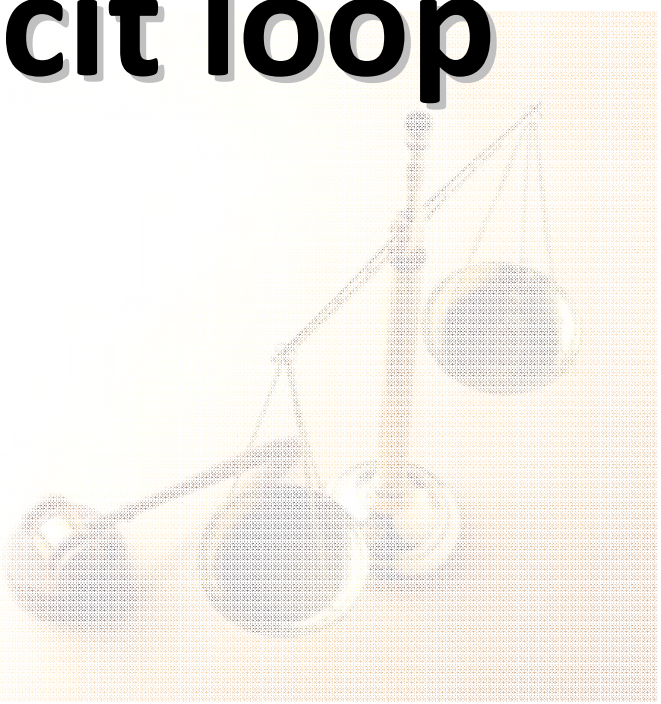
Overspecify

Tx Memory (what, not how)

Don't

Foreach vs. explicit loop

Overspecify



Tx Memory (what, not how)

Don't

Foreach vs. explicit loop

Overspecify

foreach seq(1..n)

Avoid Sideeffects

Avoid Sideeffects

Functional

Avoid

Sideeffects

Functional

Stateless

Avoid

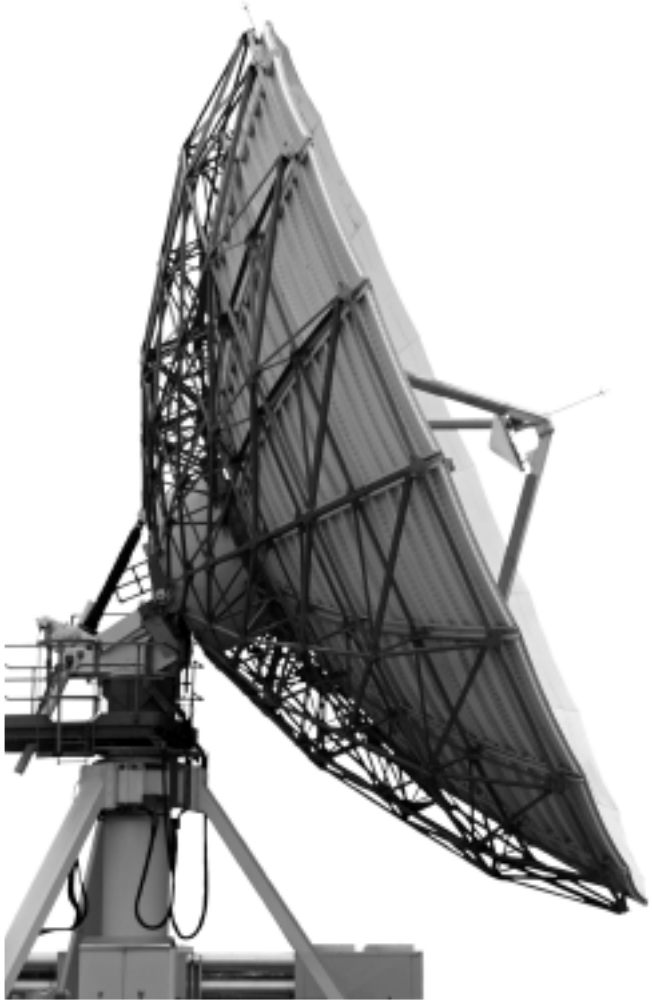
Sideeffects

Functional

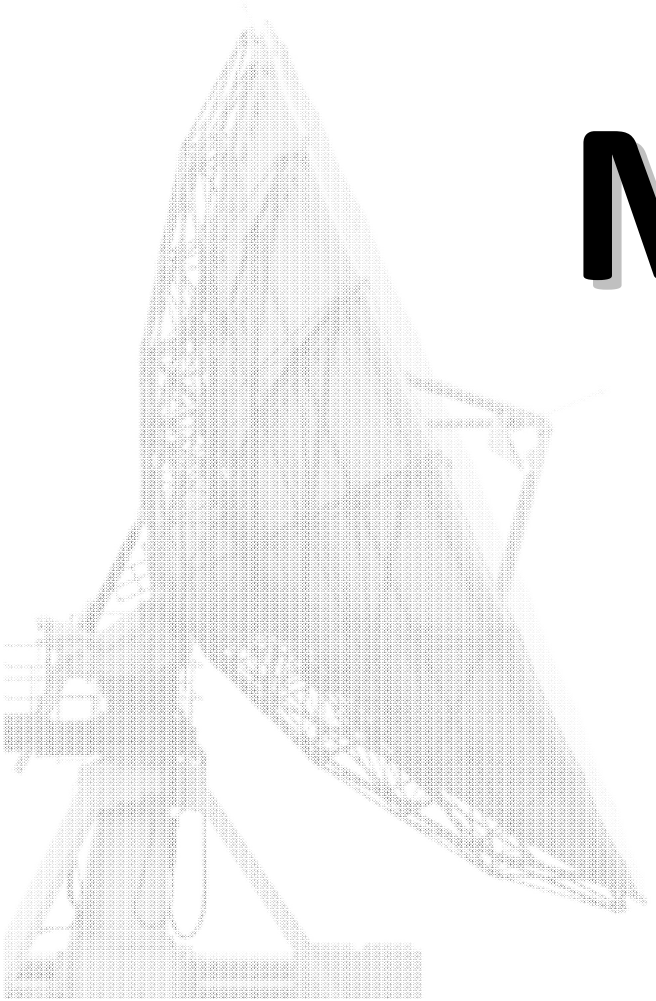
Stateless

Immuntable

Tracking



Tracking



Monads

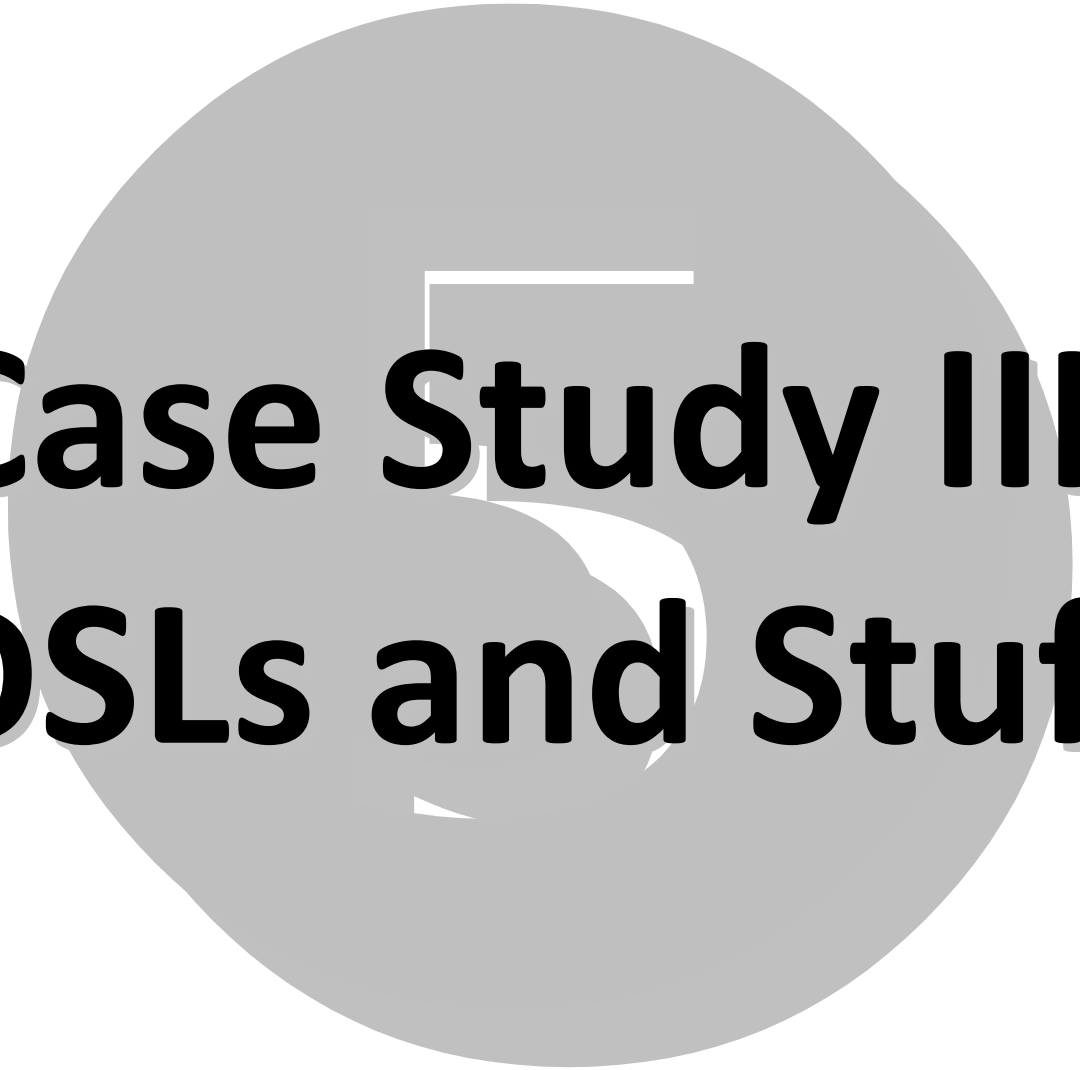
Automate

Automate

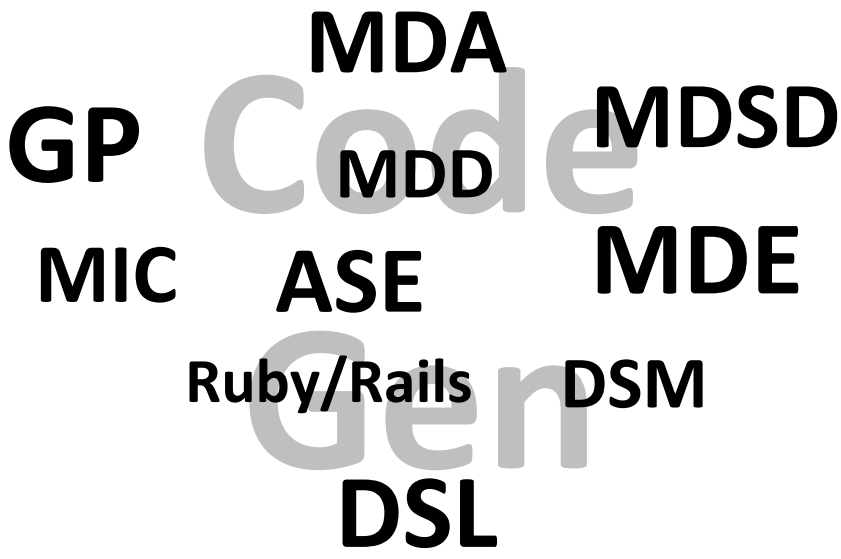
Tx Memory

≅

Garbage Collection



Case Study III: DSLs and Stuff


MDA
GP
Code
MDD
MDSD
MIC
ASE
MDE
Ruby/Rails
Gen
DSM
DSL

DEFinition

A DSL is a **focussed**, **processable language** for describing a specific **concern** when building a system in a specific **domain**. The **abstractions** and **notations** used are natural/suitable for the **stakeholders** who specify that particular concern.

capture

domain

concepts


Abstraction

Domain Model

Abstraction

Domain Model

Domain Analysis

Abstraction

Domain Model
Domain Analysis
Domain Expert Interviews

Abstraction

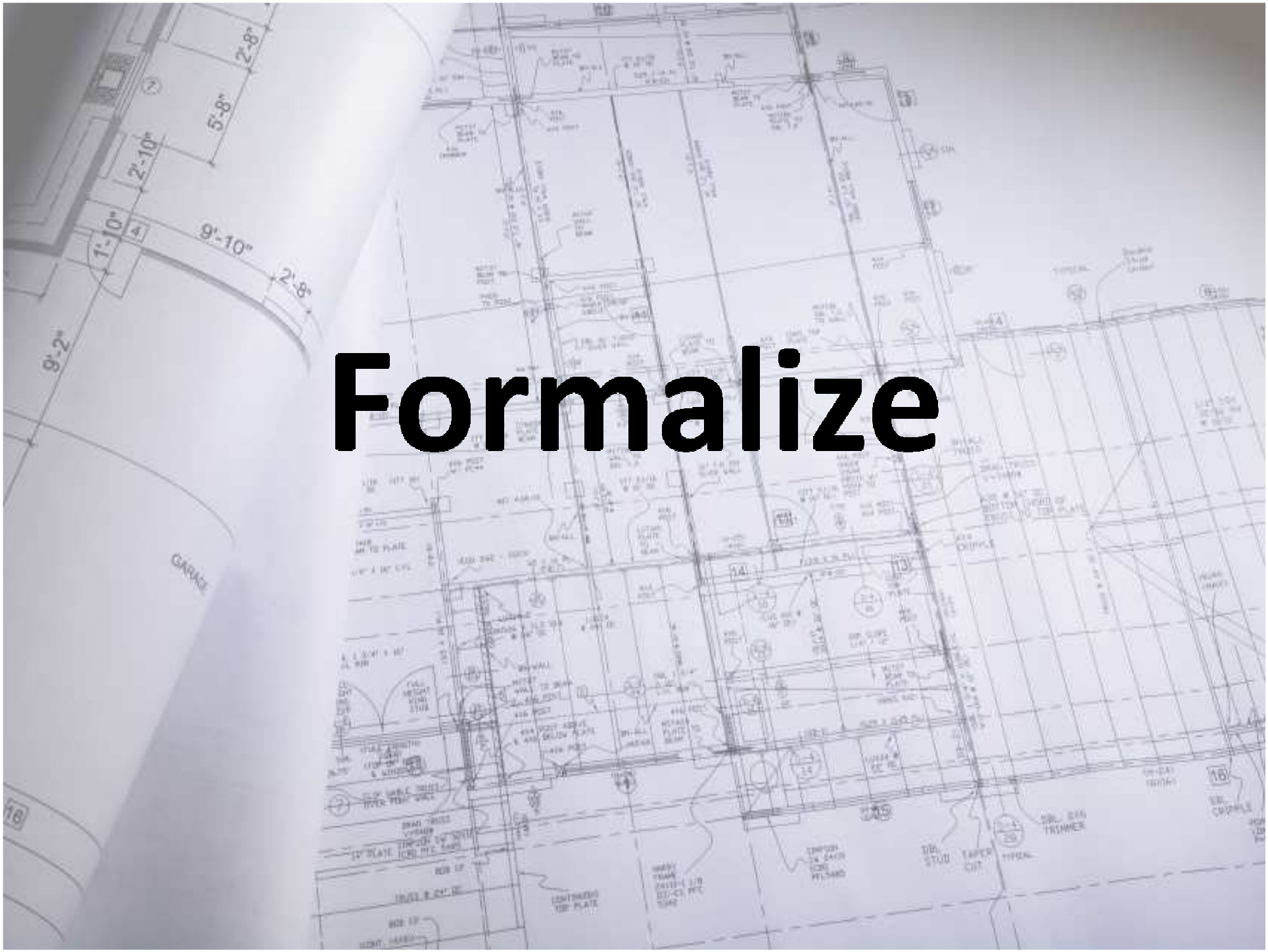
Domain Model

Domain Analysis

Domain Expert Interviews

Abstraction

APIs


Formalize

Metamodel

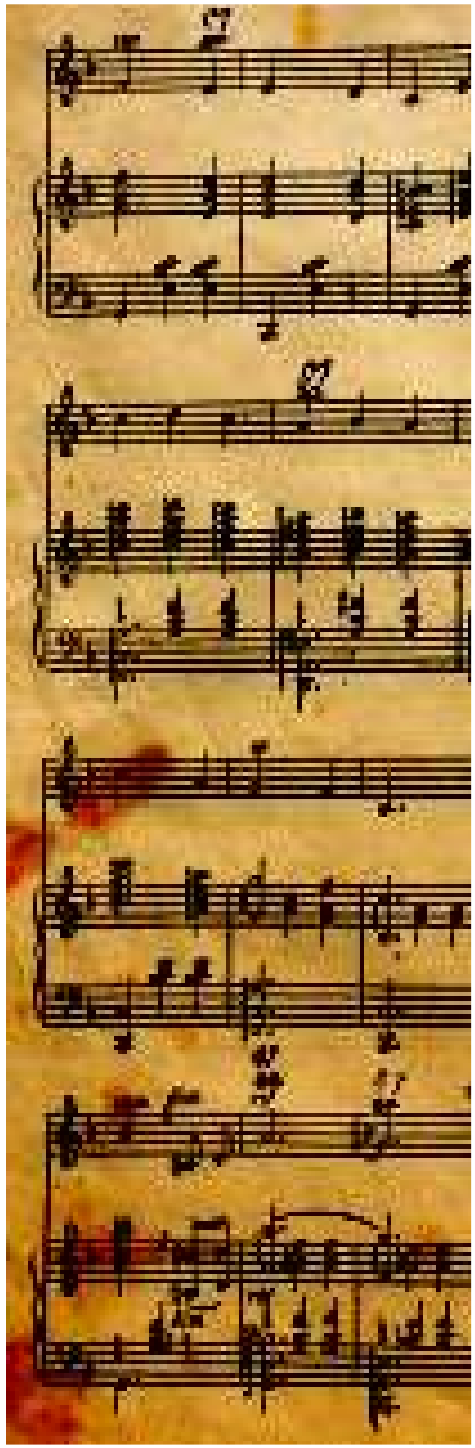
Formalize

Metamodel

Formalize

Abstract Syntax

Notation

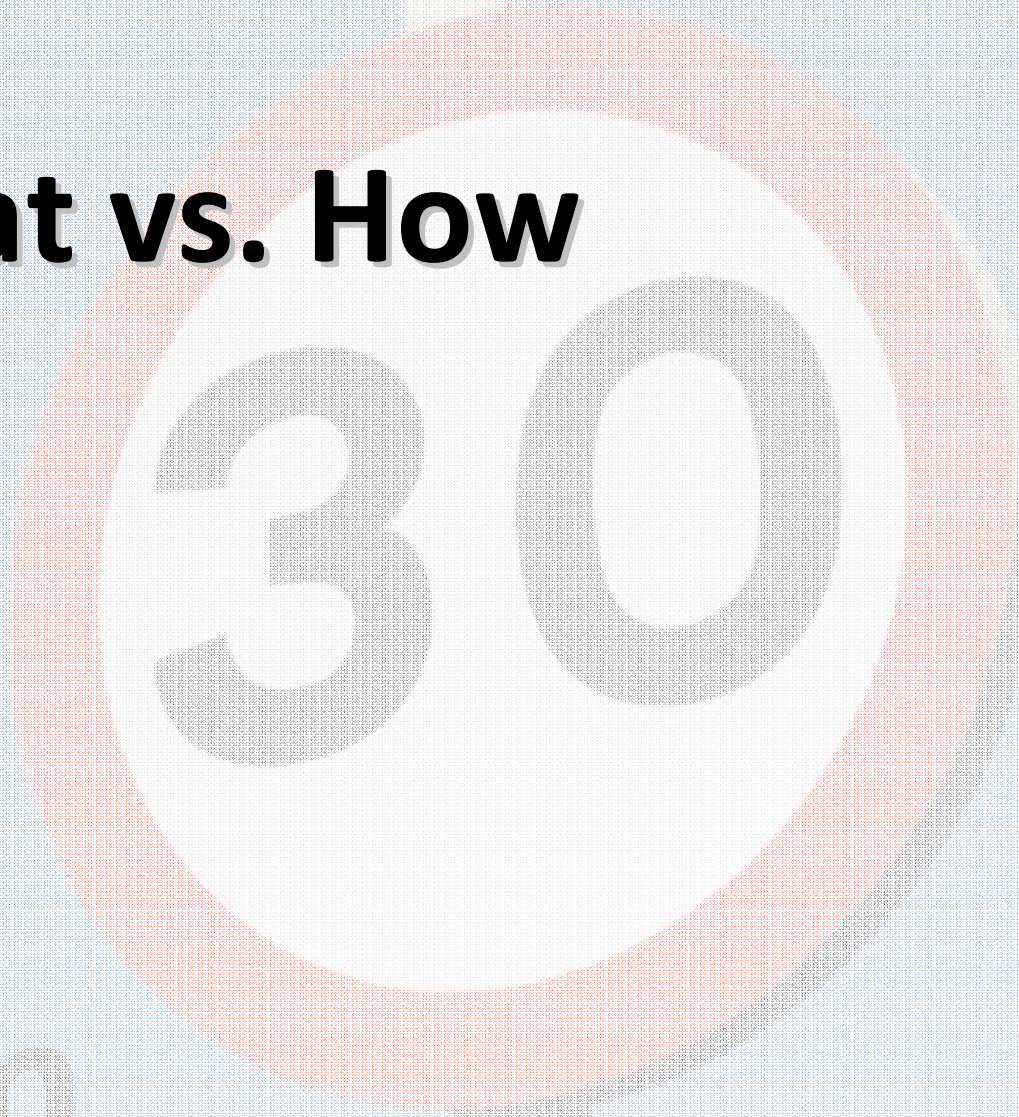


Concrete Syntax
Graphical/Textual

Notation


30
Limit
Freedom

What vs. How



limit

freedom

What vs. How

Generated Skeletons

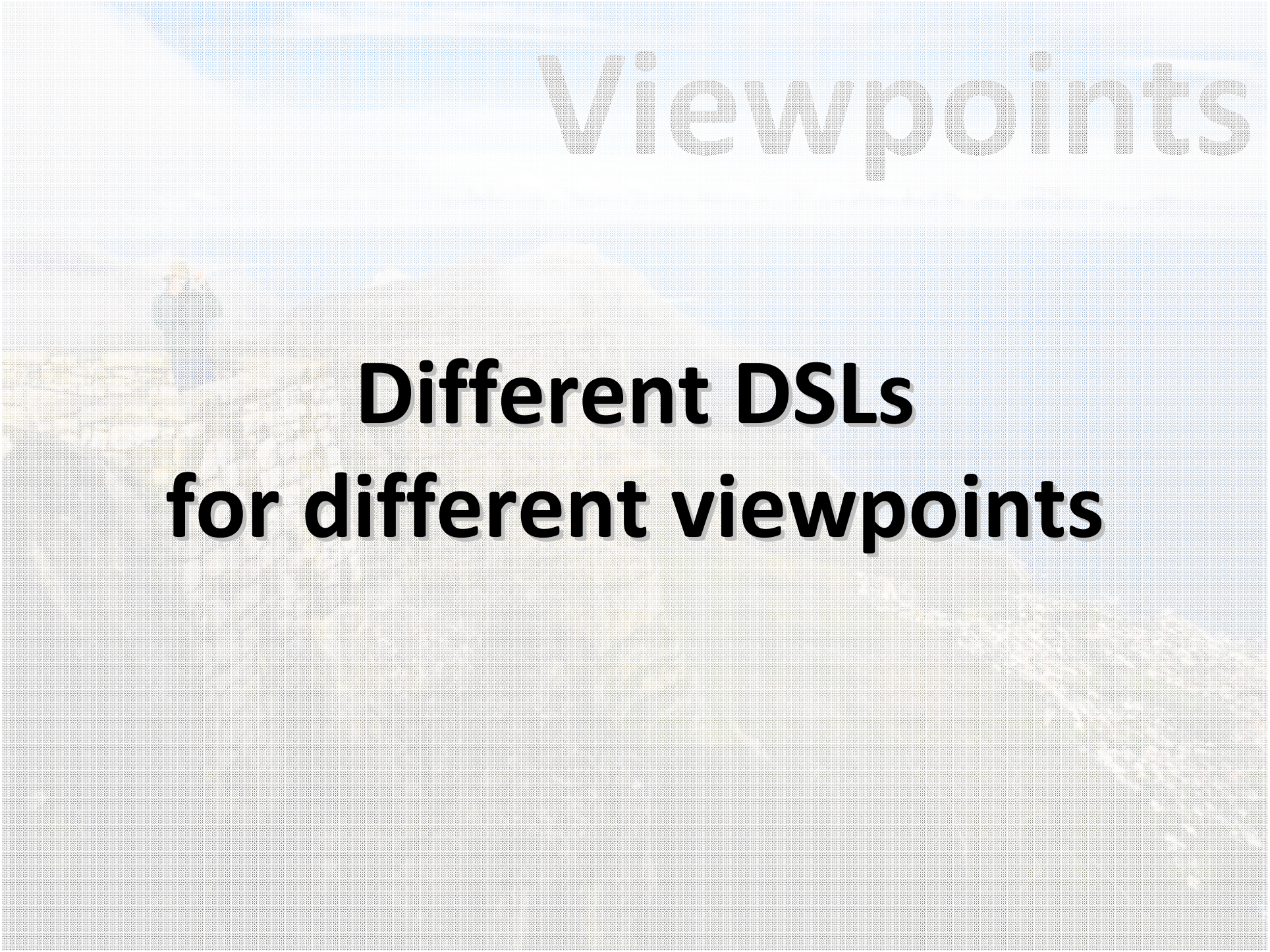
What vs. How

Generated Skeletons

Architecture Enforcement

What vs. How
Generated Skeletons
Architecture Enforcement
Pattern „Implementation“


Viewpoints


Viewpoints
Different DSLs
for different viewpoints

Platforms

Platforms

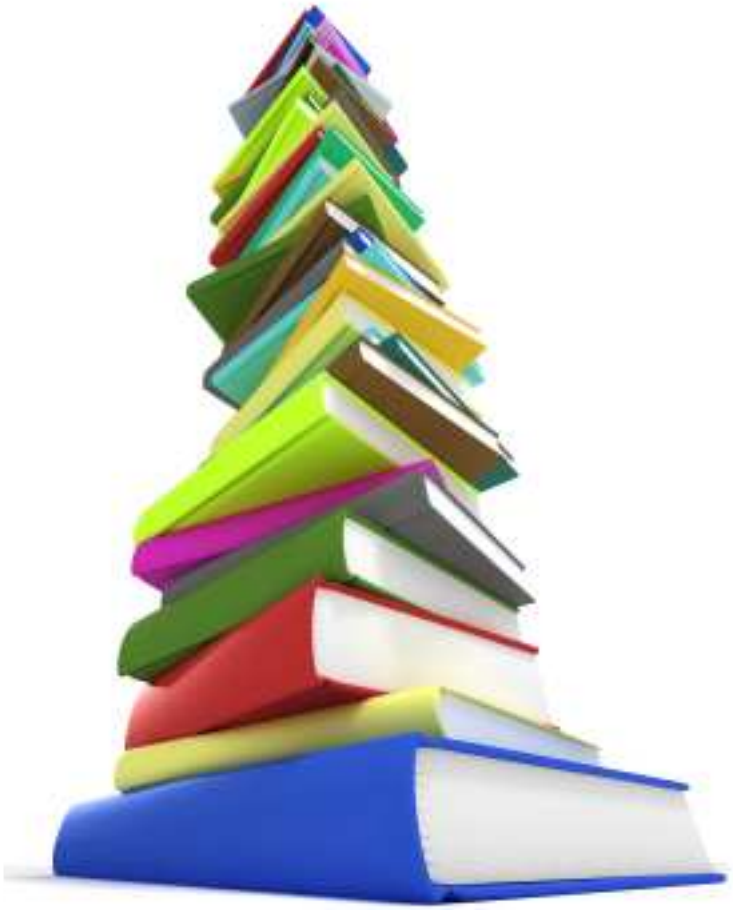
Generate Code based on a Platform

Bootstrapping

Use Language tool to build language tool

Ecore.ecore

Standard Library



Standard library

Language that can define abstractions itself

„grow the language“


Go Meta

Constraints

Constraints
Generators

Interpreters

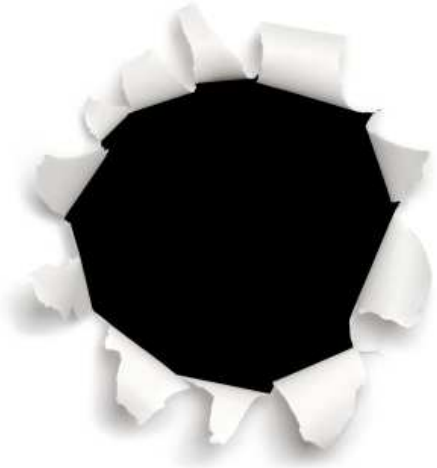
Translate

execute DSL program by translating to existing language

Interpret

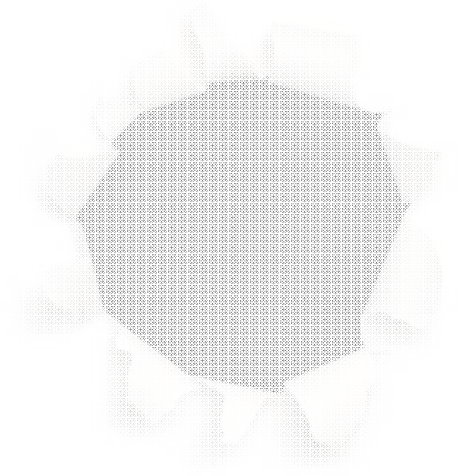
A (meta) program
That inspects a DSL
Program and executes
Side effects

Go Down



JAVA extensions

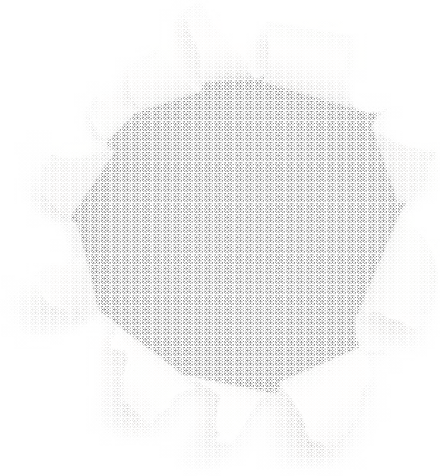
Go Down



JAVA extensions

Go Down

Protected Regions

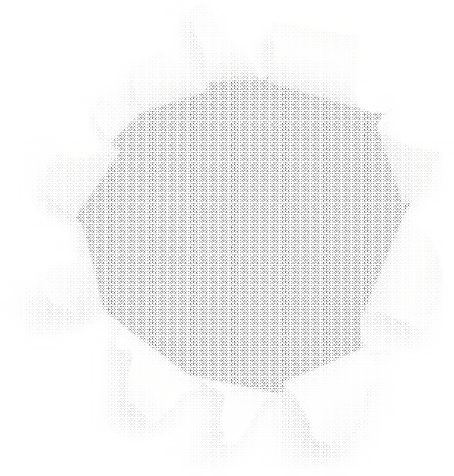


JAVA extensions

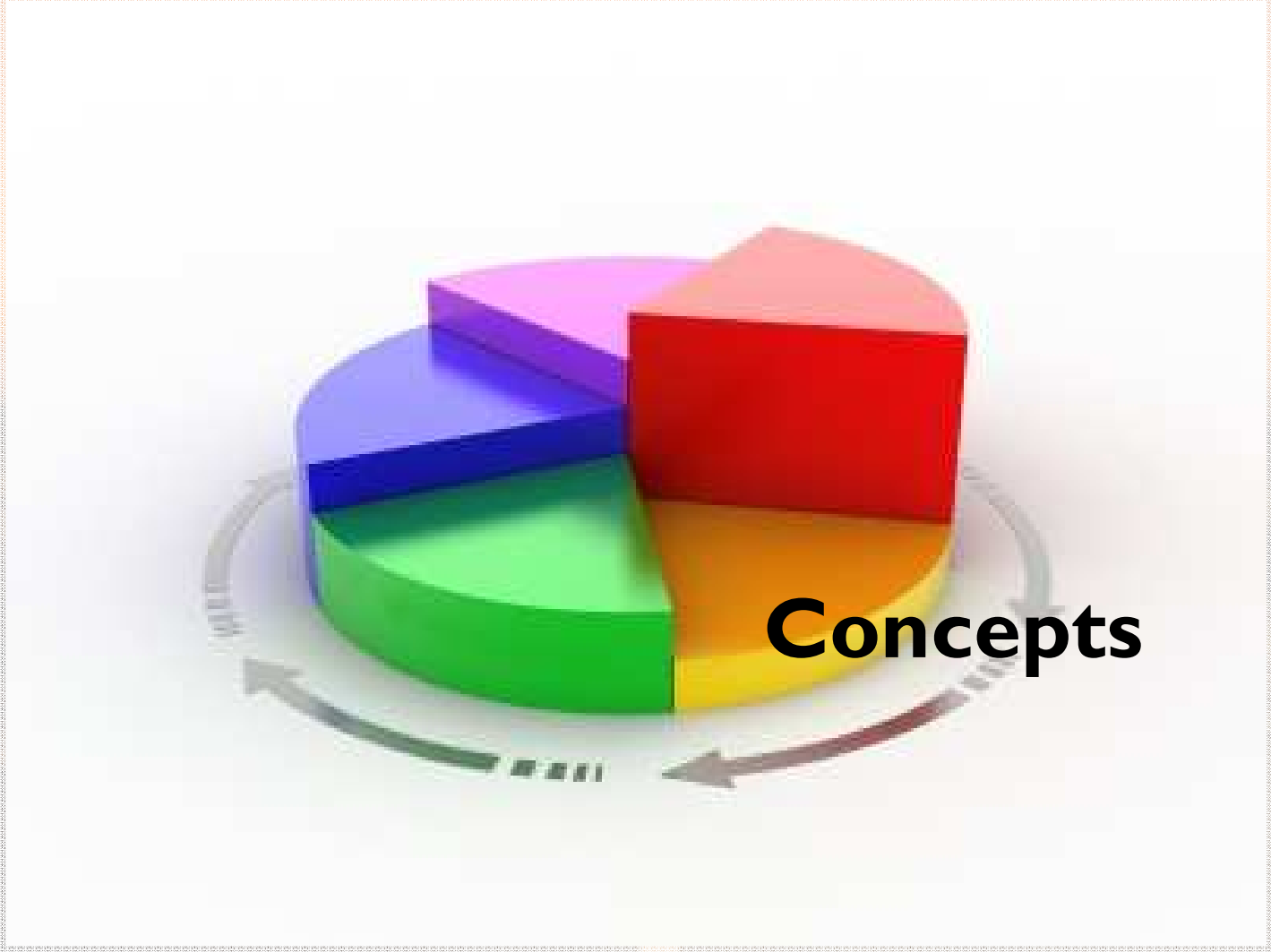
Go Down

Protected Regions

Interpreter-Plugins

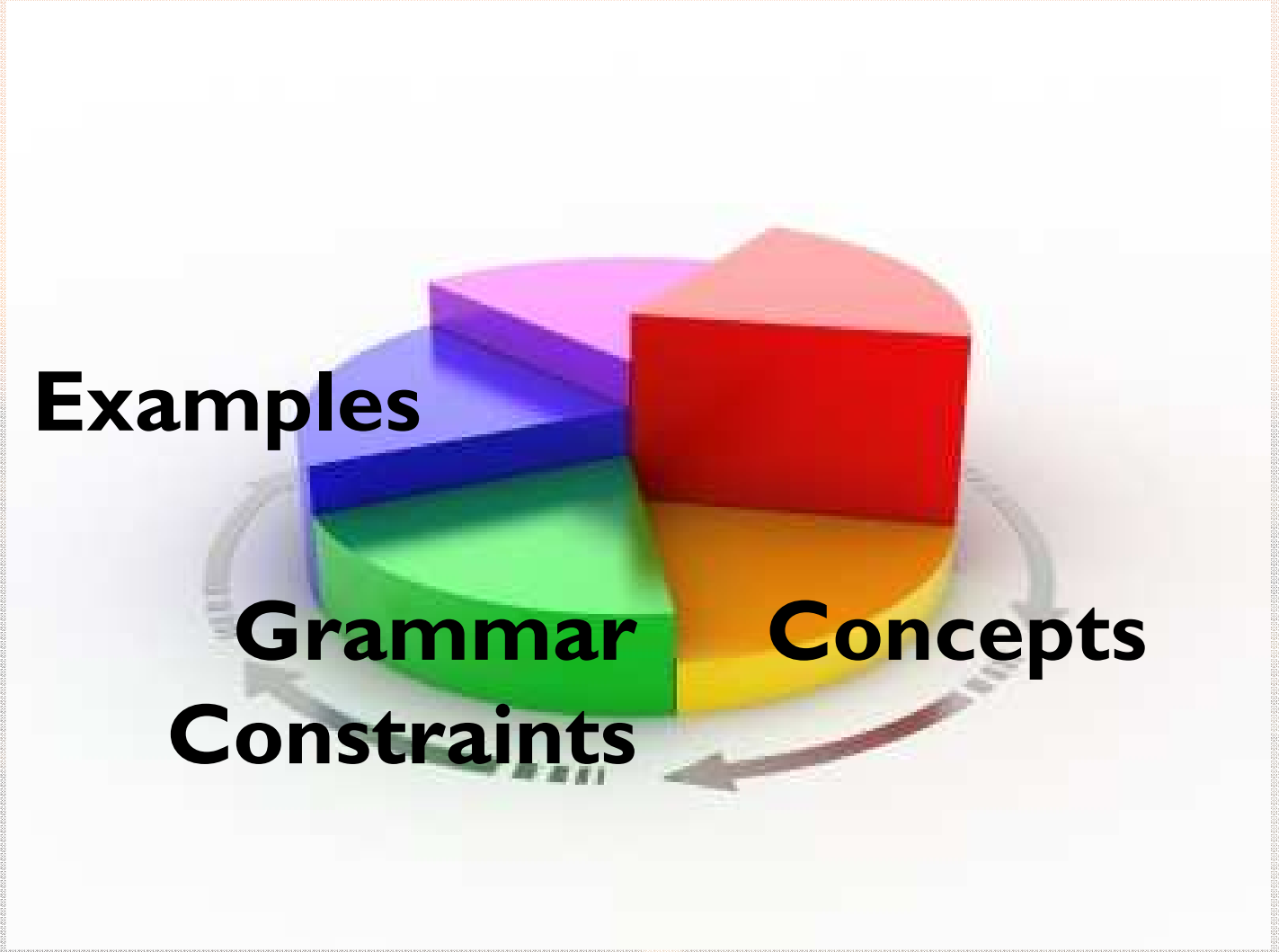


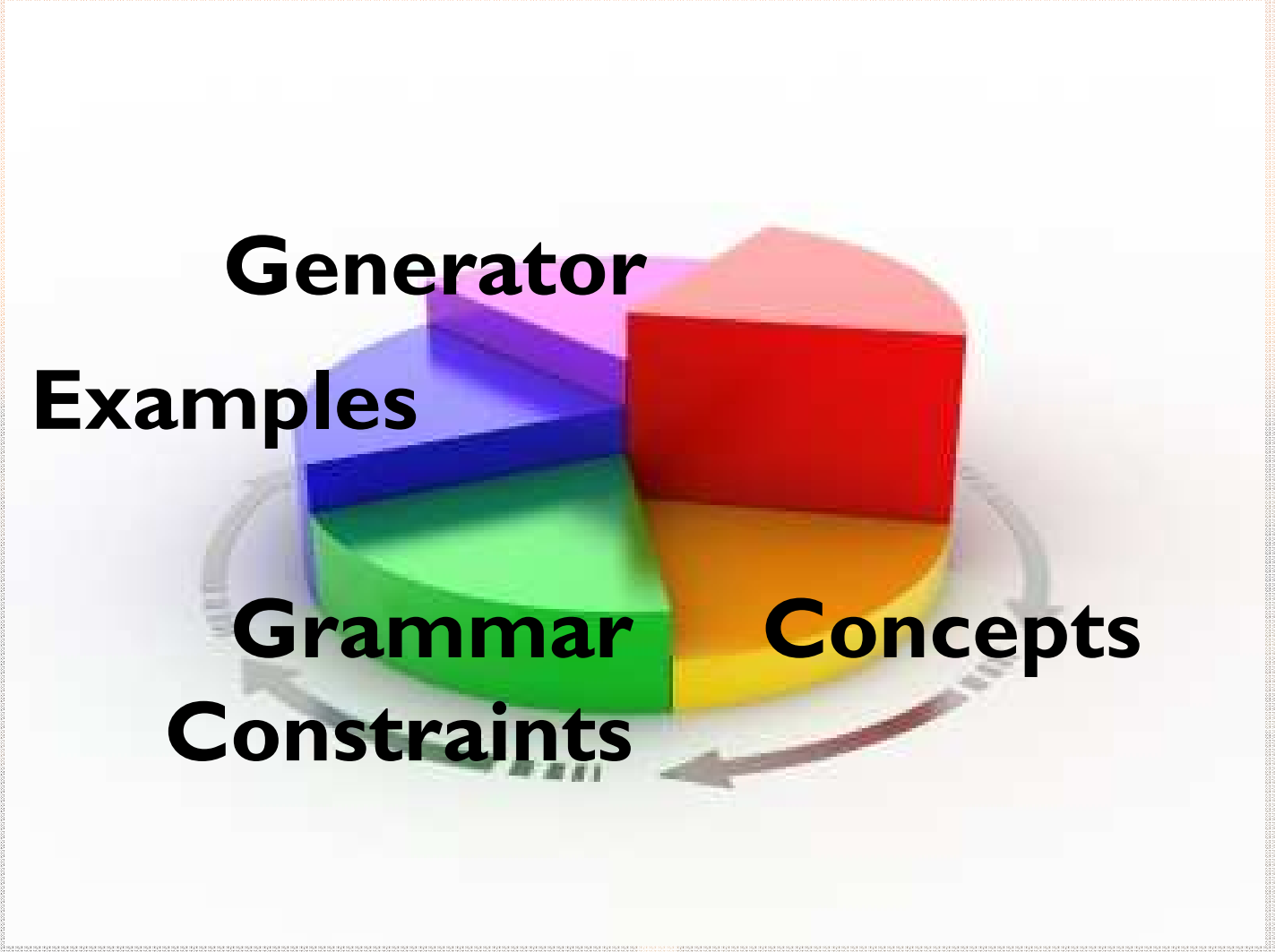

Iterate



Concepts


Grammar
Concepts
Constraints


Examples
Grammar
Concepts
Constraints

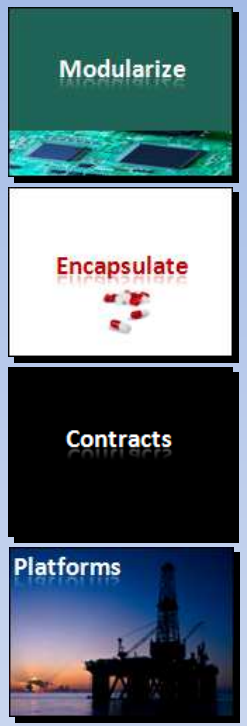

Generator
Examples
Grammar
Concepts
Constraints


Generator
Editor Cust.
Examples
Grammar
Concepts
Constraints

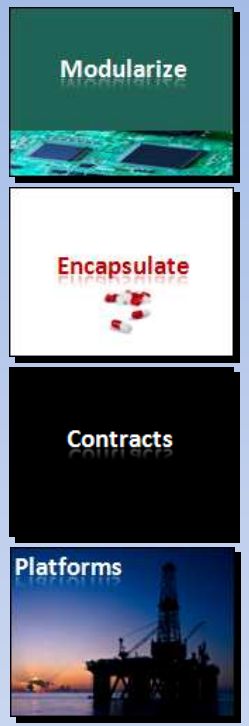

Recap

SOA

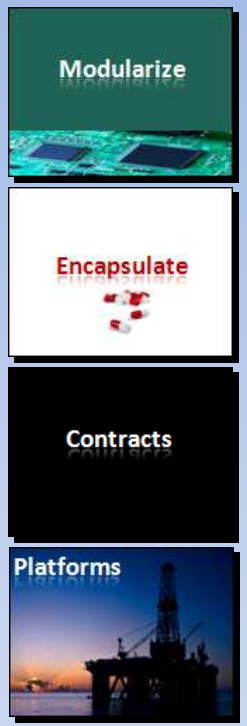
SOA



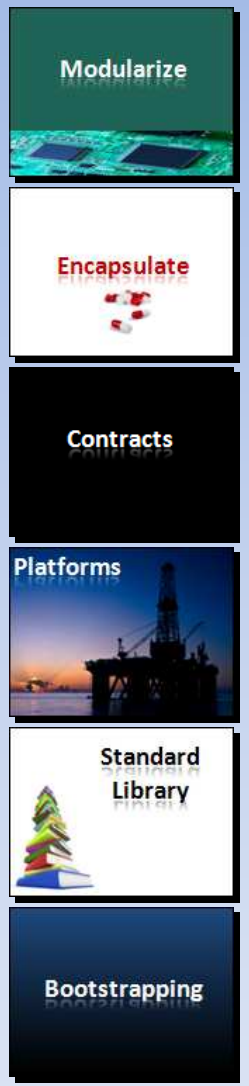
Concurrency



SOA



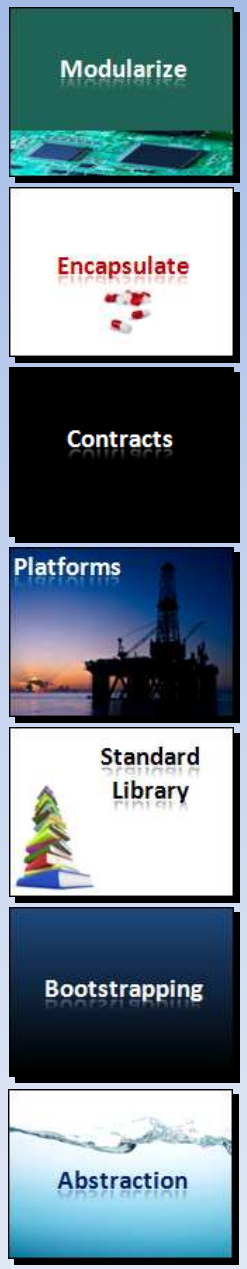
Concurrency



DSLs

SOA

Concurrency



DSLs


awareness
when building systems
Goals


checklists
for reviewing systems
Goals


education
of developers and architects
Goals

THE END.

Thank you.

Questions?