(Six) Developer Best Practices

Looking beyond the hype

Markus Völter
(voelter@acm.org)

About

Markus Voelter
Independent Consultant

About

Markus Voelter
Independent Consultant

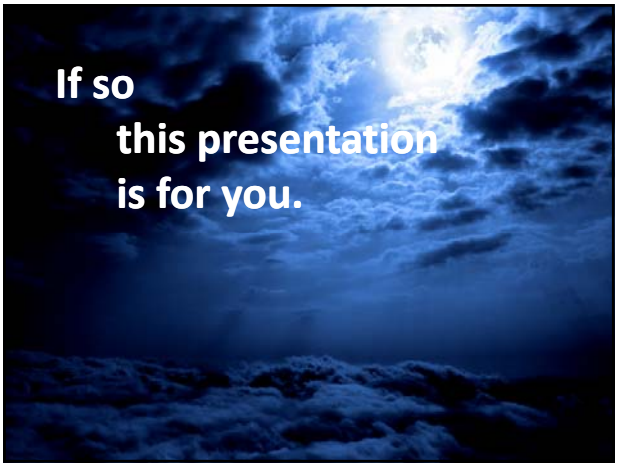
Software Architecture
DSLs & MDSD
Product Lines

About

Markus Voelter
Independent Consultant

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

1 Introduction



People often talk about technologies instead of core concepts.

Technology-discussions create a lot of accidental complexity.

This presentation:

1 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
Concurrency 4

This presentation:

Three Case Studies

3 SOA
4 Concurrency
5 DSLs and Stuff

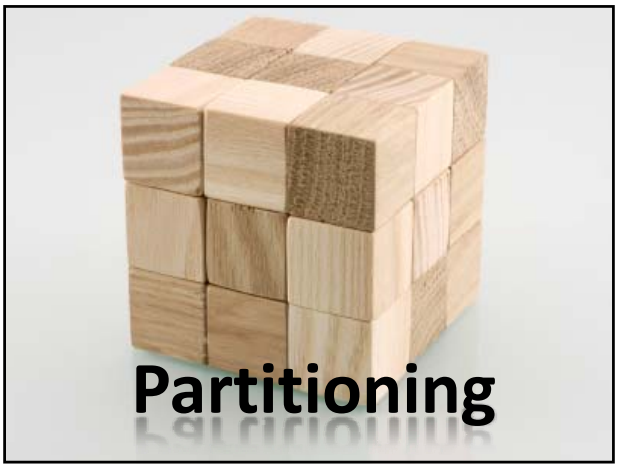
This presentation:

6 Recap at the End

2 Principles Quick Tour

Modularize

Encapsulate

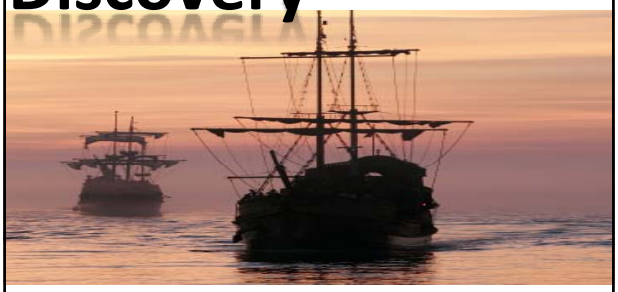


Contracts

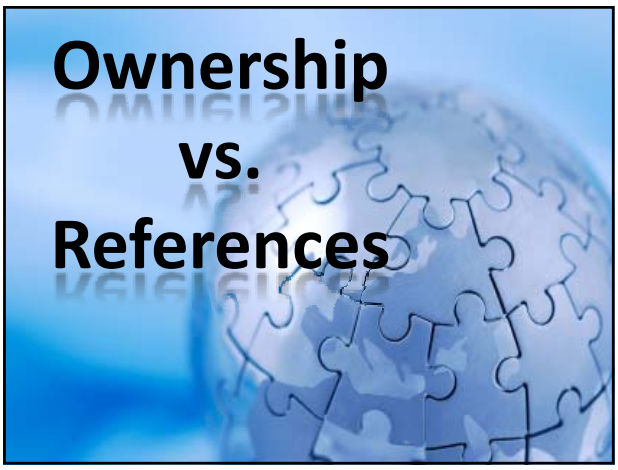
Decoupling

Indirection

Discovery



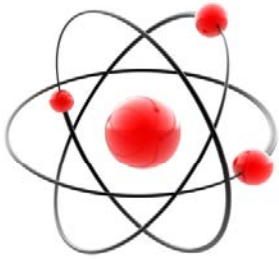
Ownership
vs.
References



Handle Crosscuts

Go Down

Isolate



Atomicity

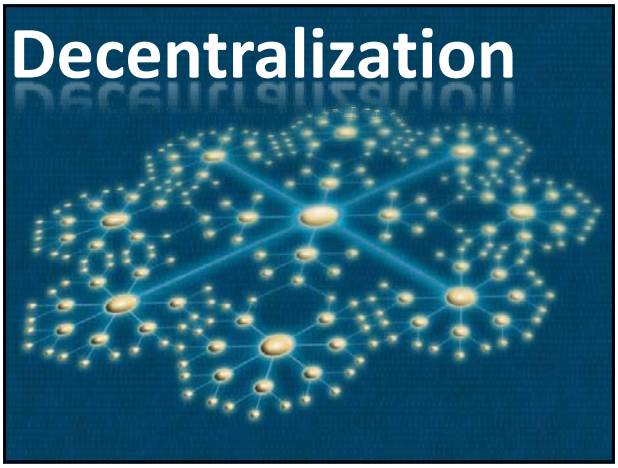
Isolate Technology

Parametrization

Staging

Simplicity

Decentralization



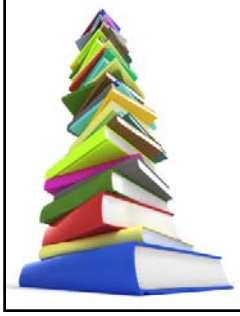
Open/Closed World

Distribution vs. Local Caching

Compensation

Deferred Consistency

Bootstrapping



Standard Library

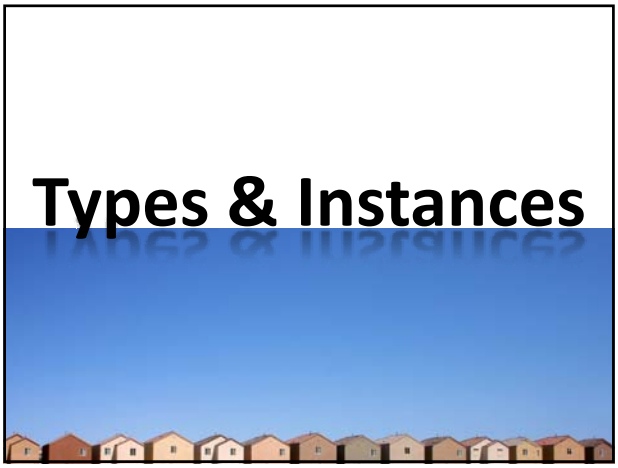
Platforms

Containers

Orthogonality

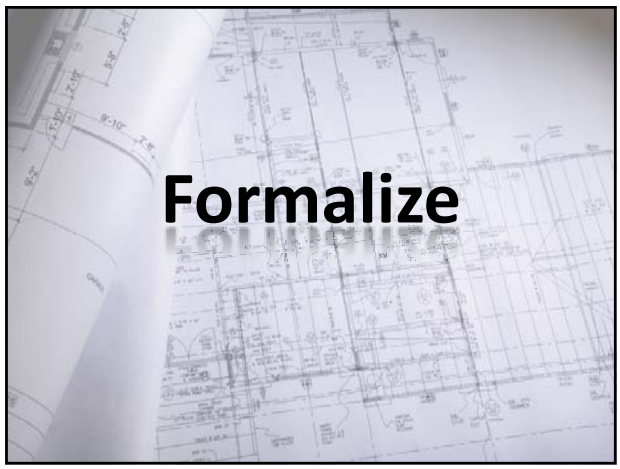

Identity


Abstraction


Types & Instances

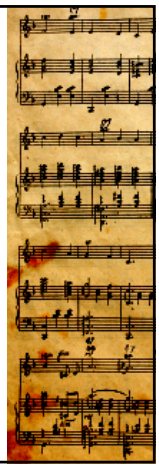

Hierarchical
Decomposition


Specialize

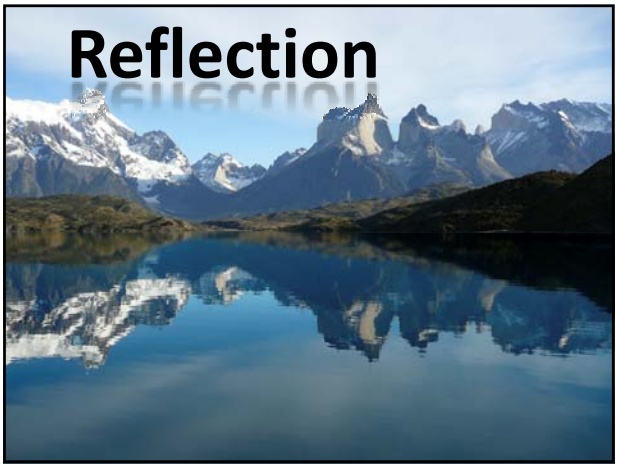

Formalize


Viewpoints

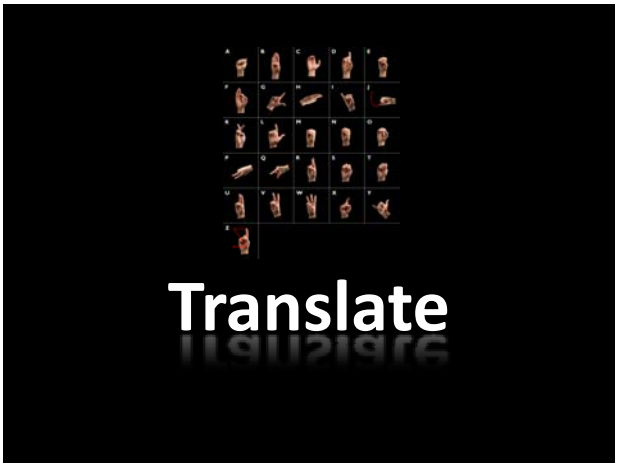
Notation




Go Meta

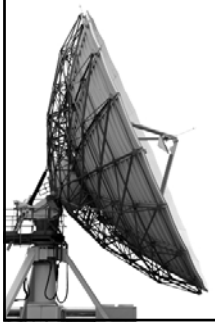

Reflection


Evolution/
Self Modification


Translate

Interpret

Tracking



Automate

Protocols

Make Explicit

Lazy vs. Eager

Declaration

Implementation

Don't
Overspecify

Avoid
Sideeffects

Capture
Best
Practices

Measure

[redacted]

3

Case Study I:
S O A

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

DEFINITION

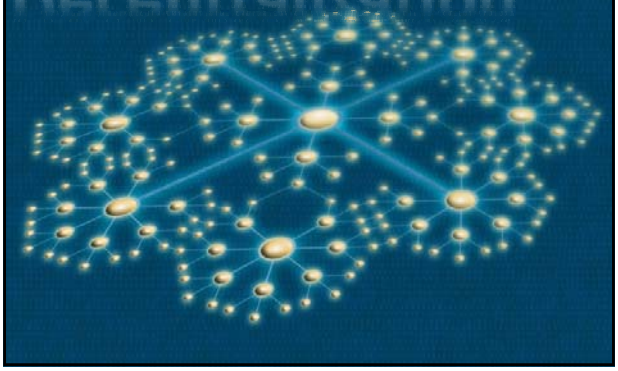
A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

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

Deferred Consistency

Batch jobs

Deferred Consistency

Batch jobs

Delayed replication

Deferred Consistency

Batch jobs

Delayed replication

Session death

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)

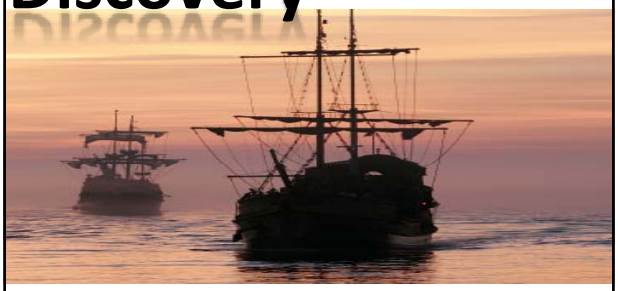
Indirection

Logical Names

ESB (conversion)

Proxies (monitoring)

Discovery



Discovery

Lookup based on Props

Discovery

Lookup based on Props

Load Balancing

… still keep things **reliable**

Contracts

Contracts

SLA

Contracts SLA
Orchestration

Contracts SLA
Orchestration
Interfaces (DBC)

Interoperability

Maintainability

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

Make Transparent

Platform Differences
Data Conversion

Make Transparent

Platform Differences
Data Conversion
Versioning
Make Transparent

Platform Differences
Data Conversion
Versioning
ESB Routing
Make Transparent

Developers need to be aware of the

distributed nature

of the overall system.

Make Explicit

Service Calls

Make Explicit

Service Calls
Potential Distribution

Make Explicit

Service Calls
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

4

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
Locking APIs

Protocols

Rely on user to
„do the right thing“

Protocols

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.

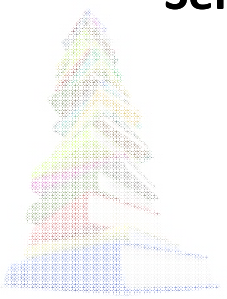
Standard

Library

Standard

Library

Semaphores

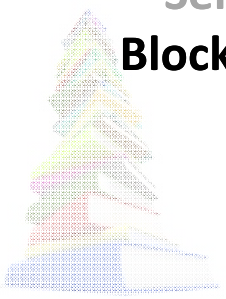


Standard

Library

Semaphores

Blocking Queues



Standard Library

Semaphores
Blocking Queues
Read/Write Lock

Standard Library

Semaphores
Blocking Queues
Read/Write Lock
Monitors

Bootstrapping

Bootstrapping

Blocking Queue
uses Monitors

Bootstrapping

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)

Tx Memory (what, not how)
Foreach vs. explicit loop

Tx Memory (what, not how)
Foreach vs. explicit loop
foreach seq(1..n)

Avoid Sideeffects

Avoid Sideeffects

Functional

Avoid Sideeffects

Functional

Stateless

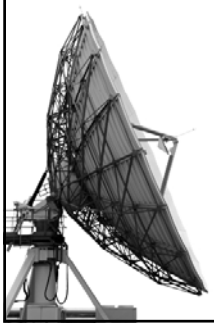
Avoid Sideeffects

Functional

Stateless

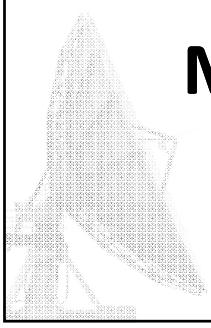
Immuntable

Tracking



Tracking

Monads



Automate

Automate

Tx Memory

≅

Garbage Collection

Case Study III: DSLs and Stuff

5

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

Domain Model

Abstraction

Domain Model
Domain Analysis

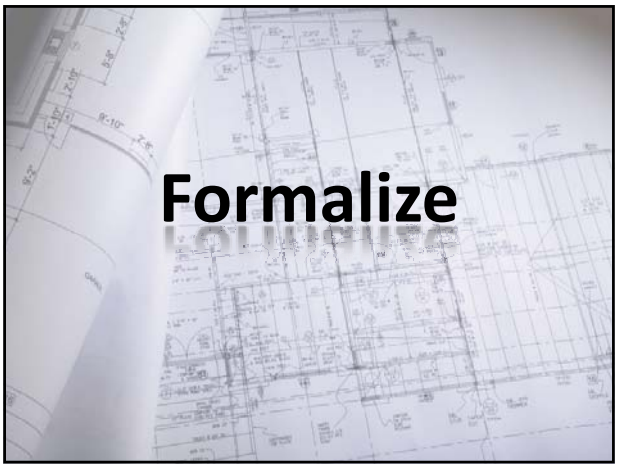
Abstraction

Domain Model
Domain Analysis
Domain Expert Interviews

Abstraction

Domain Model
Domain Analysis
Domain Expert Interviews
APIs

Abstraction

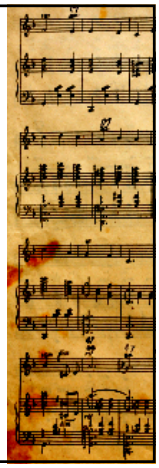


Metamodel

Metamodel

Abstract Syntax

Notation



Concrete Syntax

Graphical/Textual

Limit

Freedom

What vs. How

What vs. How
Generated Skeletons

What vs. How
Generated Skeletons
Architecture Enforcement

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

Viewpoints

Different DSLs
for different viewpoints

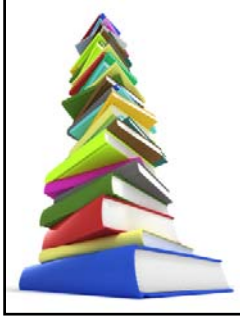
Platforms

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“

Constraints

Constraints
Generators

Constraints
Generators
Interpreters

Translate

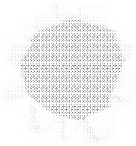
execute DSL
program by translating
to existing language

Interpret

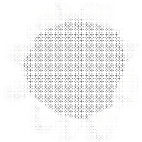
A (meta) program
That inspects a DSL
Program and executes
Side effects

Go Down

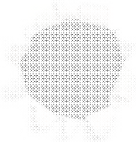
JAVA extensions



JAVA extensions
Protected Regions



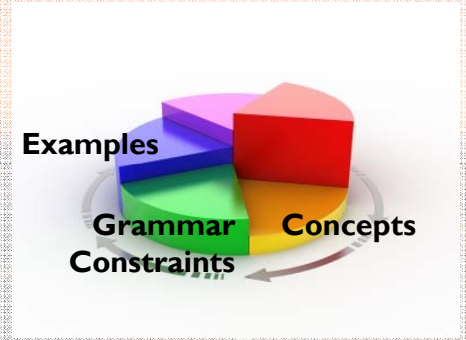
JAVA extensions
Protected Regions
Interpreter-Plugins

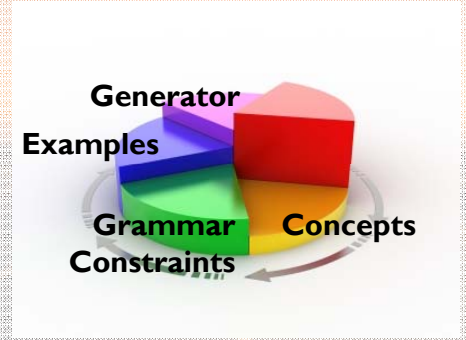



Iterate


Concepts

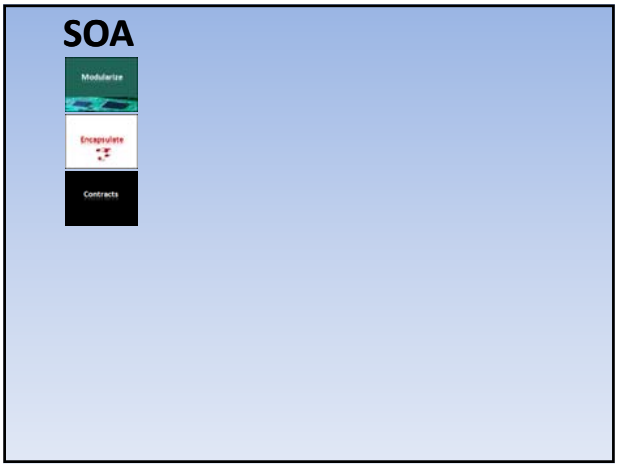

Grammar
Concepts
Constraints

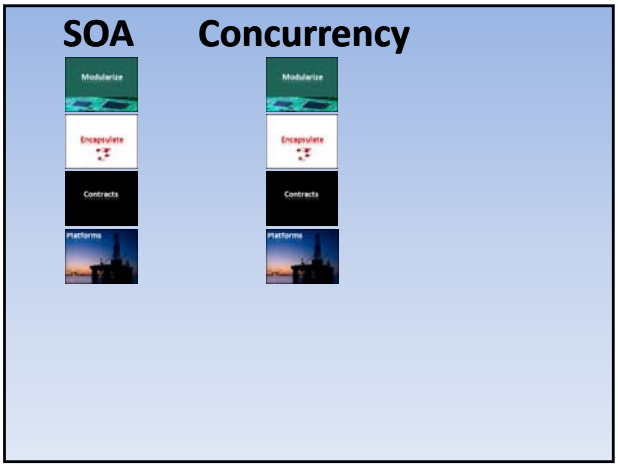

Examples
Grammar
Concepts
Constraints

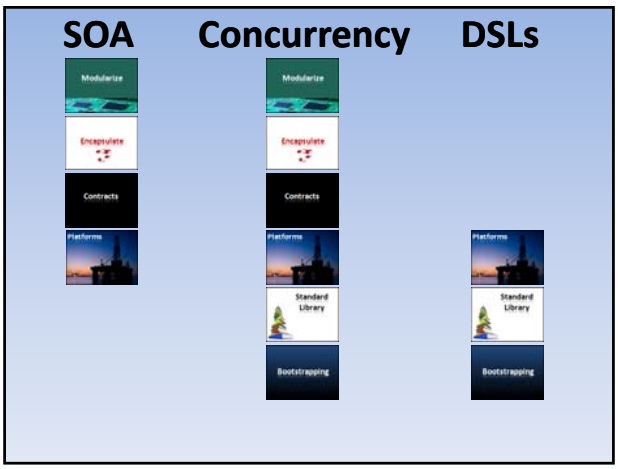

Generator
Examples
Grammar
Concepts
Constraints

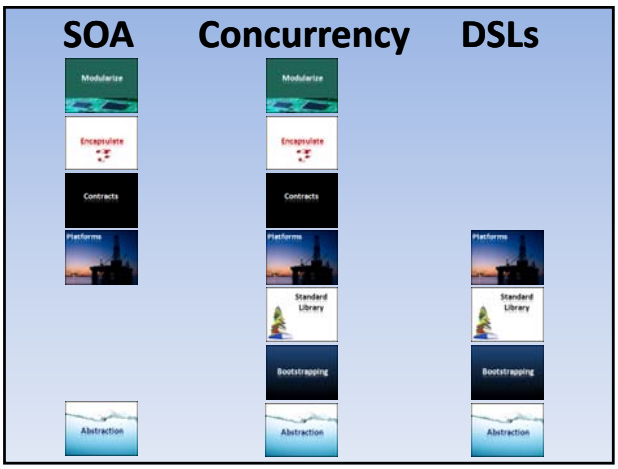

Generator
Editor Cust.
Examples
Grammar
Concepts
Constraints


6
Recap


SOA
Modularize
Encapsulate
Contracts


SOA
Concurrency
Modularize
Encapsulate
Contracts
Platforms
Modularize
Encapsulate
Contracts
Platforms


SOA
Concurrency
DSLs
Modularize
Encapsulate
Contracts
Platforms
Modularize
Encapsulate
Contracts
Platforms
Standard Library
Bootstrapping
Platforms
Standard Library
Bootstrapping


SOA
Concurrency
DSLs
Modularize
Encapsulate
Contracts
Platforms
Abstraction
Modularize
Encapsulate
Contracts
Platforms
Standard Library
Bootstrapping
Abstraction
Platforms
Standard Library
Bootstrapping
Abstraction

Goals

awareness

when building systems

Goals

checklists

for reviewing systems

Goals

education

of developers and architects

THE END.

Thank you.

Questions?