(Six) Developer
Best Practices
Looking beyond the hype

Markus Völter
(voelter@acm.org)

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

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:
Three Case Studies
SOA
Concurrency
DSLs and Stuff

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

01.12.2008
Staging

Simplicity

Decentralization

Open/Closed World

Distribution vs.
Local Caching

Compensation
Viewpoints

Notation

Go Meta

Reflection

Evolution/
Self Modification

Translate
Interpret

Automate

Make Transparent

Tracking

Protocols

Make Explicit

01.12.2008
Lazy vs. Eager

Limit
Freedom

Declaration
Implementation

Don’t
Overspecify

Avoid
Sideeffects

Iterate
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.
Distributed Responsibility

"Business Agility"

Decentralization

Service Ownership

Service Ownership
Technical Decisions

Service Ownership
Technical Decisions
Operation
Operate Independently
Evolve Independently
Shut Down Independently
Restart Independently
Decoupling Message Queues
Decoupling Message Queues No rigid Data Structs

Decoupling Message Queues No rigid Data Structs ESB (convert data)

Deferred Consistency

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

Batch jobs

Deferred Consistency Delayed replication
Encapsulate
Keep Impl Private

Indirection
Logical Names
ESB (conversion)
Proxies (monitoring)
Discovery

Lookup based on Props

Load Balancing

Contracts

... still keep things reliable

SLA
SLA Orchestration

Interfaces (DBC)

Interoperability

Maintainability

Platforms

ESBs
JEE
SCA
Developers should not have to care about different platforms

Isolate Technology

WSDL
SDO
SCA
MDSD

Make Transparent

Platform Differences

Data Conversion
Developers need to be aware of the **distributed nature** of the overall system.
Make Explicit
Service Calls
Potential Distribution
Ownership Transfer

Keep the system running smoothly...

Service Use (GC, Billing)

Measure

Service Use (GC, Billing)
Load (Scaling)

Measure

Service Use (GC, Billing)
Load (Scaling)
Performance
Case Study II: Concurrency

IMPORTANT!

Traditional Approach

Modularize

More manageable concurrent Units „Smaller Problems“
Encapsulate

Encapsulate inside module – don’t expose! „Thread-safe interface“

Platforms

Servlets

EJB

COM+ (Apartments)
... and finally

API

Contracts

Thread safety

Locking APIs

Protocols

Rely on user to „do the right thing“
Rely on user to „do the right thing“

Protocols
Acquire/Release

Build more powerful abstractions from basics.

Standard Library

Semaphores

Blocking Queues
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

Don’t
Overspecify

Tx Memory
(I want this atomic, don’t care about locks)

Tx Memory
(what, not how)

Foreach vs. explicit loop

Tx Memory
(what, not how)

Foreach vs. explicit loop
foreach seq(1..n)
Avoid Sideeffects

Functional
Stateless
Immutatable

Tracking

Monads
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
Domain Analysis
Abstraction

Domain Model
Domain Analysis
Domain Expert Interviews

Domain Model
Domain Analysis
Domain Expert Interviews
APIs
Formalize

Metamodel

Abstract Syntax

Notation

Concrete Syntax

Graphical/Textual

Limit

Freedom
What vs. How
Generated Skeletons
Limit Freedom
What vs. How
Generated Skeletons
Arc 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

Language that can define abstractions itself „grow the language“
Go Meta

Constraints

Generators

Interpreters

Translate program by translating to existing language
A (meta) program that inspects a DSL program and executes side effects.

Go Down

JAVA extensions

Protected Regions

JAVA extensions

Interpreter-Plugins
Recap

SOA

Concurrency

DSLs
awareness when building systems

Goals

checklists for reviewing systems

Goals

education of developers and architects

Goals

THE END.
Thank you.

Questions?