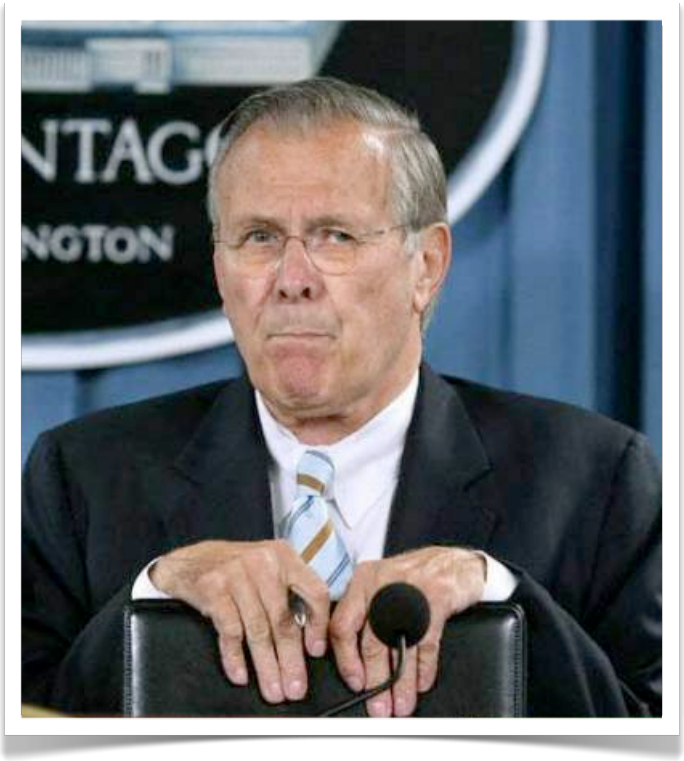

DEVNATION
April 13-17, 2014
San Francisco, California
Agile Architecture & Design
Neal Ford
Director / Software Architect /
Meme Wrangler
ThoughtWorks®
2002 Summit Blvd, Level 3, Atlanta, GA 30319, USA
T: +1 40 4242 9929 Twitter: @neal4d
E: nford@thoughtworks.com W: thoughtworks.com



"There are known unknowns.

That is to say there are things that we now know we don't know.

But there are also unknown unknowns.

There are things we do not know we don't know."

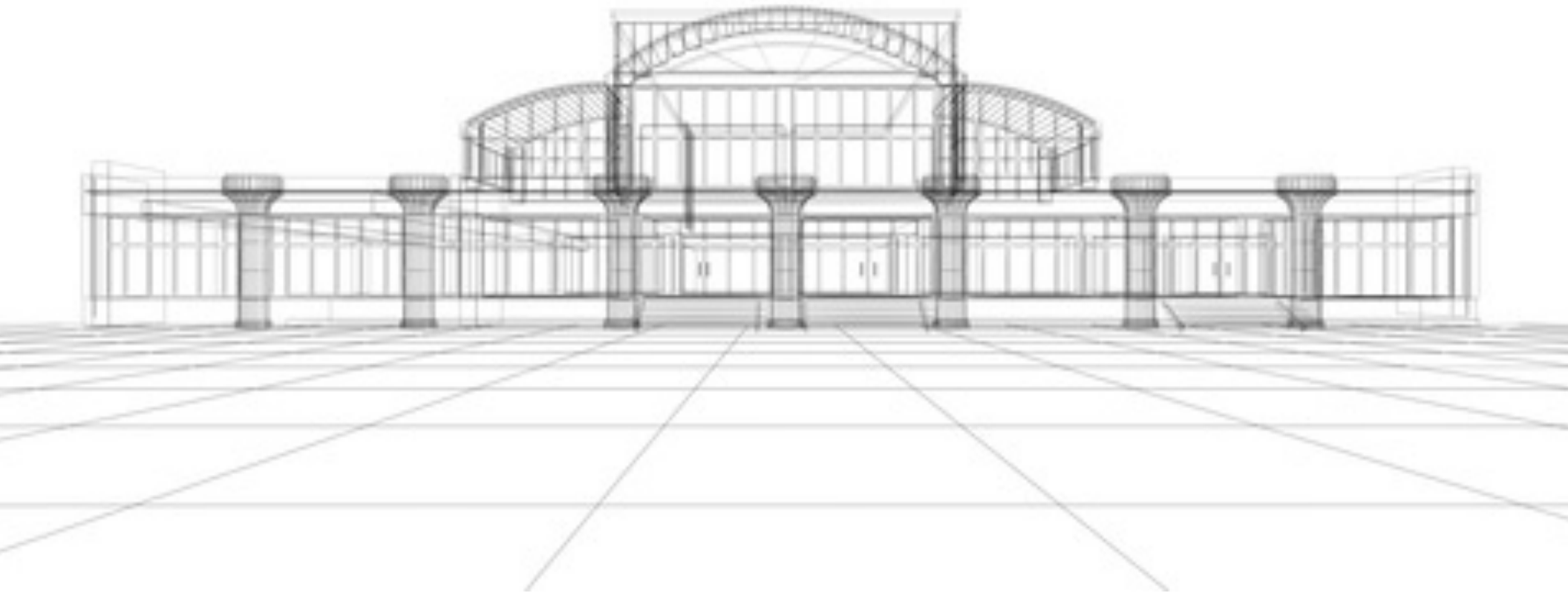
There are things we do not know we don't know."

unknown
unknowns

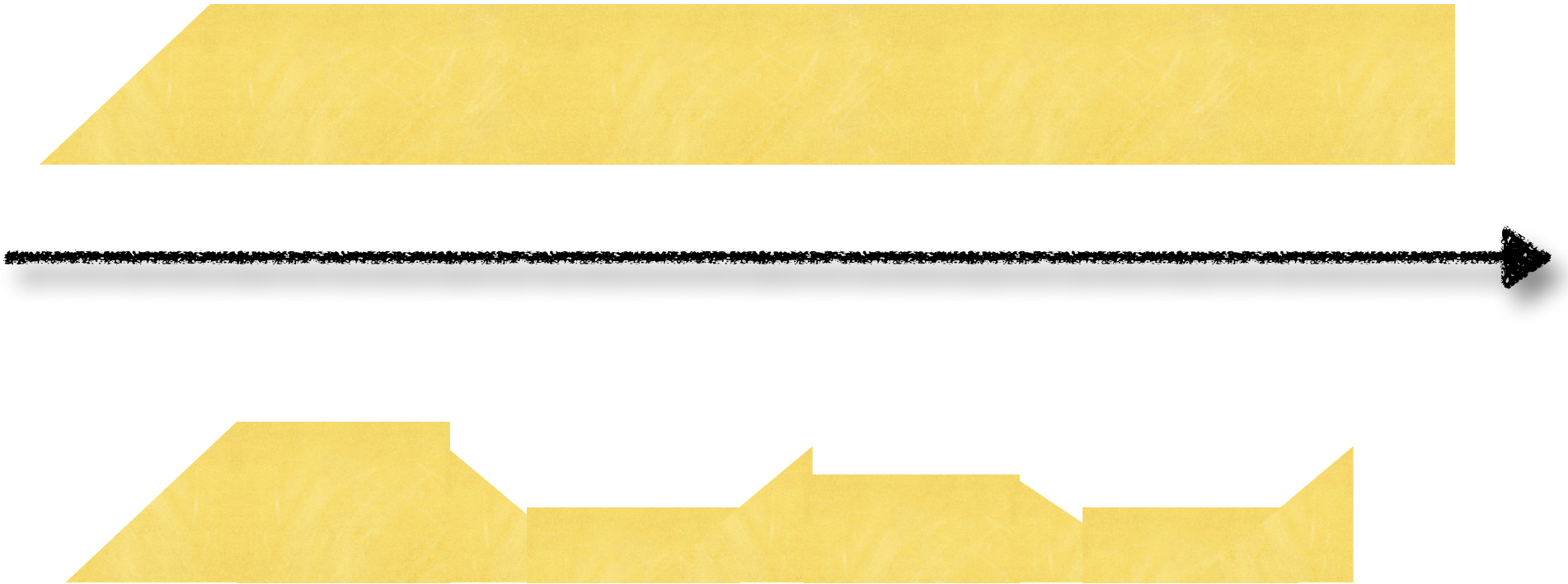
focus on proven practices

the future is hard to predict!

traditional vs agile architecture



traditional role



agile role

consequences of < time up front

good feel for current events

hands-on

ability to switch from micro <=> macro often & easily

fewer boat anchors


DEVNATION
April 13-17, 2014
San Francisco, California
Agile Architecture &
Design
Neal Ford
Director / Software Architect /
Meme Wrangler
ThoughtWorks®
2002 Summit Blvd, Level 3, Atlanta, GA 30319, USA
T: +1 40 4242 9929 Twitter: @neal4d
E: nford@thoughtworks.com W: thoughtworks.com

Rising or emerging out of anything that covers or conceals; issuing; coming to light.
[1913 Webster]

Emergent Design

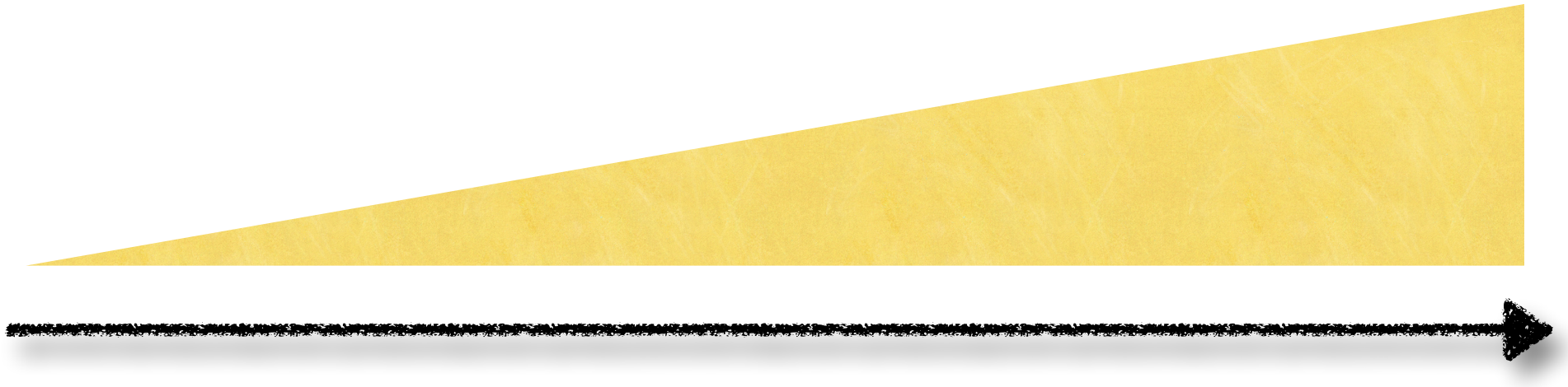
Suddenly appearing; arising unexpectedly; calling for prompt action; urgent.
[1913 Webster]


finding
abstractions &
patterns

finding abstractions & patterns

last responsible moment

last responsible moment



the longer the delay => more relevant data for decision

finding
abstractions &
patterns

last responsible moment

emergent design

```
public void addOrder(final ShoppingCart cart, String userName,
                     Order order) throws SQLException {
    Connection c = null; PreparedStatement ps = null;
    Statement s = null; ResultSet rs = null;
    boolean transactionState = false;
    try {
        c = dbPool.getConnection();
        s = c.createStatement();
        transactionState = c.getAutoCommit();
        int userKey = getUserKey(userName, c, ps, rs);
        c.setAutoCommit(false);
        addSingleOrder(order, c, ps, userKey);
        int orderKey = getOrderKey(s, rs);
        addLineItems(cart, c, orderKey);
        c.commit();
        order.setOrderKey(orderKey);
    } catch (SQLException sqlx) {
        s = c.createStatement();
        c.rollback();
        throw sqlx;
    } finally {
        try {
            c.setAutoCommit(transactionState);
            dbPool.release(c);
            if (s != null)  s.close();
            if (ps != null) ps.close();
            if (rs != null) rs.close();
        } catch (SQLException ignored) {
        }
    }
}
```

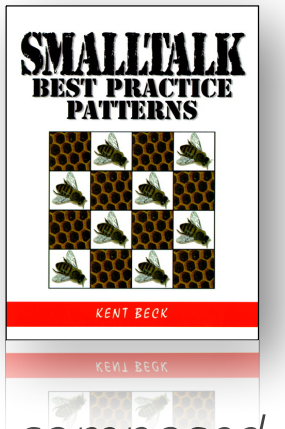
finding & harvesting idiomatic patterns

finding & harvesting idiomatic patterns

```
public void addOrder(final ShoppingCart cart, String userName,
                     Order order) throws SQLException {
    Connection connection = null; PreparedStatement ps = null;
    Statement statement = null; ResultSet rs = null;
    boolean transactionState = false;
    try {
        connection = dbPool.getConnection();
        statement = connection.createStatement();
        transactionState = setupTransactionStateFor(connection, transactionState);
        addSingleOrder(order, connection, ps, userKeyFor(userName, connection));
        order.setOrderKey(generateOrderKey(statement, rs));
        addLineItems(cart, connection, order.getOrderKey());
        completeTransaction(connection);
    } catch (SQLException sqlx) {
        rollbackTransactionFor(connection);
        throw sqlx;
    } finally {
        cleanUpDatabaseResources(connection, transactionState, statement, ps, rs);
    }
}
```

```
public void addOrderFrom(ShoppingCart cart, String userName,
                         Order order) throws SQLException {
    setupDataInfrastructure();
    try {
        add(order, userKeyBasedOn(userName));
        addLineItemsFrom(cart, order.getOrderKey());
        completeTransaction();
    } catch (SQLException sqlx) {
        rollbackTransaction();
        throw sqlx;
    } finally {
        cleanUp();
    }
}
```

finding & harvesting idiomatic patterns



see the *composed method* pattern

Smalltalk Best Practice Patterns Kent Beck

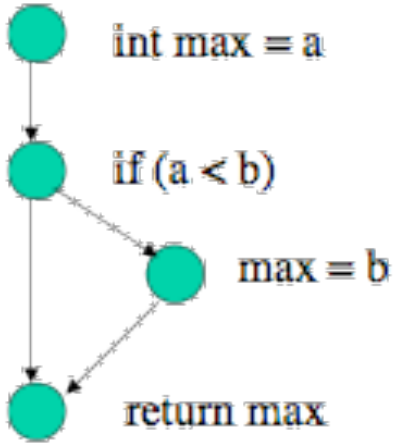
harvesting idiomatic patterns

cyclomatic complexity

measures the complexity of a method/function

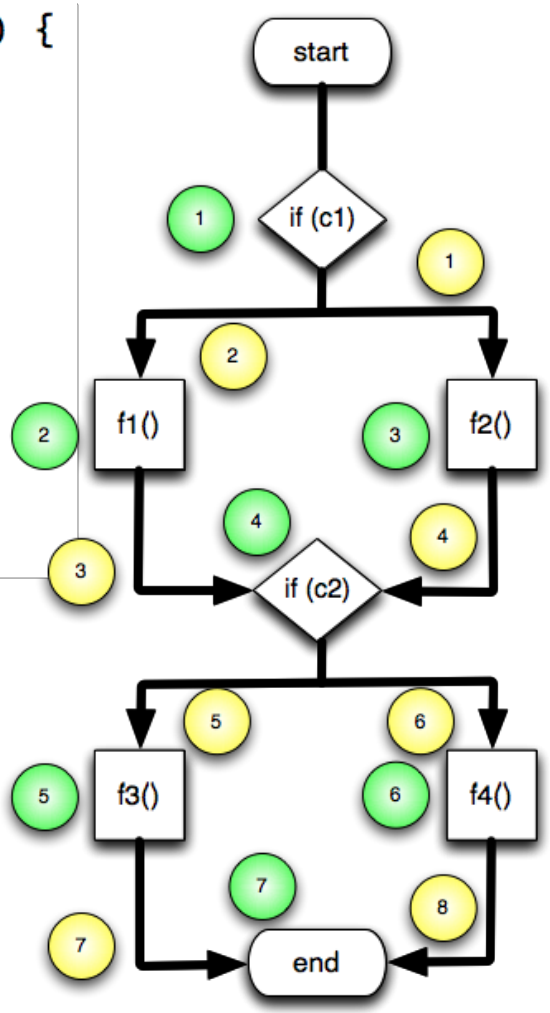
V(G)= e - n + 2
V(G) = cyclomatic complexity of G
e= # edges
n= # of nodes

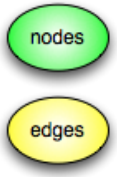
```
int max (int a, int b) {
    int max = a;
    if (a < b) {
        max = b;
    }
    return max;
}
```



cyclomatic complexity

```
public void doIt() {
    if (c1) {
        f1();
    } else {
        f2();
    }
    if (c2) {
        f3();
    } else {
        f4();
    }
}
```

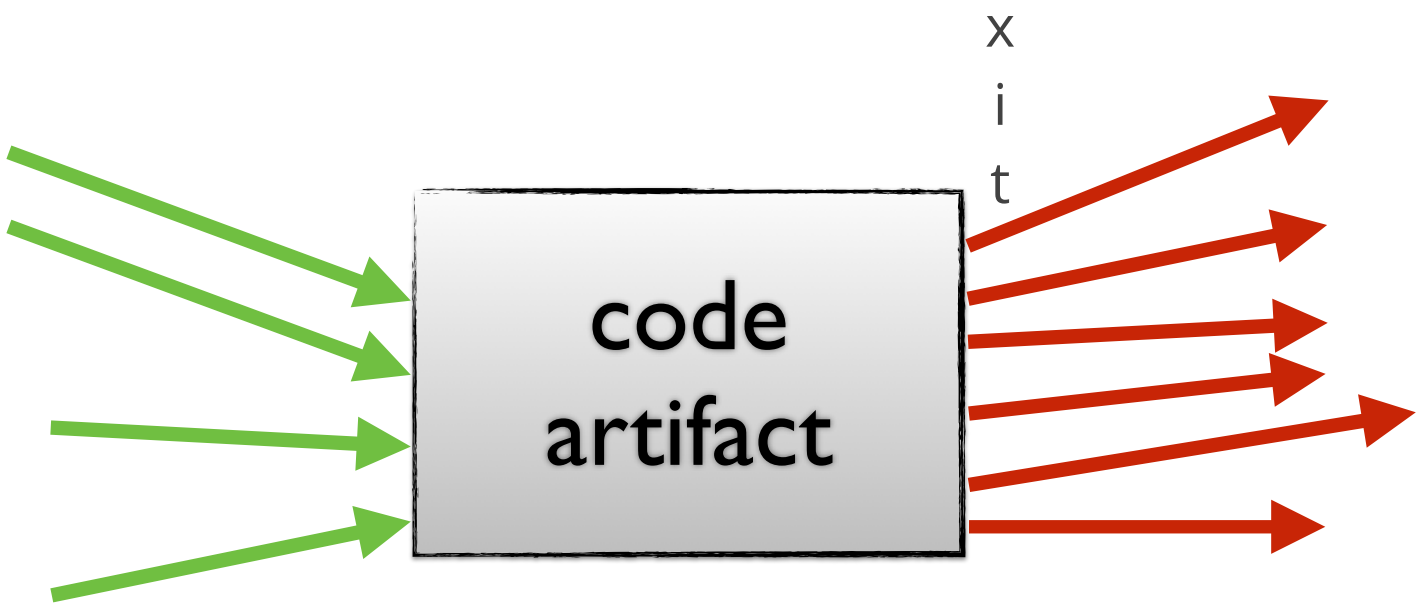




coupling

afferent

efferent



classname	WMC	Ca
org.apache.struts2.components.Component	28	177
org.apache.struts2.views.freemarker.tags.TagModel	7	47
org.apache.struts2.views.velocity.components.AbstractDirective	8	43
org.apache.struts2.StrutsException	7	23
org.apache.struts2.components.UIBean	53	22
org.apache.struts2.dispatcher.mapper.ActionMapping	13	20
org.apache.struts2.views.jsp.ComponentTagSupport	6	19
org.apache.struts2.dispatcher.Dispatcher	37	19
org.apache.struts2.views.jsp.ui.AbstractUITag	34	18
org.apache.struts2.views.xslt.AdapterFactory	9	16
org.apache.struts2.views.xslt.AdapterNode	10	15
org.apache.struts2.ServletActionContext	11	15
org.apache.struts2.components.table.WebTable	33	12
org.apache.struts2.dispatcher.mapper.ActionMapper	2	11
org.apache.struts2.components.template.TemplateEngine	2	10
org.apache.struts2.components.template.Template	7	10
org.apache.struts2.dispatcher.StrutsResultSupport	13	10
org.apache.struts2.components.Form	24	10
org.apache.struts2.components.ListUIBean	8	9
org.apache.struts2.util.MakeIterator	3	8
org.apache.struts2.StrutsStatics	0	7

```
public void evaluateParams() {
    String templateDir = getTemplateDir();
    String theme = getTheme();

    addParameter("templateDir", templateDir);
    addParameter("theme", theme);
    addParameter("template", template != null ? findString(template) : getDefaultTemplate());
    addParameter("dynamicAttributes", dynamicAttributes);
    addParameter("themeExpansionToken", uiThemeExpansionToken);
    addParameter("expandTheme", uiThemeExpansionToken + theme);

    String name = null;
    String providedLabel = null;

    if (this.key != null) {

        if(this.name == null) {
            this.name = key;
        }

        if(this.label == null) {
            // lookup the label from a TextProvider (default value is the key)
            providedLabel = TextProviderHelper.getText(key, key, stack);
        }

    }

    if (this.name != null) {
        name = findString(this.name);
        addParameter("name", name);
    }

    if (label != null) {
        addParameter("label", findString(label));
    } else {
        if (providedLabel != null) {
            // label found via a TextProvider
            addParameter("label", providedLabel);
        }
    }

    if (labelSeparator != null) {
        addParameter("labelseparator", findString(labelSeparator));
    }

    if (labelPosition != null) {
        addParameter("labelposition", findString(labelPosition));
    }

    if (requiredPosition != null) {
```

evaluate.*Params?

find . -name "*.java" | xargs grep -l "void evaluate.*Params"

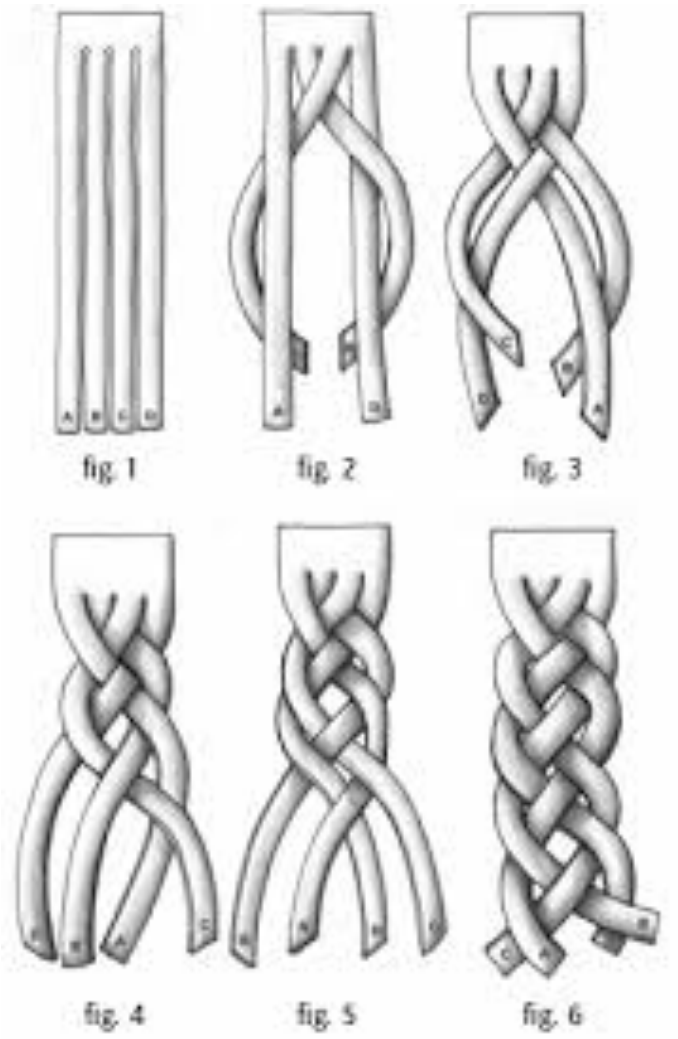
./org/apache/struts2/components/AbstractRemoteCallUIBean.java
./org/apache/struts2/components/Anchor.java
./org/apache/struts2/components/Autocompleter.java
./org/apache/struts2/components/Checkbox.java
./org/apache/struts2/components/ComboBox.java
./org/apache/struts2/components/DateTimePicker.java
./org/apache/struts2/components/Div.java
./org/apache/struts2/components/DoubleListUIBean.java
./org/apache/struts2/components/DoubleSelect.java
./org/apache/struts2/components/File.java
./org/apache/struts2/components/Form.java
./org/apache/struts2/components/FormButton.java
./org/apache/struts2/components/Head.java
./org/apache/struts2/components/InputTransferSelect.java
./org/apache/struts2/components/Label.java
./org/apache/struts2/components/ListUIBean.java
./org/apache/struts2/components/OptionTransferSelect.java
./org/apache/struts2/components/Password.java
./org/apache/struts2/components/Reset.java
./org/apache/struts2/components/Select.java
./org/apache/struts2/components/Submit.java
./org/apache/struts2/components/TabbedPanel.java
./org/apache/struts2/components/table/WebTable.java
./org/apache/struts2/components/TextArea.java
./org/apache/struts2/components/TextField.java
./org/apache/struts2/components/Token.java
./org/apache/struts2/components/Tree.java
./org/apache/struts2/components/UIBean.java
./org/apache/struts2/components/UpDownSelect.java

harvest

build frameworks

agile architecture

yesterday's best practice is tomorrow's anti-pattern

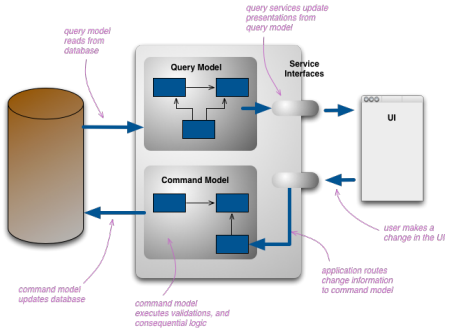


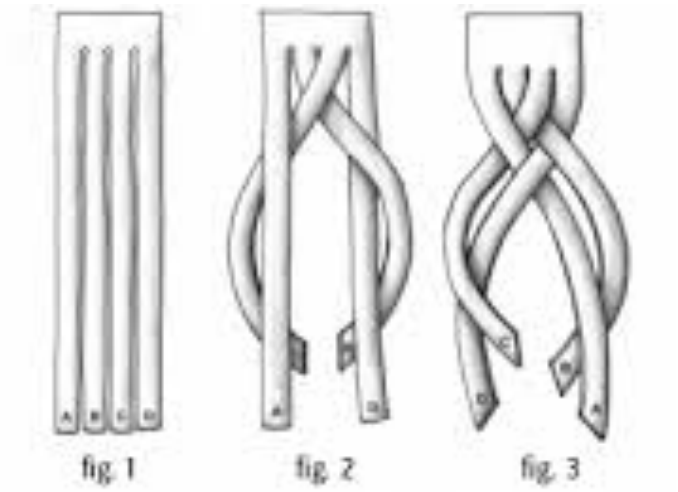
complect

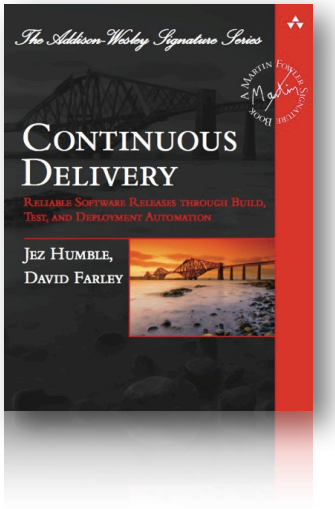
to interweave or entwine

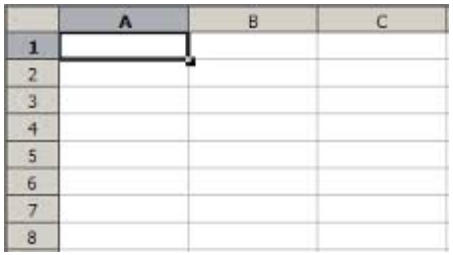
from Latin *complectī*

http://www.infoq.com/presentations/Simple-Made-Easy

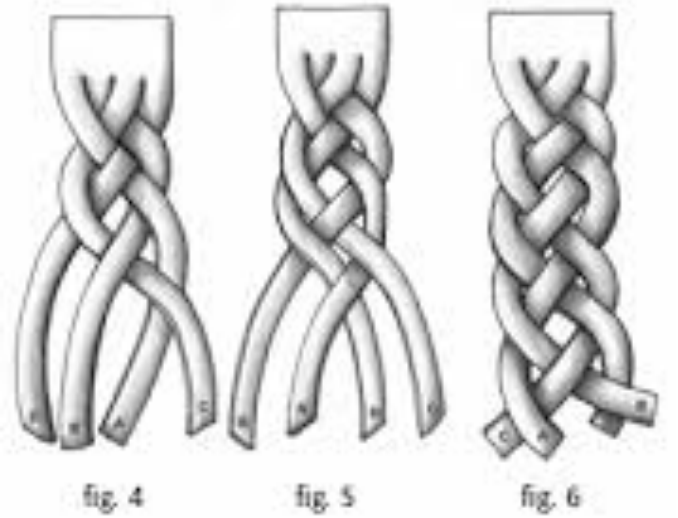


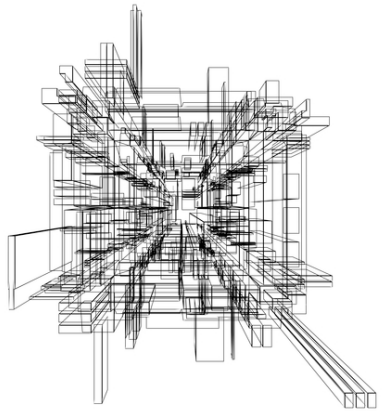




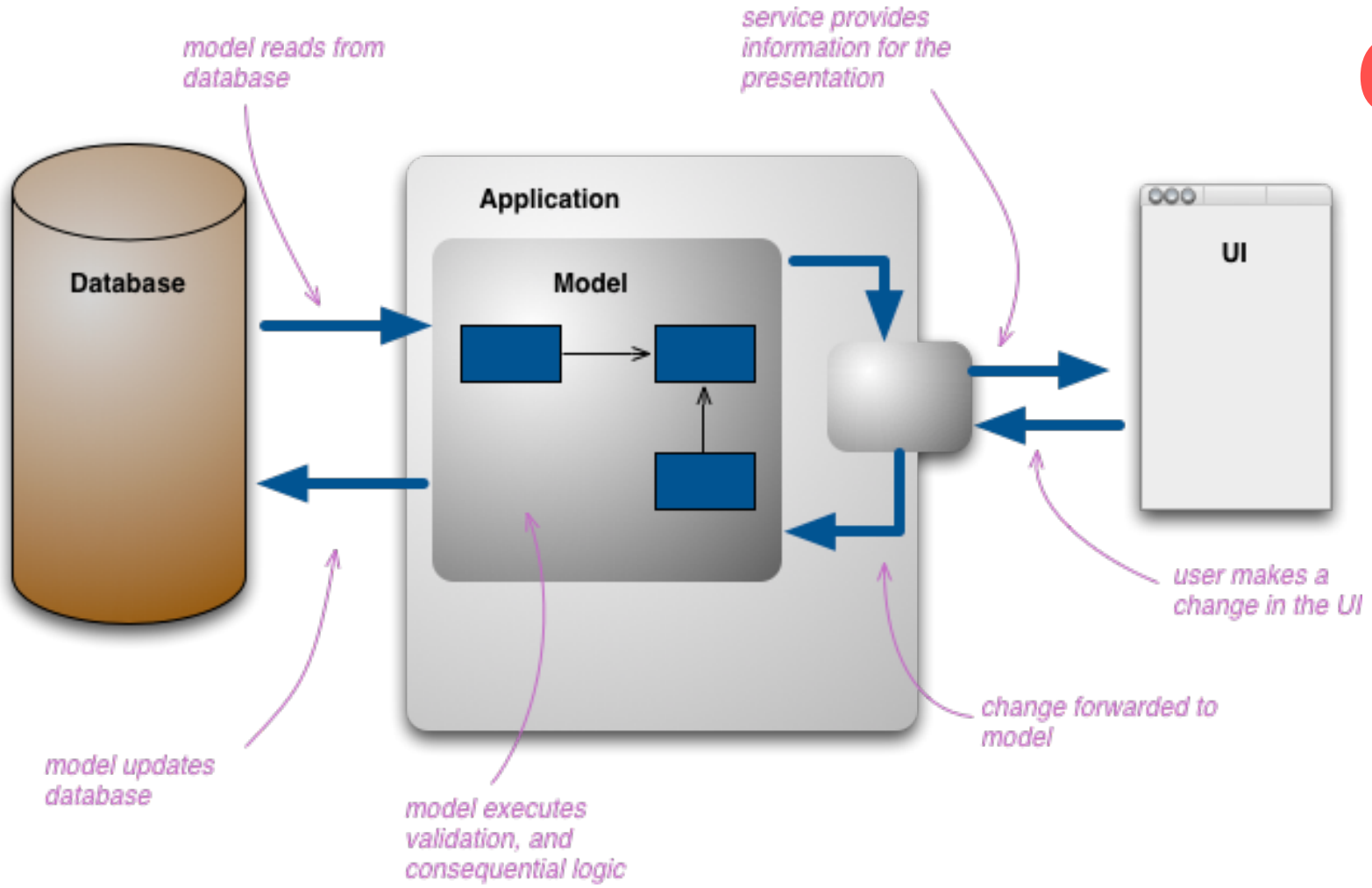


	A	B	C
1			
2			
3			
4			
5			
6			
7			
8			





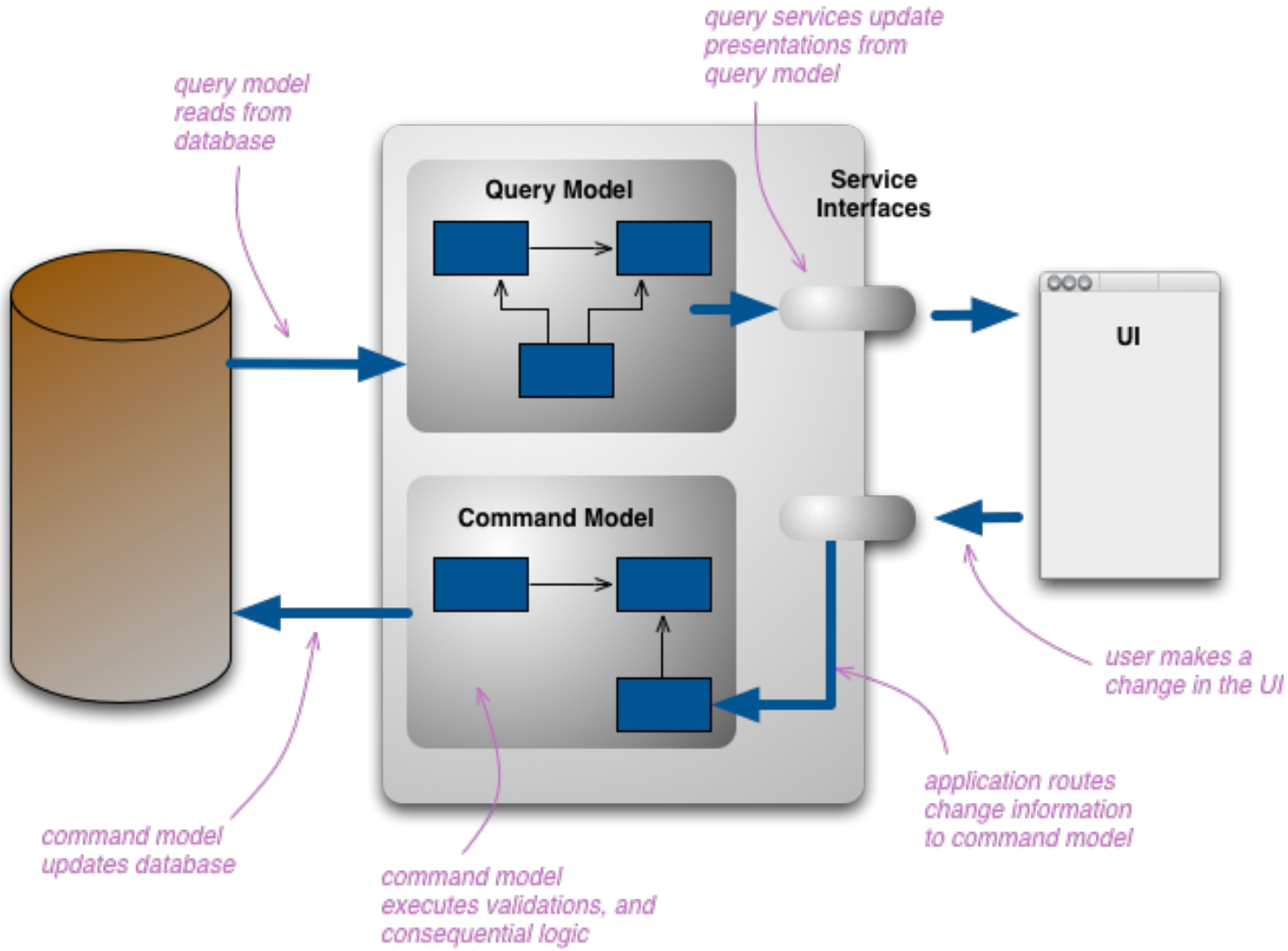
CRUD



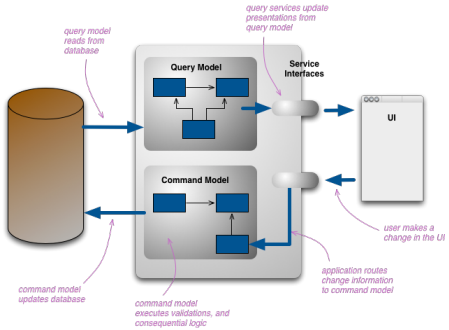
images by Martin Fowler: martinfowler.com/bliki/CQRS.html

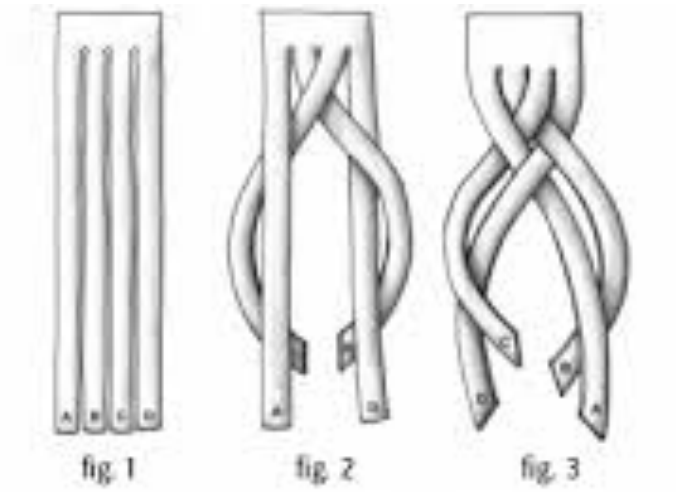
CQRS

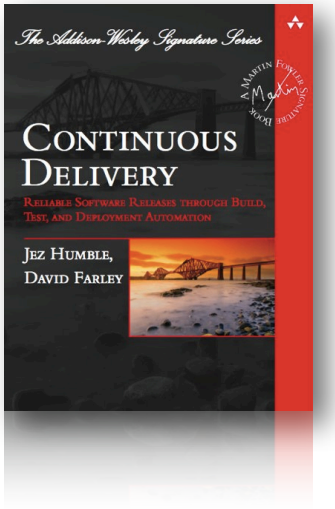
Command-Query Responsibility Segregation

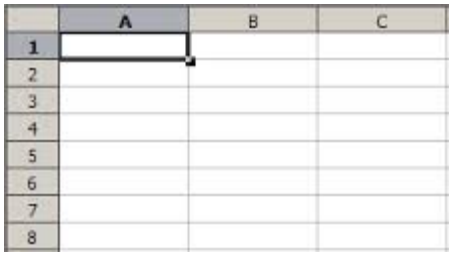


http://codebetter.com/gregyoung/2010/02/16/cqrs-task-based-uis-event-sourcing-agh/

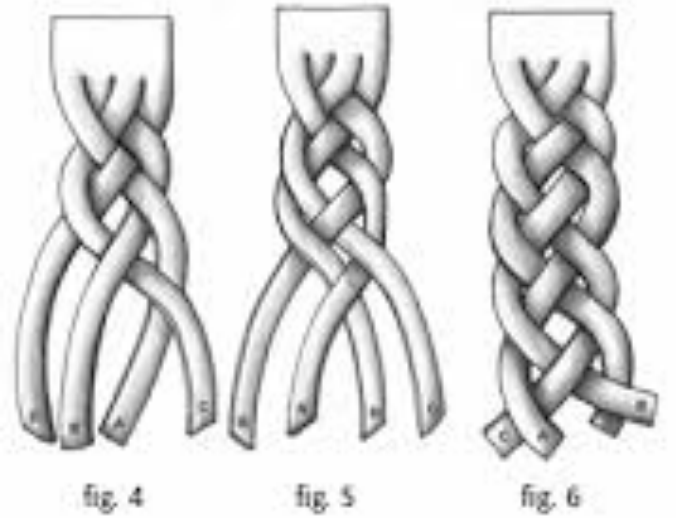


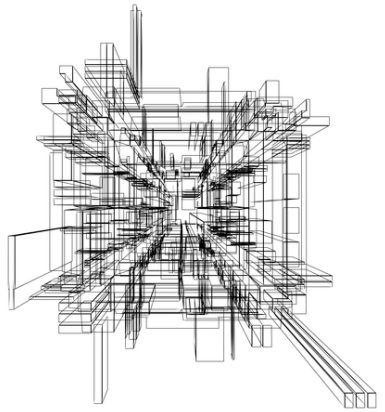




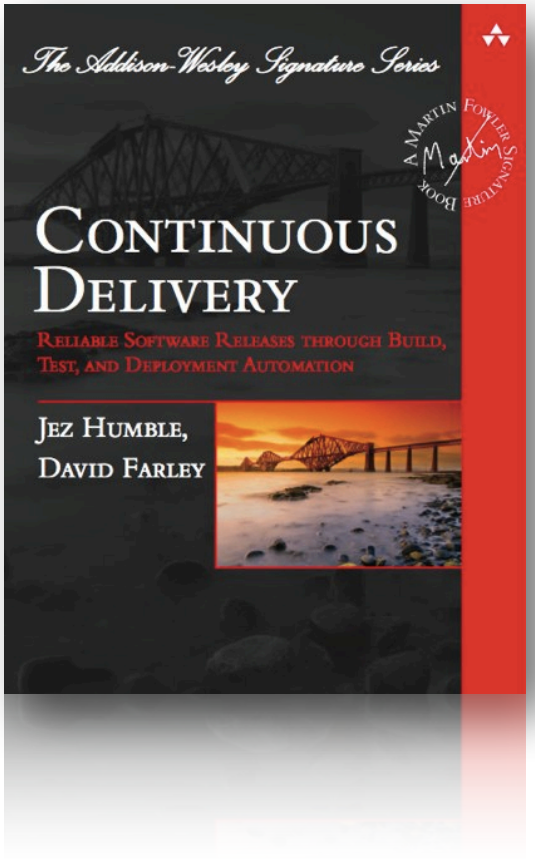


	A	B	C
1			
2			
3			
4			
5			
6			
7			
8			





Continuous Delivery



continuous...

integration

everyone commits to trunk at least once a day

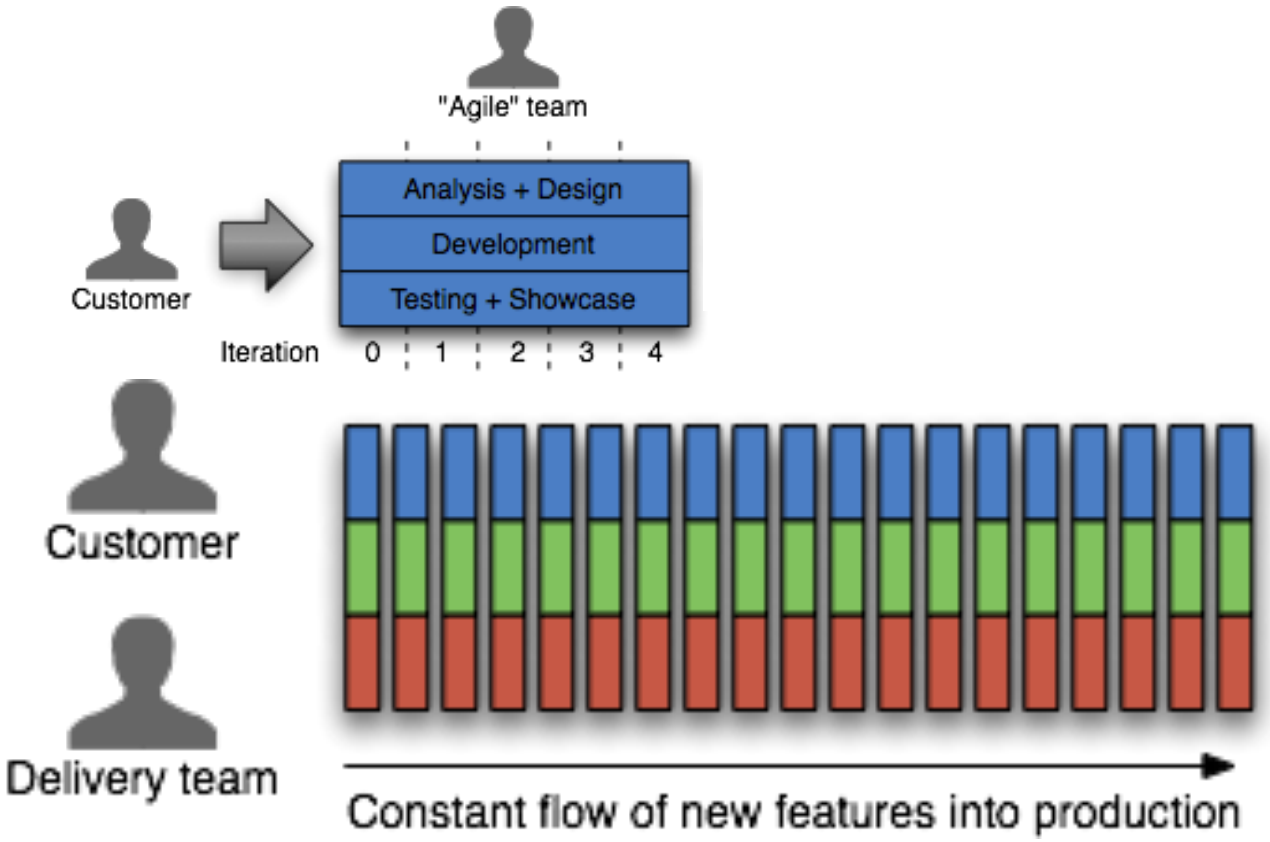
deployment

deploy as the last stage of continuous integration

delivery

software is always in a deployable state

agile 101



always production ready

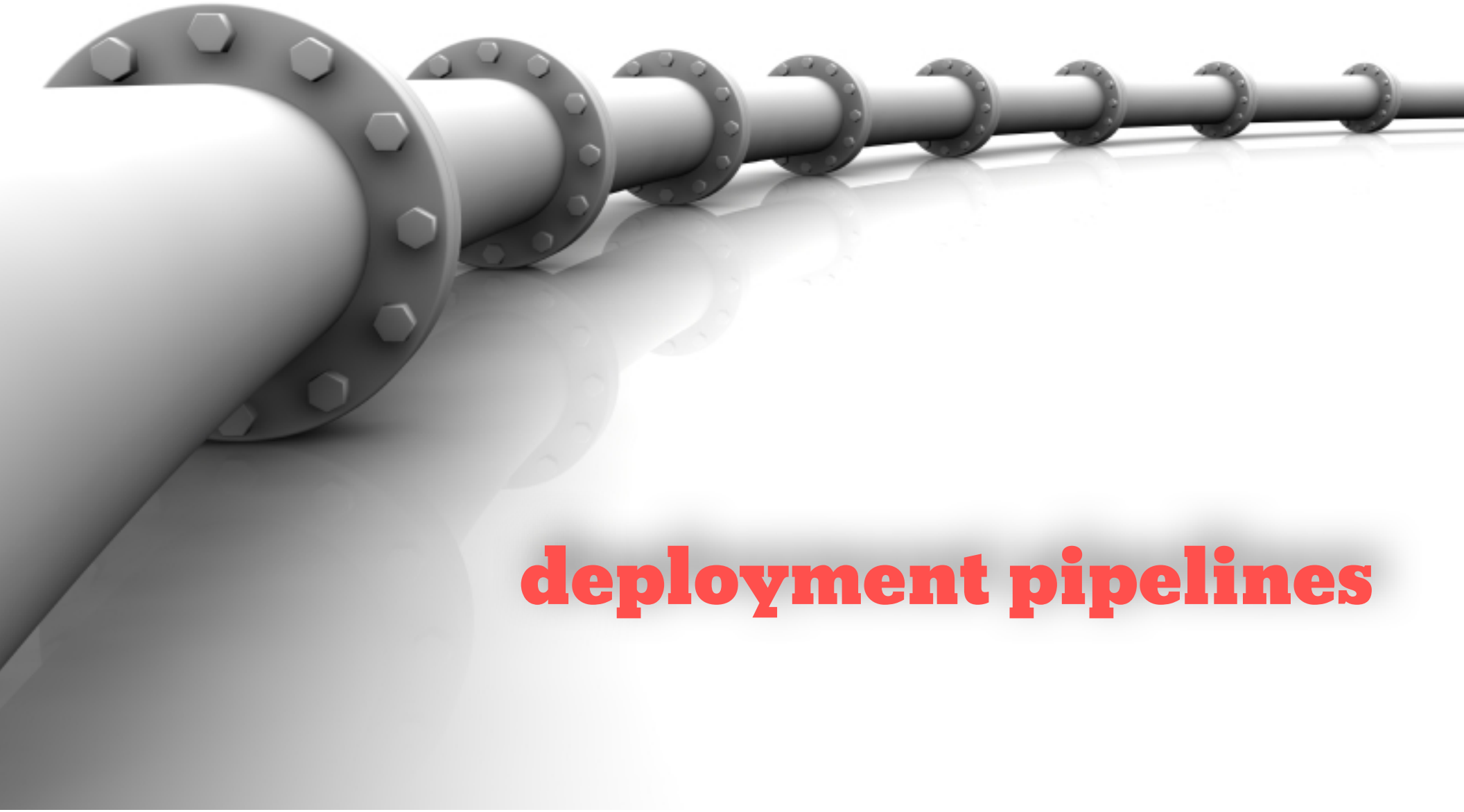
business needs > operational concerns

continuous integration

Fast, automated feedback on the correctness of your application every time there is a change to code

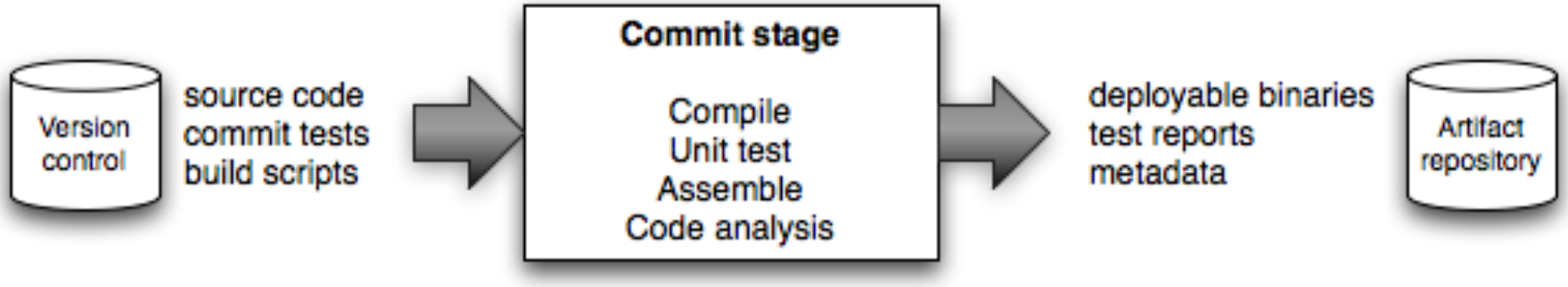
continuous delivery

Fast, automated feedback on the *production readiness* of your application every time there is a change — to *code*, *infrastructure*, or *configuration*



deployment pipelines

commit stage



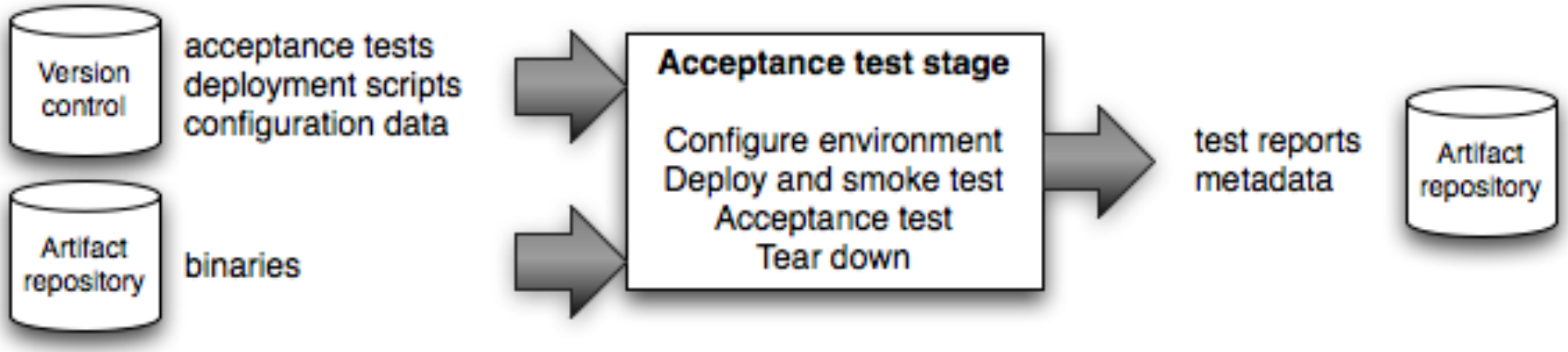
run against every check-in

mimics continuous integration

builds down-stream artifacts

if it fails, fix it immediately

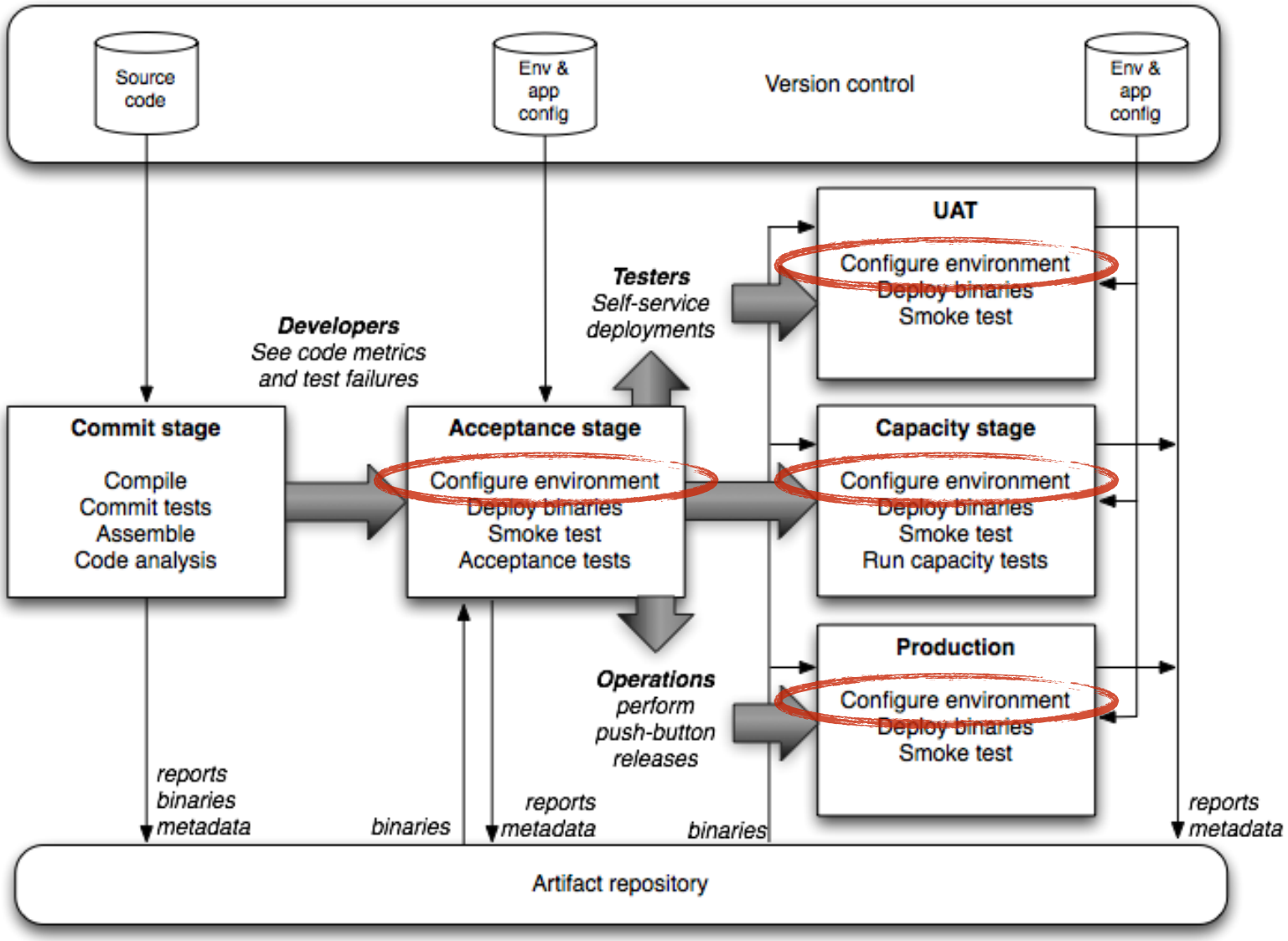
milestone (acceptance phase)



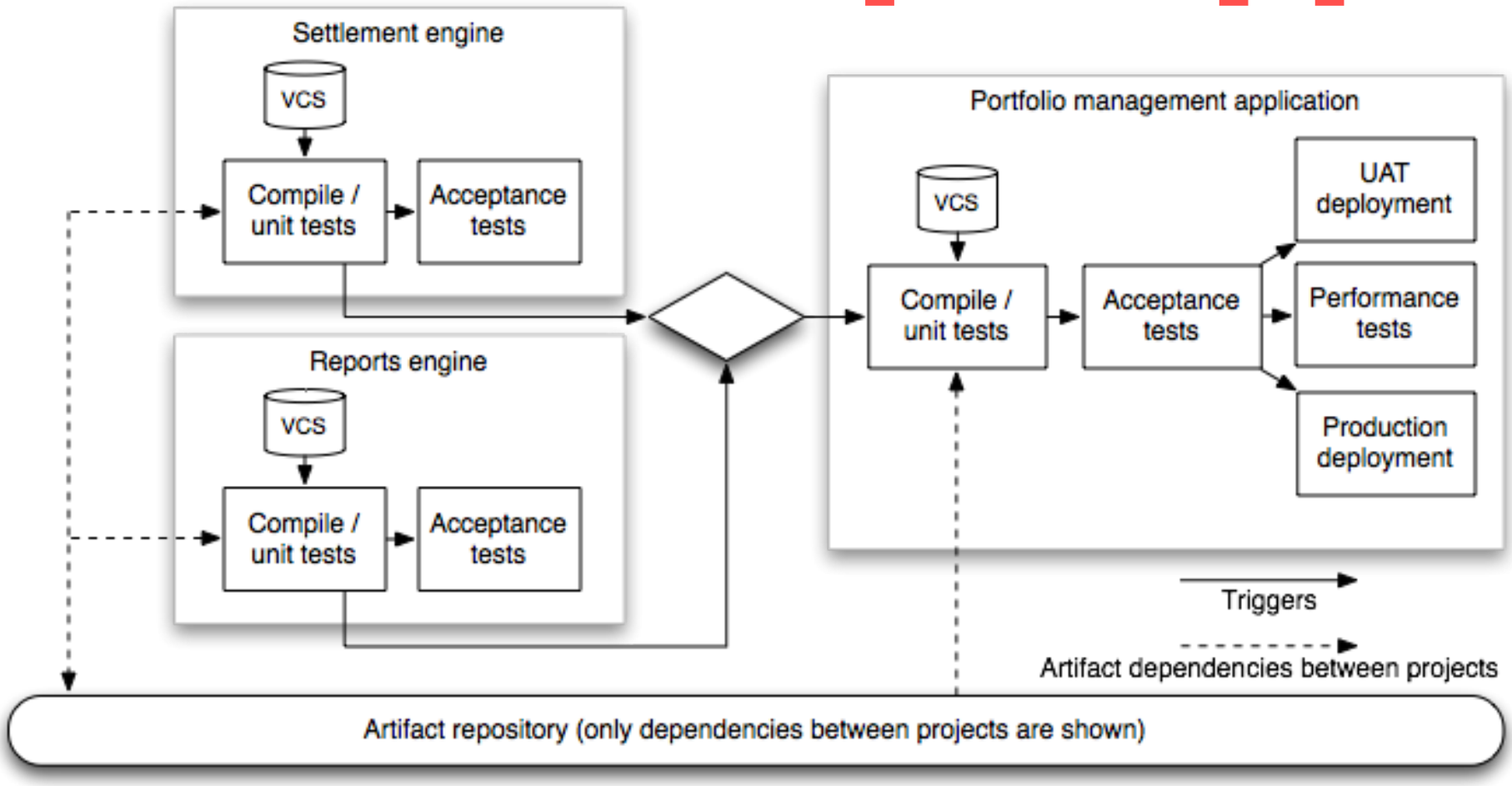
end-to-end tests in a production-like environment

triggered by upstream success

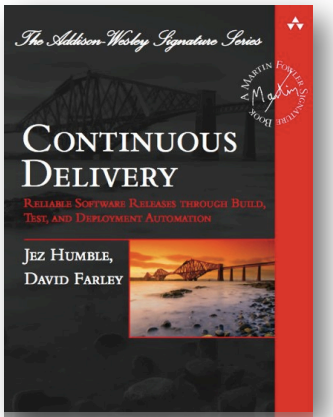
all downstream tests will create their environments

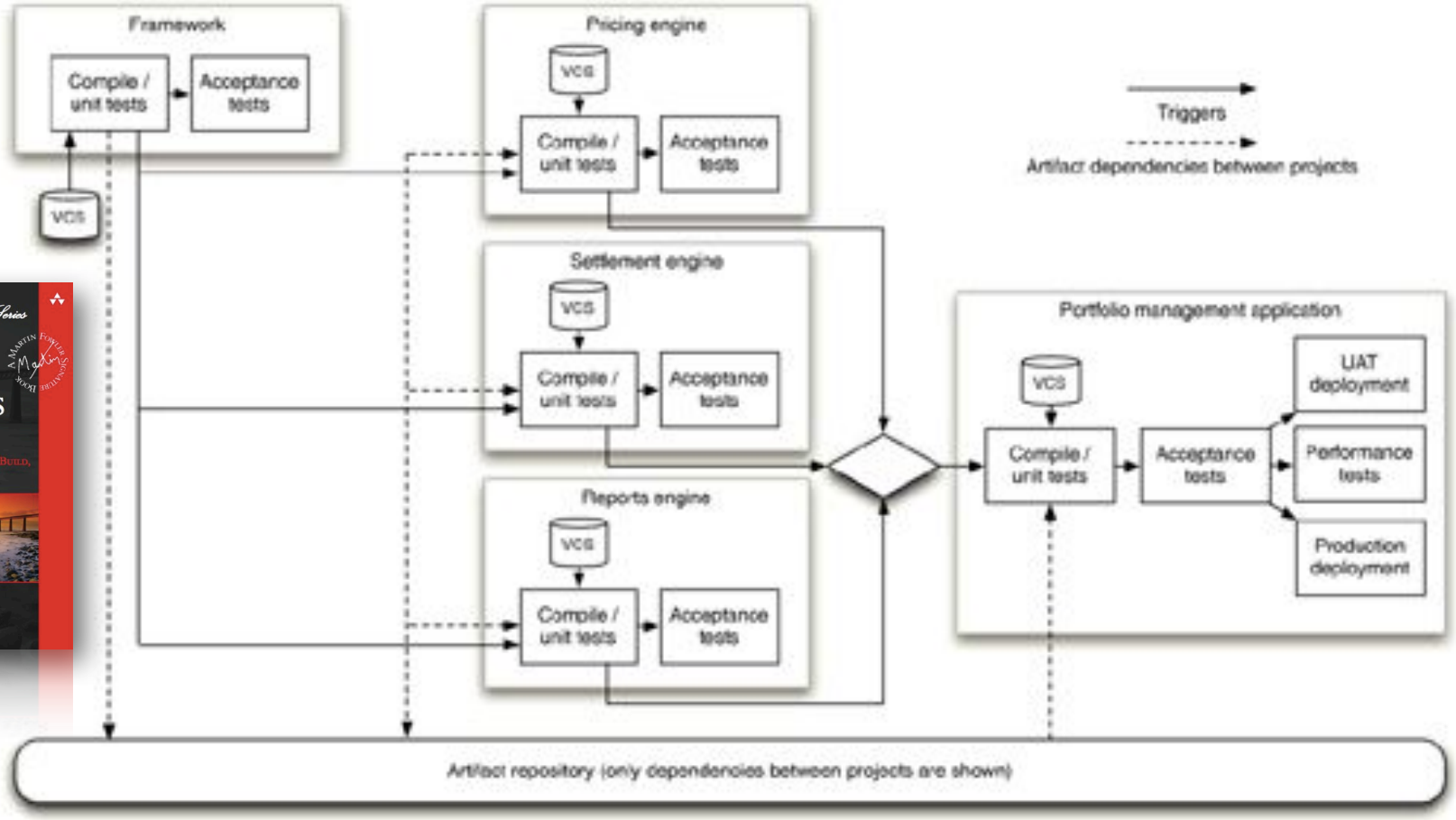

Source code
Env & app config
Version control
Env & app config
UAT
Configure environment
Deploy binaries
Smoke test
Testers
Self-service deployments
Developers
See code metrics and test failures
Commit stage
Compile
Commit tests
Assemble
Code analysis
Acceptance stage
Configure environment
Deploy binaries
Smoke test
Acceptance tests
Capacity stage
Configure environment
Deploy binaries
Smoke test
Run capacity tests
Production
Configure environment
Deploy binaries
Smoke test
Operations
perform push-button releases
reports
binaries
metadata
binaries
reports
metadata
binaries
reports
metadata
Artifact repository

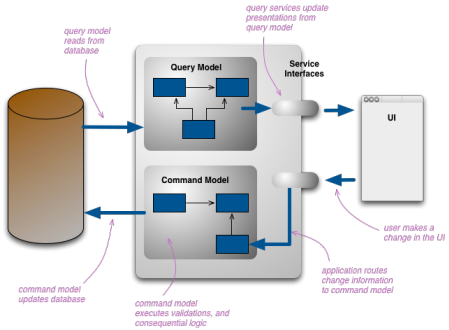
component pipeline

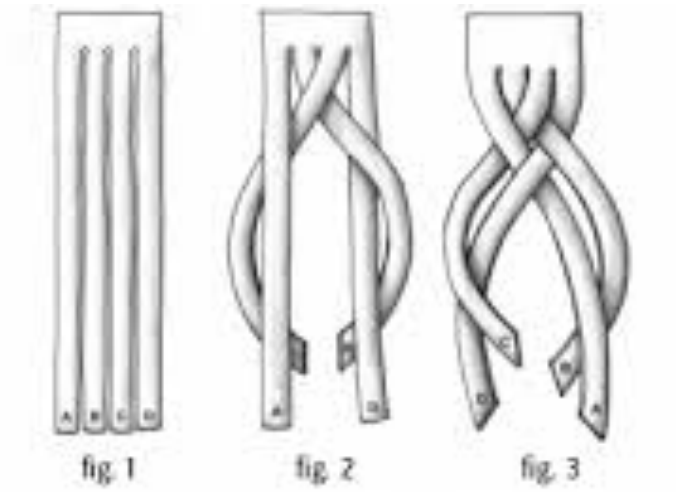


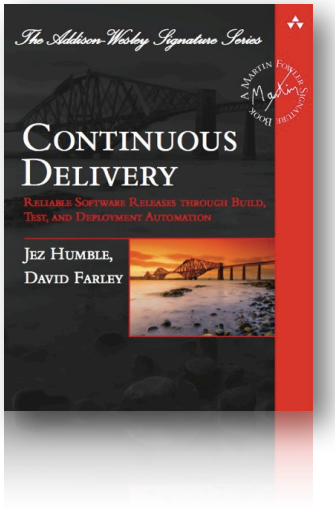
pipelining libraries



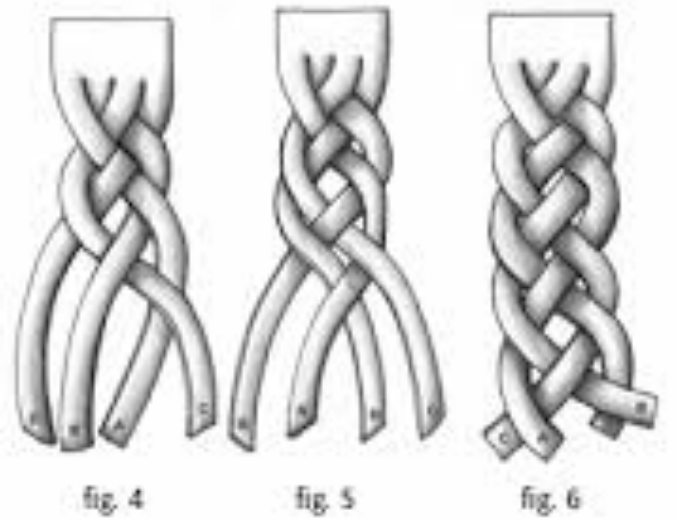


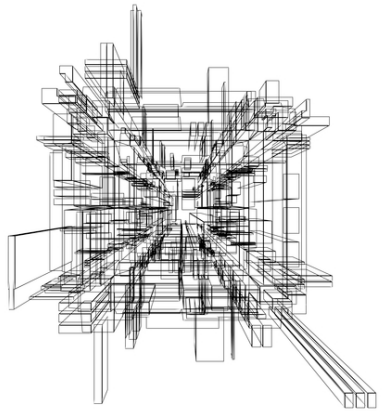




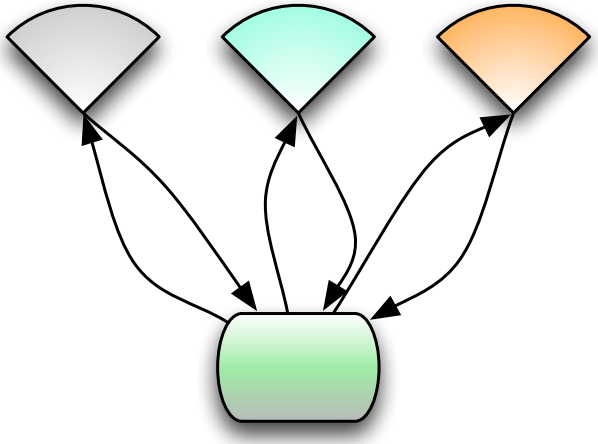


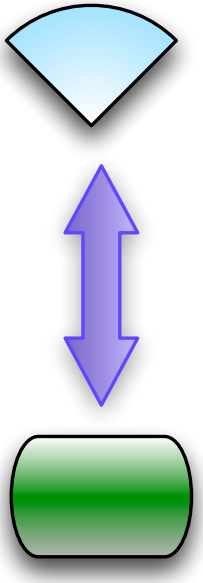
	A	B	C
1			
2			
3			
4			
5			
6			
7			
8			

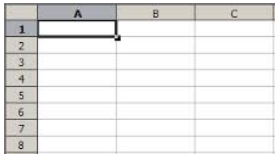




functional reactive programming







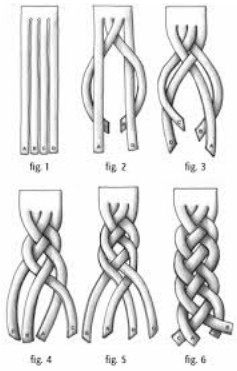
Java 8 & Streams

```
public class TheCompanyProcess {
    public String cleanNames(List<String> listOfNames) {
        StringBuilder result = new StringBuilder();
        for(int i = 0; i < listOfNames.size(); i++) {
            if (listOfNames.get(i).length() > 1) {
                result.append(capitalizeString(listOfNames.get(i))).append(",");
            }
        }
        return result.substring(0, result.length() - 1).toString();
    }

    public String capitalizeString(String s) {
        return s.substring(0, 1).toUpperCase() + s.substring(1, s.length());
    }
}
```

iterate

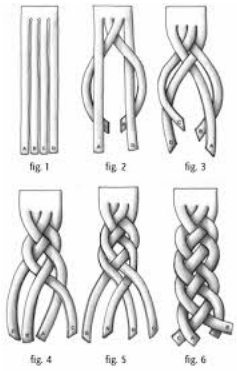
filter

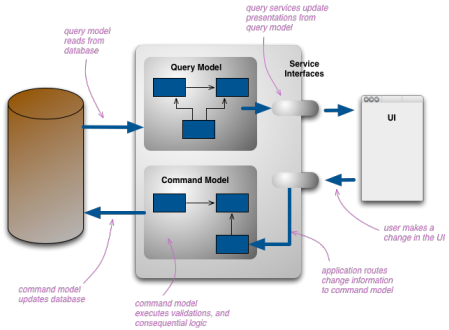
transform

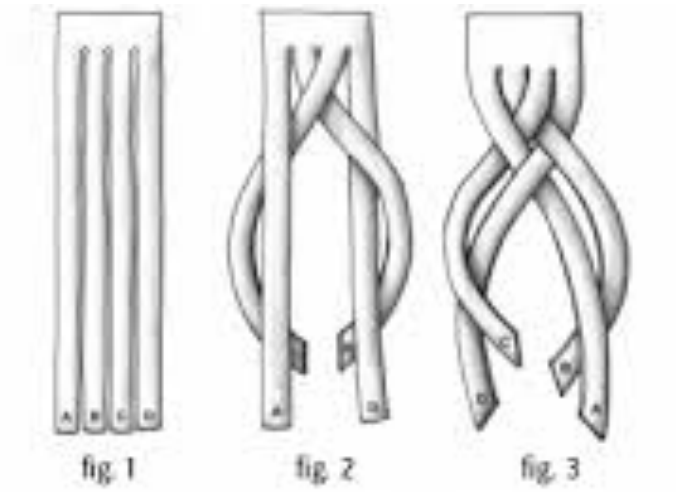


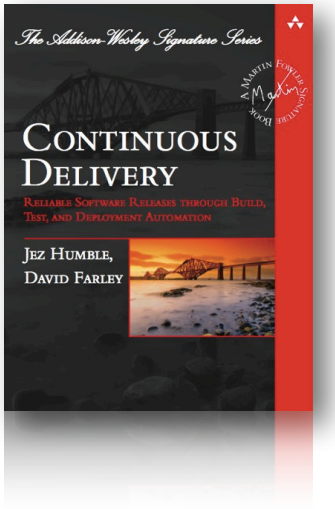
Java 8 & Streams

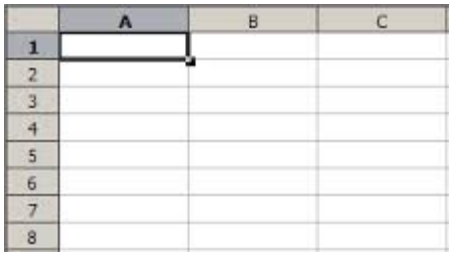
```
public String cleanNames(List<String> names) {
    return names
            .stream()
            .filter(name -> name.length() > 1)
            .map(name -> capitalize(name))
            .collect(Collectors.joining(","));
}

private String capitalize(String e) {
    return e.substring(0, 1).toUpperCase() + e.substring(1, e.length());
public String cleanNamesP(List<String> names) {
    return names
            .parallelStream()
            .filter(n -> n.length() > 1)
            .map(e -> capitalize(e))
            .collect(Collectors.joining(","));
}
```



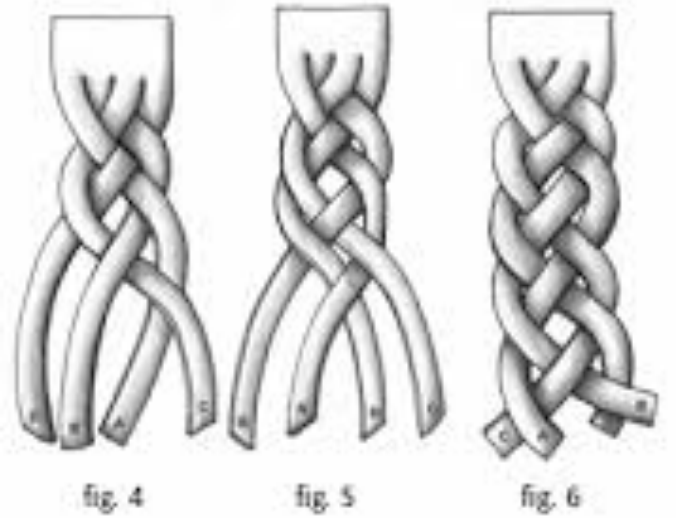


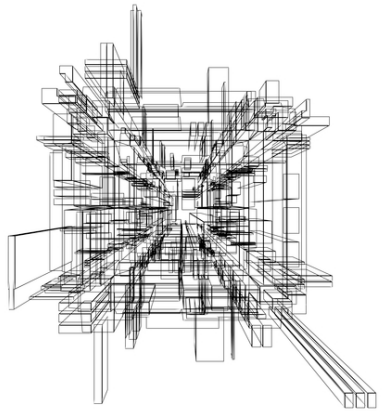




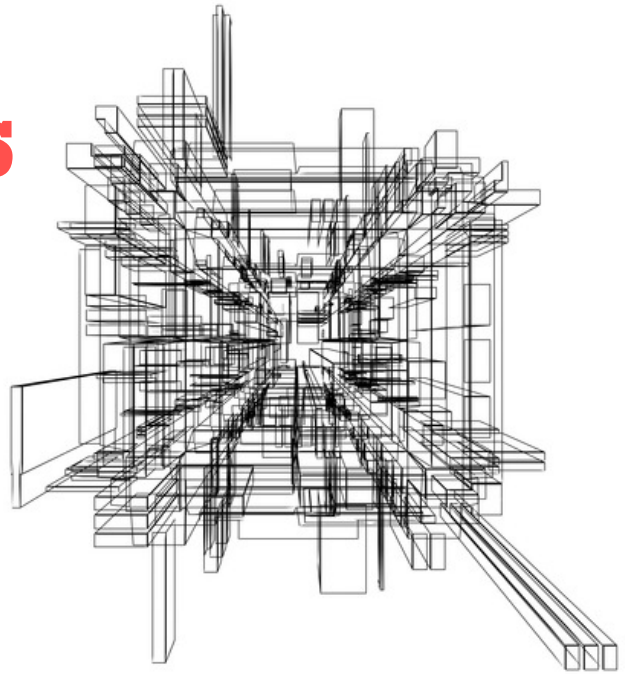


	A	B	C
1			
2			
3			
4			
5			
6			
7			
8			

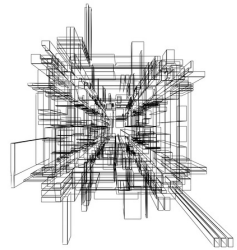


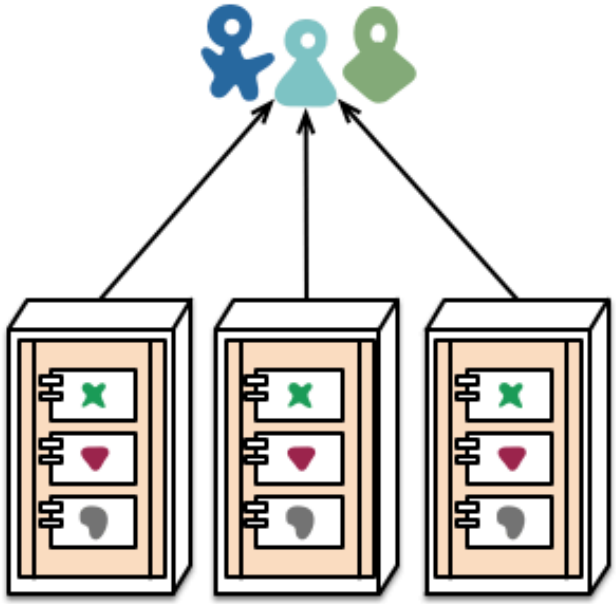


micro-services

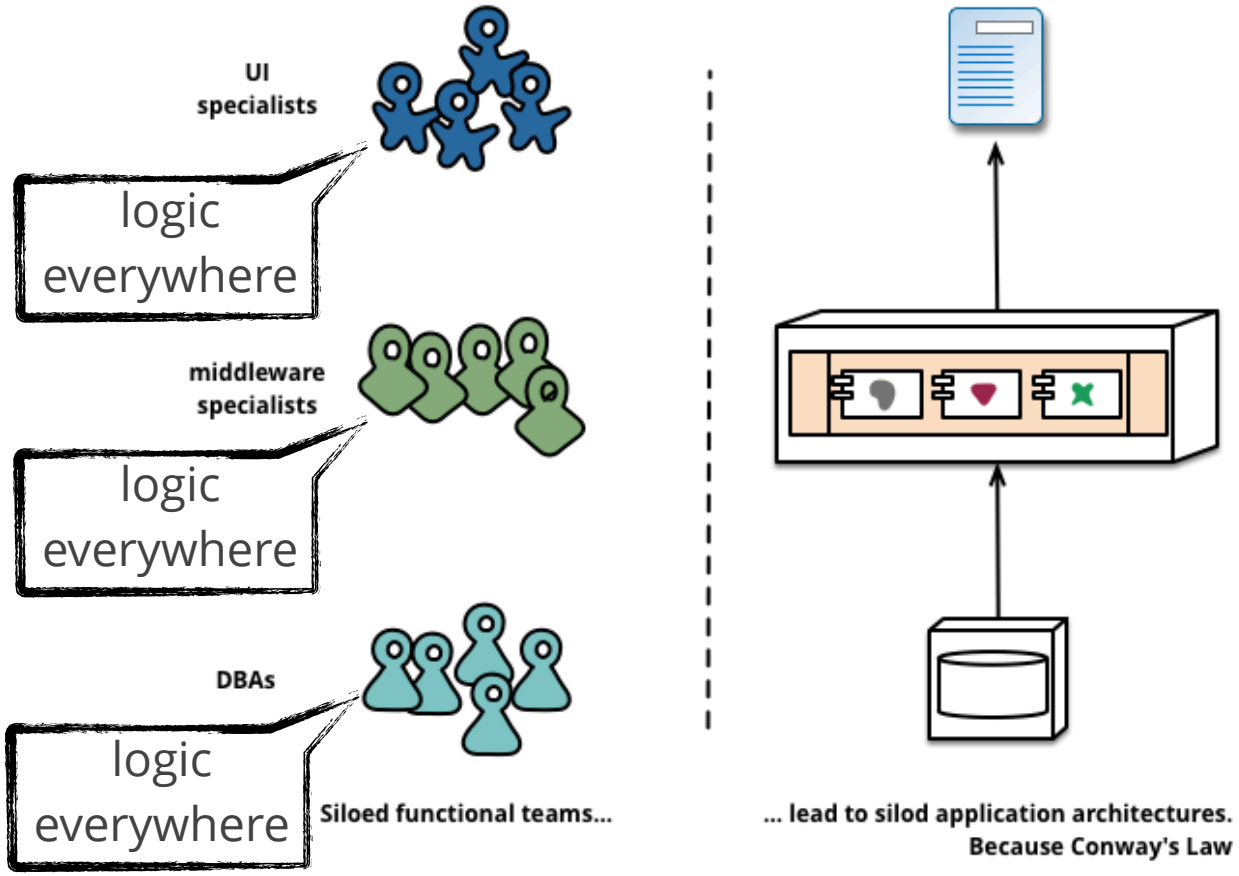


monoliths vs. micro-services

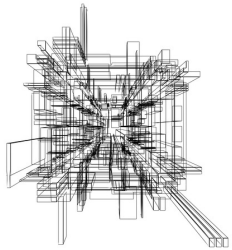


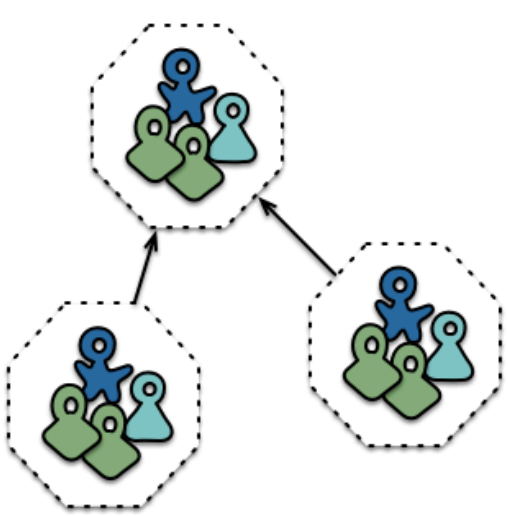


monolith - multiple modules in the same process

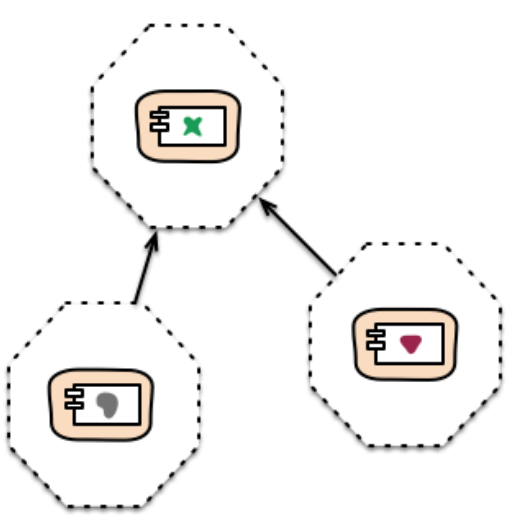


organized around business capabilities



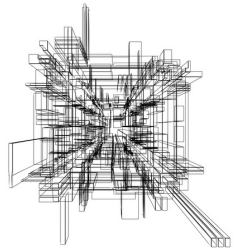


Cross-functional teams...

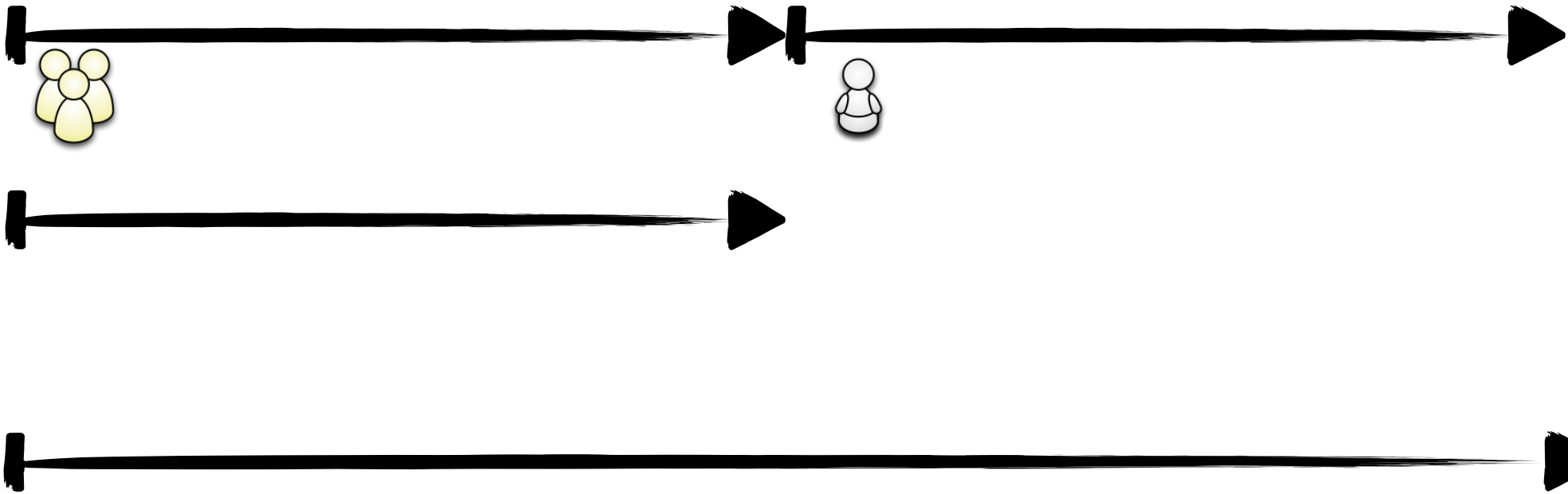


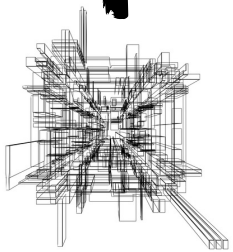
... organised around capabilities
Because Conway's Law

organized around business capabilities



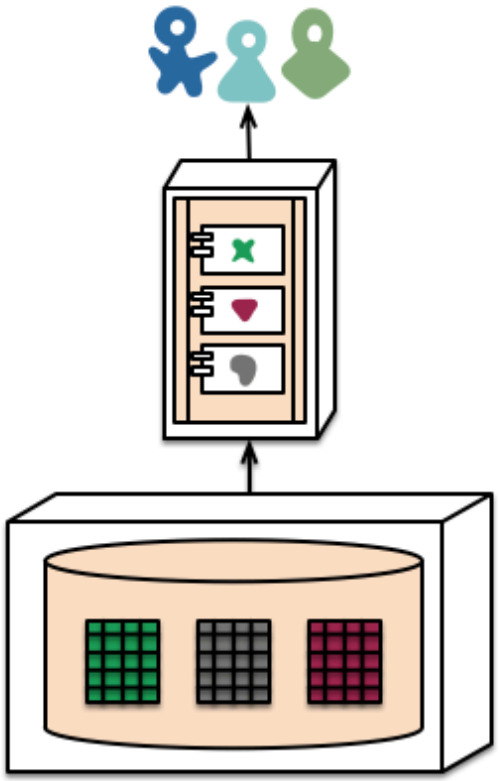
products, not projects



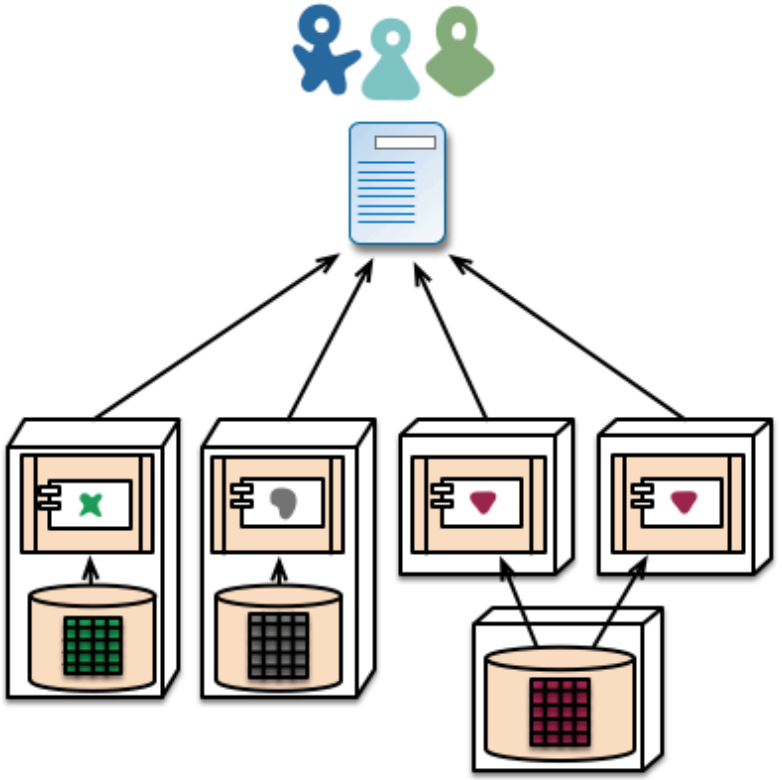


“you build it, you run it!”

decentralized data management

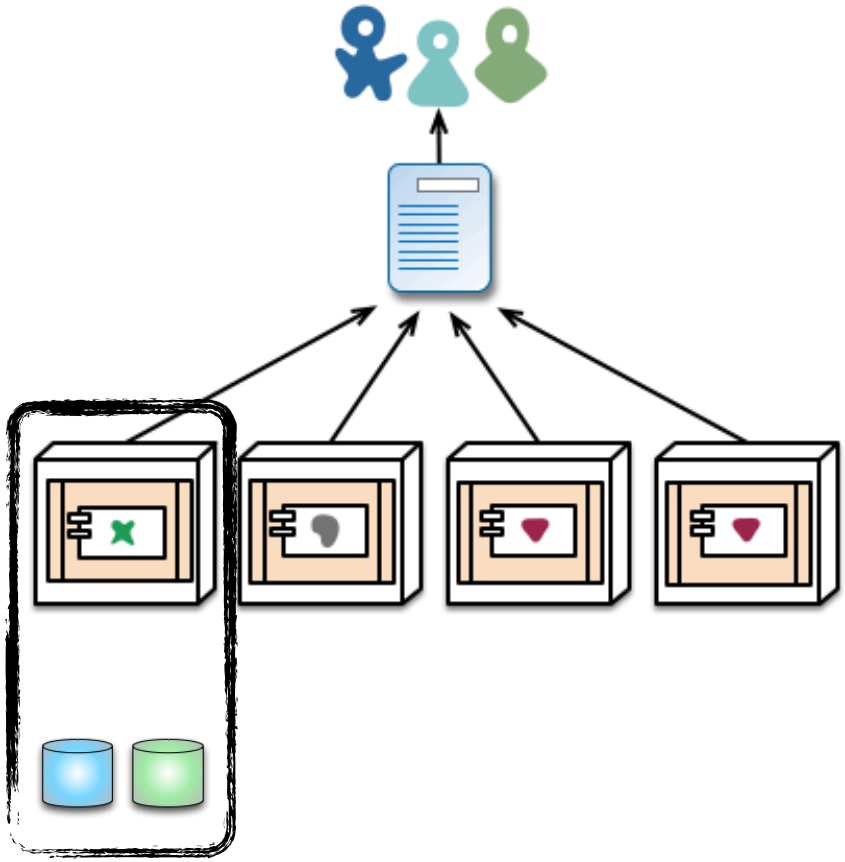


monolith - single database

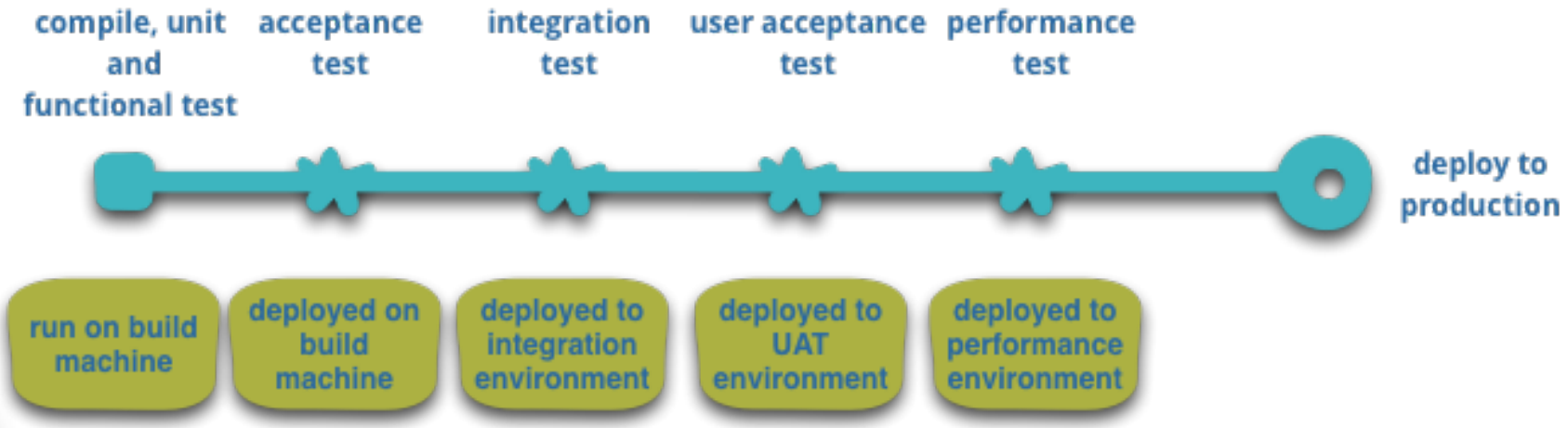


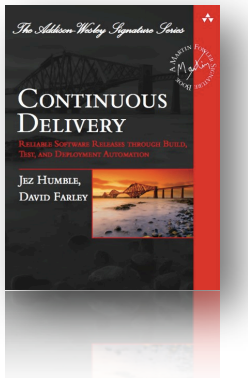
microservices - application databases

decentralized governance



infrastructure automation





design for failure

clients must respond gracefully to provider failure

aggressive monitoring:

— business relevant

— architectural

— semantic

small, single responsibility

“small enough to fit in your head”

don’t maintain — rewrite!

containerless, Unix services

embedded web container (Jetty / SimpleWeb)

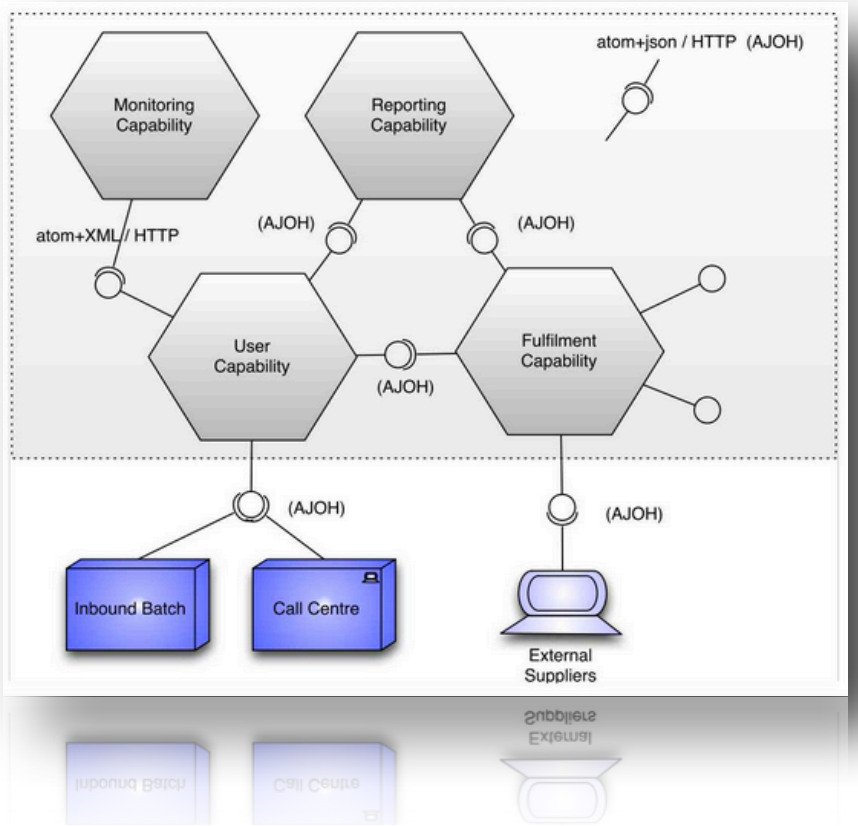
packaged as an executable

installed the same way as httpd & similar

Micro Services

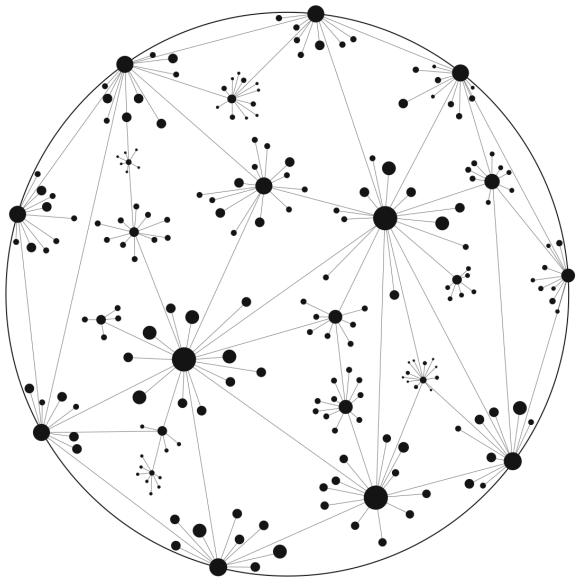
Java, the Unix way

James Lewis tells the story of building a resource oriented, event driven system out of applications about 1000 lines long.



http://www.infoq.com/presentations/Micro-Services

holistic engineering



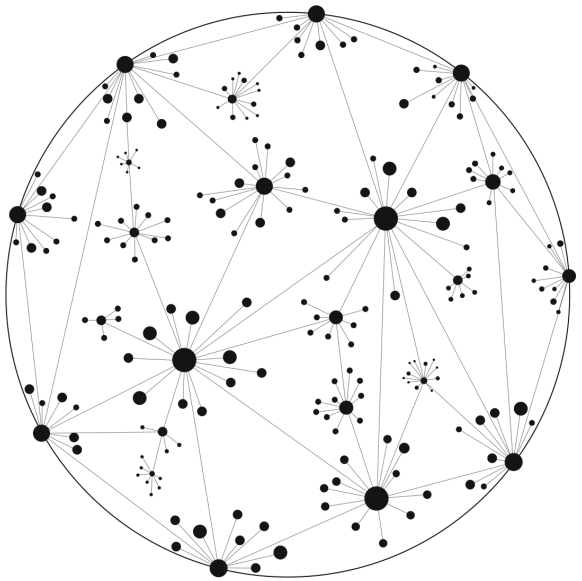
don’t over-optimize around a particular tool or practice and harm your overall engineering efficiency

everything interconnects

cannot separate process from architecture

tools offer a primrose path

- awesome while you're on the path
- awful when you need to step off



http://kent.spillner.org/blog/work/2009/11/14/java-build-tools.html

"Maven builds are an infinite cycle of despair that will slowly drag you into the deepest, darkest pits of hell (where Maven itself was forged)."

composable

Rake Gant

languages

- less implicit behavior
- better building blocks
- greater eventual power
- less initial power
- more flexibility

contextual

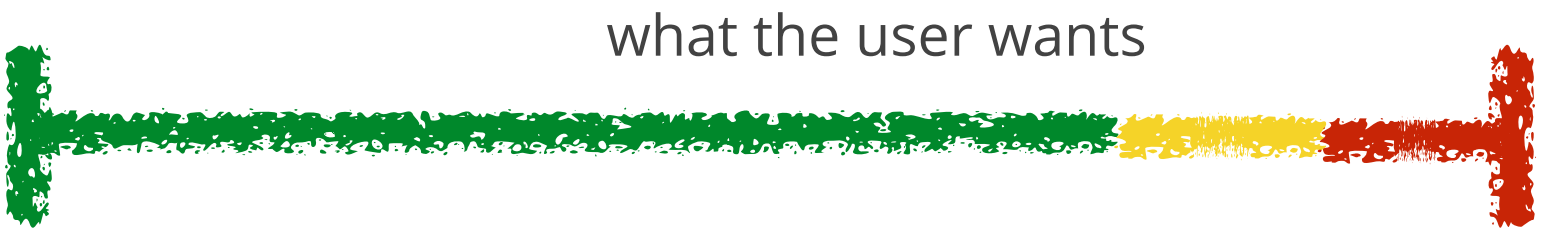
PowerShell

maven

frameworks

- more “out of the box”
- better contextual intelligence
- less flexibility
- less ability to evolve

Dietzler's Law



"Users always want 100% of what they want."

how do you choose?!?

generic

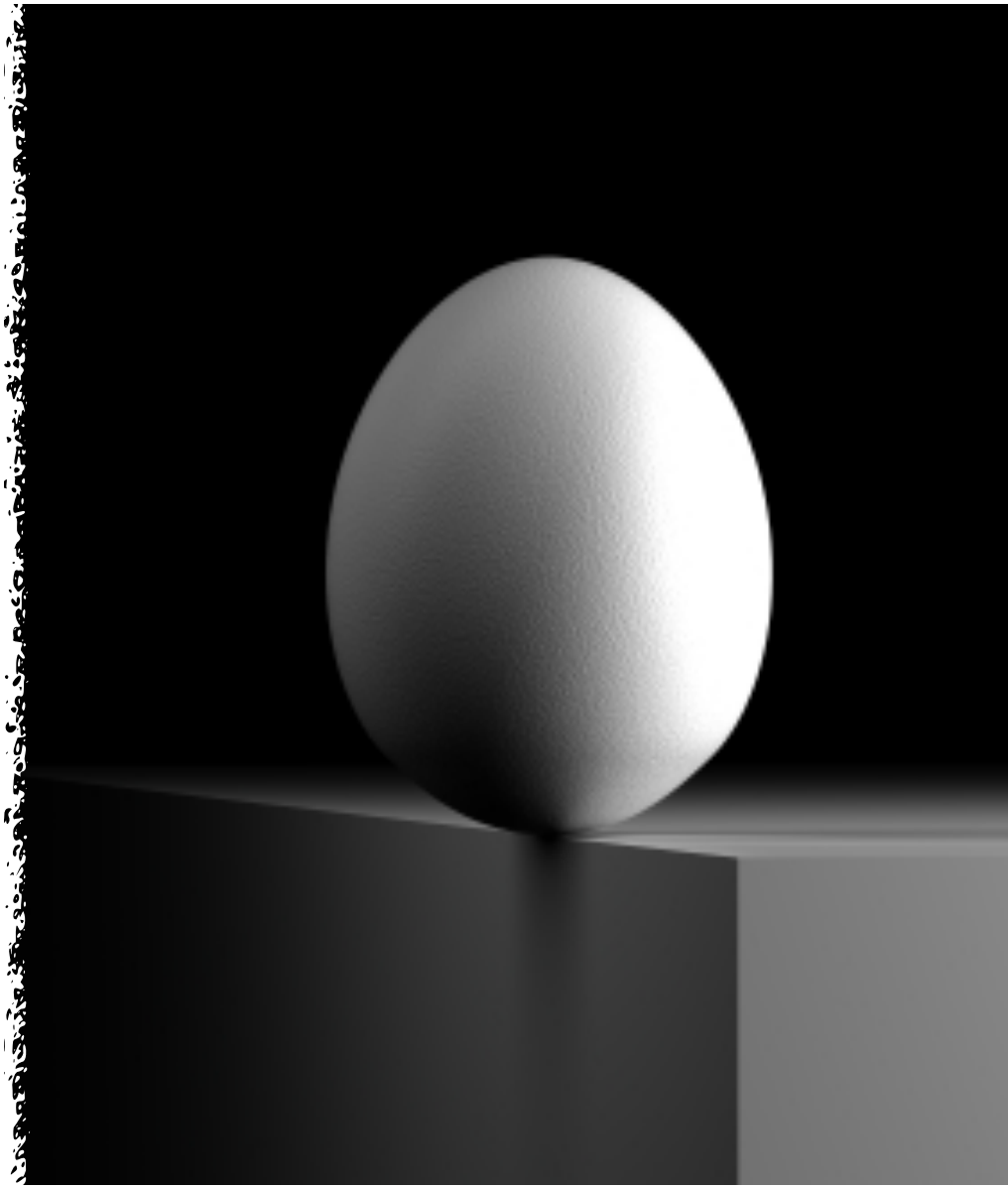
opinionated

start with the easiest

rigid

dogmatic

never
wonderful
again



cut & run!

malleability

malleable, n. - able to be hammered or pressed permanently out of shape without breaking or cracking.

capable of being altered or controlled by outside forces

emphasize malleability

always tends towards less μ

annealing

refactoring and restructuring exercises require increasing effort for the same result

plan escalating effort towards remedial architecture & design

tradeoff for reduced up-front effort

anneal, n. - heat (metal or glass) and allow to cool slowly, in order to remove internal stresses and toughen it

- greenfield projects => emphasize malleability
- brownfield projects => maximize annealing efforts

prefer pro/reactive to predictive

remove friction

simplify, un-tangle

evolve

deliver !


DEVNATION
April 13-17, 2014
San Francisco, California
Thank you
Enjoy the conference!
Agile Architecture &
Design
Neal Ford
Director / Software Architect /
Meme Wrangler
ThoughtWorks®
2002 Summit Blvd, Level 3, Atlanta, GA 30319, USA
T: +1 40 4242 9929 Twitter: @neal4d
E: nford@thoughtworks.com W: thoughtworks.com