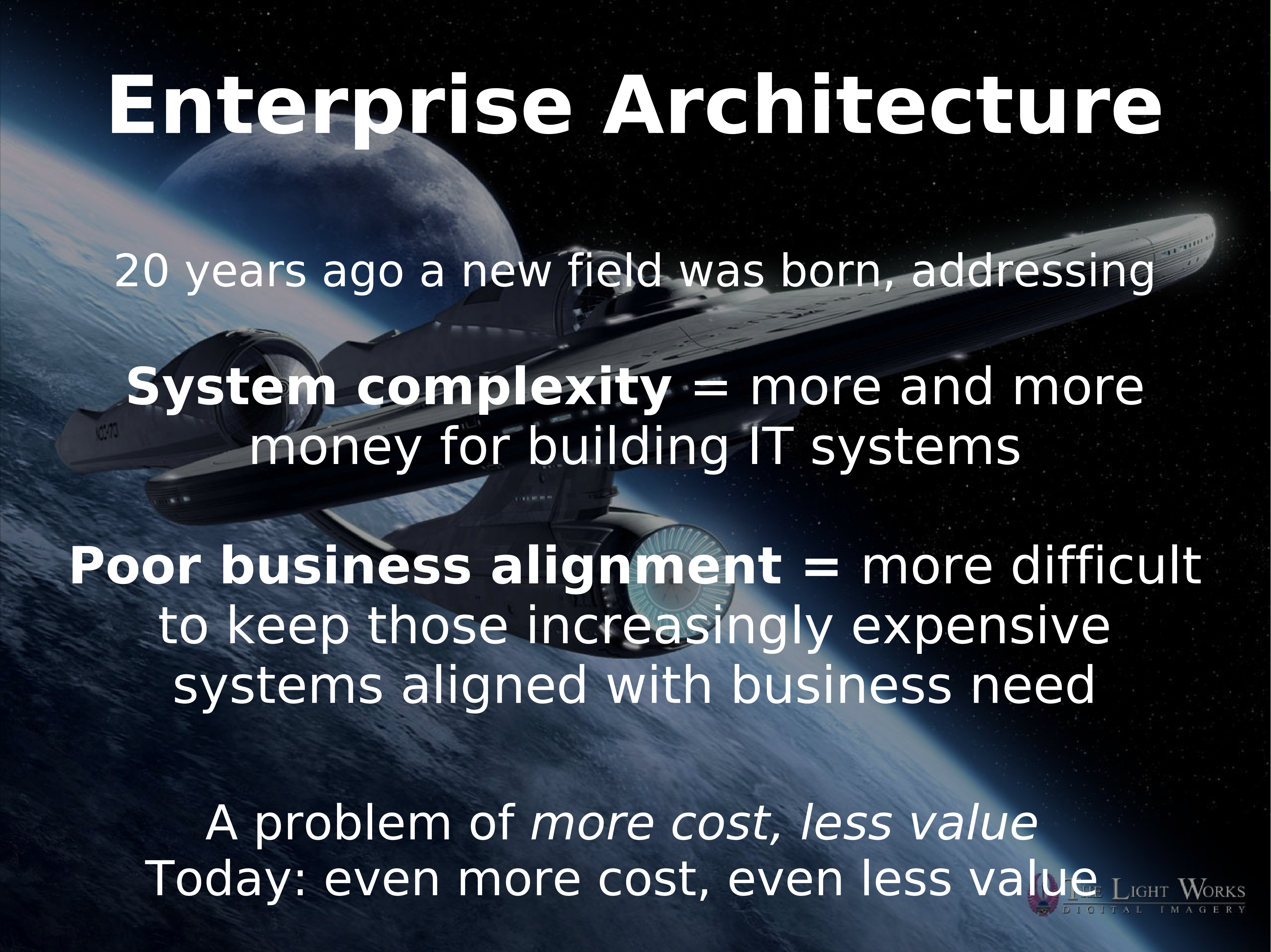
Simple Pure Java

codeborne
well-crafted software

Anton Keks
anton@codeborne.com

The Enterprise…



Enterprise Architecture

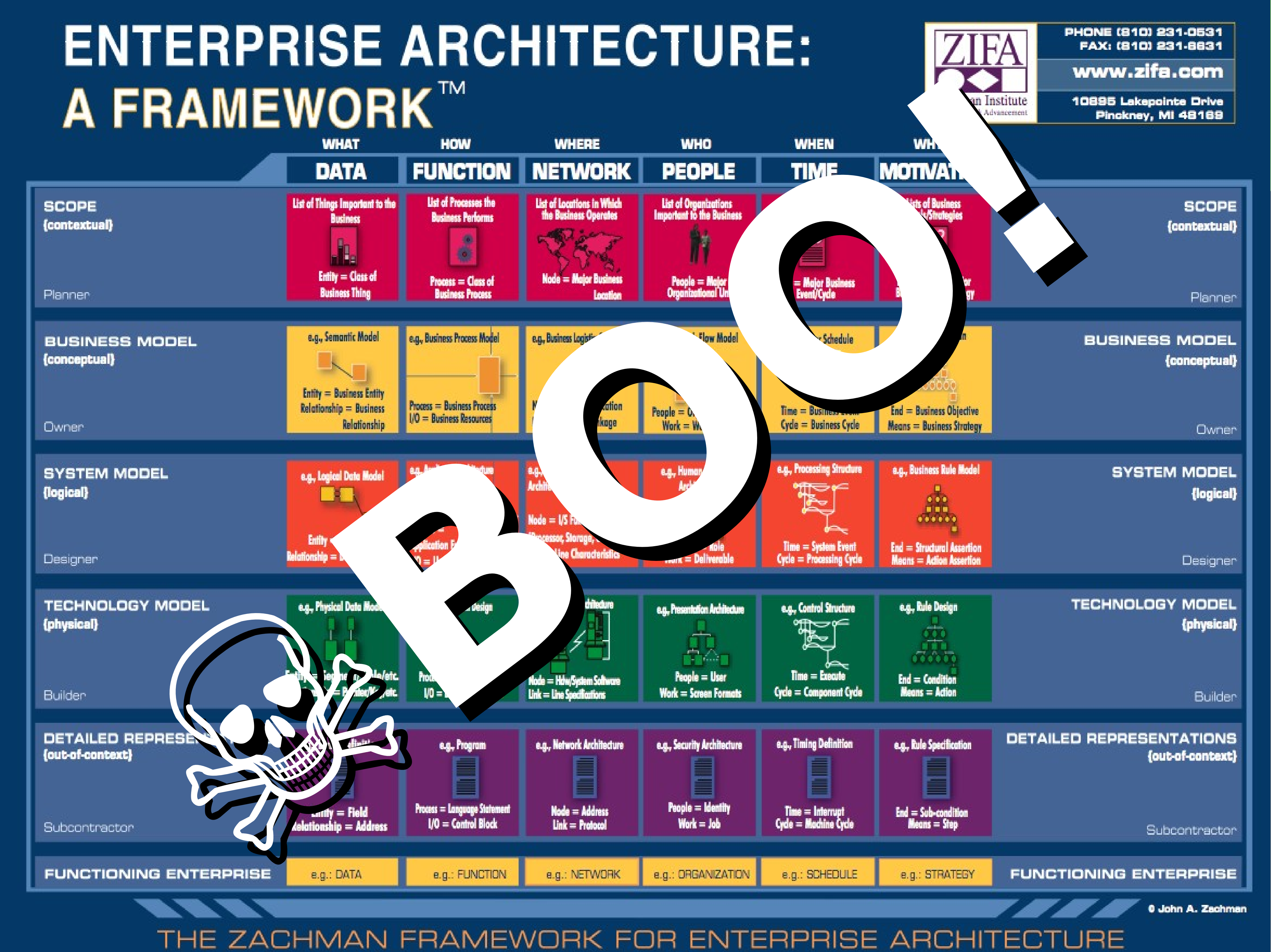
20 years ago a new field was born, addressing

System complexity = more and more money for building IT systems

Poor business alignment = more difficult to keep those increasingly expensive systems aligned with business need

A problem of *more cost, less value*
Today: even more cost, even less value

Complexity



ENTERPRISE ARCHITECTURE: A FRAMEWORK™

ZIFA
...n Institute
...Advancement

PHONE (810) 231-0531
FAX: (810) 231-6631
www.zifa.com
10895 Lakepointe Drive
Pinckney, MI 48169

	WHAT DATA	HOW FUNCTION	WHERE NETWORK	WHO PEOPLE	WHEN TIME	WH... MOTIVAT...	
SCOPE (contextual) Planner	List of Things Important to the Business. Entity = Class of Business Thing	List of Processes the Business Performs. Process = Class of Business Process	List of Locations in Which the Business Operates. Node = Major Business Location	List of Organizations Important to the Business. People = Major Organizational Un...	... = Major Business Event/Cycle	Lists of Business ...ls/Strategies ... B... ...or ...gy	SCOPE (contextual) Planner
BUSINESS MODEL (conceptual) Owner	e.g., Semantic Model. Entity = Business Entity Relationship = Business Relationship	e.g., Business Process Model. Process = Business Process I/O = Business Resources	e.g., Business Logi... N... ...ation ...kage	...Flow Model. People = O... Work = W...	...Schedule. Time = Business ... Cycle = Business Cycle	...n. End = Business Objective Means = Business Strategy	BUSINESS MODEL (conceptual) Owner
SYSTEM MODEL (logical) Designer	e.g., Logical Data Model. Entity ... Relationship = ...	e.g., A... ...ture. ...plication F... I/O = ...	e.g., ... Archi... Node = I/S F... (Processor, Storage, ...line Characteristics	e.g., Human ... Arch... ...Role ...Work = Deliverable	e.g., Processing Structure. Time = System Event Cycle = Processing Cycle	e.g., Business Rule Model. End = Structural Assertion Means = Action Assertion	SYSTEM MODEL (logical) Designer
TECHNOLOGY MODEL (physical) Builder	e.g., Physical Data Mod... Entity ...gment/Table/etc. ...Pointer/Key/etc.	...Design. Proc... I/O = ...	...hitecture. Node = Hw/System Software Link = Line Specifications	e.g., Presentation Architecture. People = User Work = Screen Formats	e.g., Control Structure. Time = Execute Cycle = Component Cycle	e.g., Rule Design. End = Condition Means = Action	TECHNOLOGY MODEL (physical) Builder
DETAILED REPRESE... (out-of-context) Subcontractor	...efinition. ...ntity = Field ...elationship = Address	e.g., Program. Process = Language Statement I/O = Control Block	e.g., Network Architecture. Node = Address Link = Protocol	e.g., Security Architecture. People = Identity Work = Job	e.g., Timing Definition. Time = Interrupt Cycle = Machine Cycle	e.g., Rule Specification. End = Sub-condition Means = Step	DETAILED REPRESENTATIONS (out-of-context) Subcontractor
FUNCTIONING ENTERPRISE	e.g.: DATA	e.g.: FUNCTION	e.g.: NETWORK	e.g.: ORGANIZATION	e.g.: SCHEDULE	e.g.: STRATEGY	FUNCTIONING ENTERPRISE

© John A. Zachman

THE ZACHMAN FRAMEWORK FOR ENTERPRISE ARCHITECTURE

Your average Java (web) app

- Framework(s)
- Model
- Portal?
- Services!
- Remote services, SOA, EJB
- JNDI
- Stubs, generated code
- How many layers?

...

- Patterns!
- Factory, Singleton, Facade
- Enterprise patterns!
- DTO, DAO, etc
- Getters/setters
  (what's the deal - generate 'em)
- JPA, JAXB
- JMS!

Bad words

Layer Tier Bus
Context Manager
Locator Assembler Bean
Broker Facade
Transfer Object DAO

...

It's always nice to see guys like

SessionServiceContextManager

or **AbstractStrategyFactoryProxyFacadeBuilder**

EAR, WAR, WAR, WAR

Configuration, descriptors!

JNDI!

ESB? ;-)

Must be a wizard to make it all run!

Improves job security index :-)

For a small thing we create

- ManagedBean
- MegaServiceWrapper
- CoreServiceLocal, Stub (JNDI lookup?)
- CoreService (&& CoreServiceImpl)
- AggregatedService
- ConcreteService
- ConcreteDAO
- JPAAdapter
- ...

The result?

- It kills productivity
- Thousands lines of code, very few functionality
- Well hidden ”business logic”
- N minute deploy time (+ N minutes app server startup)
- Oops, sometimes redeploy doesn't work, need to restart
- Slow UI responsivness

What to do?

- Concentrate on domain terminology
  - This was the intention of OOP
- Avoid overly defensive code
  - You are not writing a framework!
  - Fellow developers are friends
- Follow **Clean Code** by Robert C. Martin

Java platform and language

- Java (EE) is a victim of JCP
    - Many unreal/unusable JSRs
- Stick to proven open-source stuff
    - Less standards – the better
- Java language is ok
    - The biggest miss are closures
    - DSLs are possible: Mockito, LambdaJ
    - Don't bloat (generate) your code

Code style

- Is your code style limiting readability?
  - Avoid too many line breaks and braces
  - Emphasize what is important

```
public int size() {
  if (root == null) {
    return 0;
  }
  else {
    return root.numSiblings();
  }
}
```

```
public int size() {
  if (root == null) return 0;
  return root.numSiblings();
}
```

```
public int size() {
  return root != null ?
    root.numSiblings() : 0;
}
```

Code style

- Avoid over-indenation (or code ladder)

```
public void startCharging() {
  if (customer.hasFunds()) {
    if (!station.isCharging()) {
      if (!station.currentlyBooked()) {
        reallyStartCharging();
        return;
      }
    }
  }
  throw new UnableToStartException();
}
```

```
public void startCharging() {
  if (!customer.hasFunds()) throw new UnableToSt
  if (station.isCharging()) throw new UnableToSt
  if (station.currentlyBooked()) throw new Unabl

  reallyStartCharging();
}
```

Code style

- Prefer shorter code (use static imports)

```
List<Integer> list = Arrays.asList(1, 2, 3));
list = Collections.unmodifieableList(list);
return list;

import static java.util.Arrays.*;
import static java.util.Collections.*;
...

return unmodifiableList(asList(1, 2, 3))
```

Looks a bit like functional programming, isn't it?

Code style

- Prefer good naming to comments
- Avoid getters/setters, equals, hashCode, toString unless necessary
- Break free from 'conventions'
- Work towards a DSL

```
when(session.currentUser()).thenReturn(fakeUser);

assertThat(person.age, is(25));

sort(people, on(Person.class).getAge());
```

Proper Java app

- Jetty Launcher (esp in development)
- Know the APIs well: servlets, filters, etc
- Avoid vendor-specific stuff
- Keep environment-specific configuration in version control
- Dependency Injection
- Avoid scattering cross-cutting concerns
- DB migrations (w/ liquibase/dbdeploy)
- Start thin and simple, prefer higher SNR

Web UI

- Your framework tells you don't need to know JavaScript? (GWT, JSF, etc)
- B.S.!
- Keep it under control: learn basics of JQuery instead
- Knockout.js, Backbone.js can help
- You are not limited with Java syntax on the client side :-)

Worth reminding...

- **D**on't **R**epeat **Y**ourself
- **K**eep **I**t **S**imple **S**tupid
- **Y**ou **A**in't **G**onna **N**eed **I**t
- **T**est **D**riven **D**evelopment

Let's continue on github:

github.com/angryziber/simple-java

(or just google: ”gotocon simple java”)

job@codeborne.com