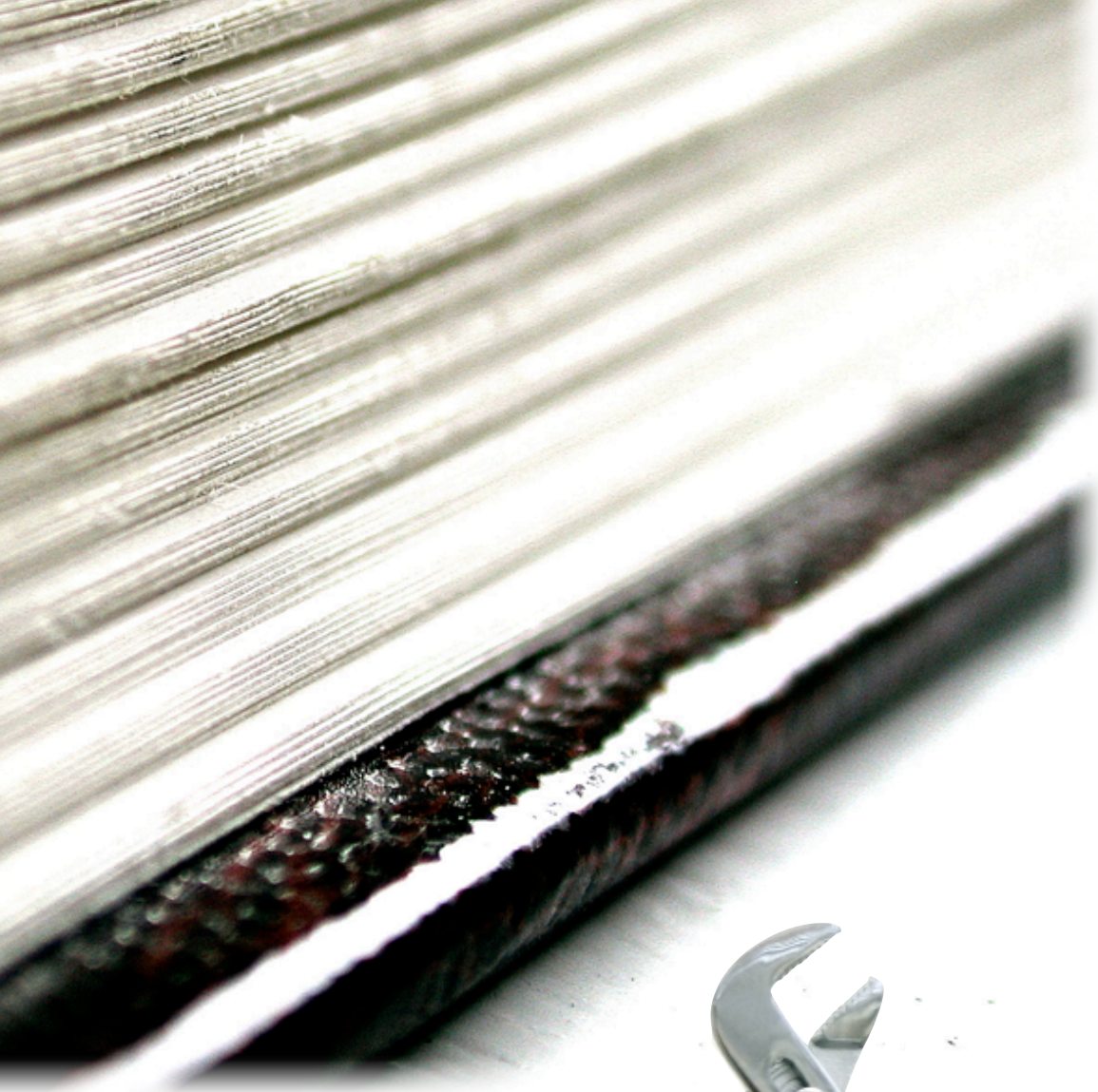
tooling & documentation



documentation

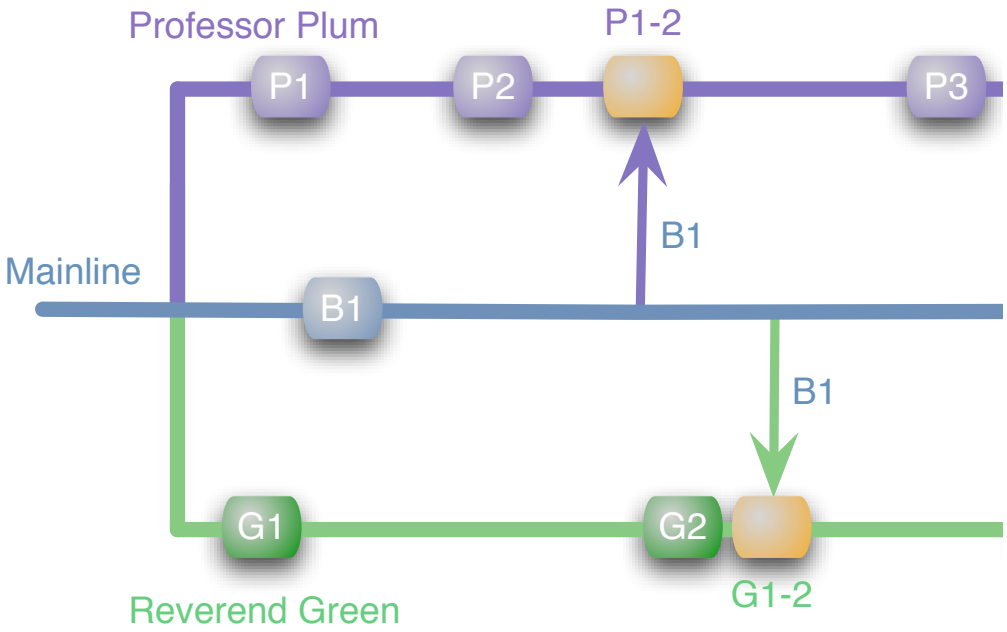
tools

tools

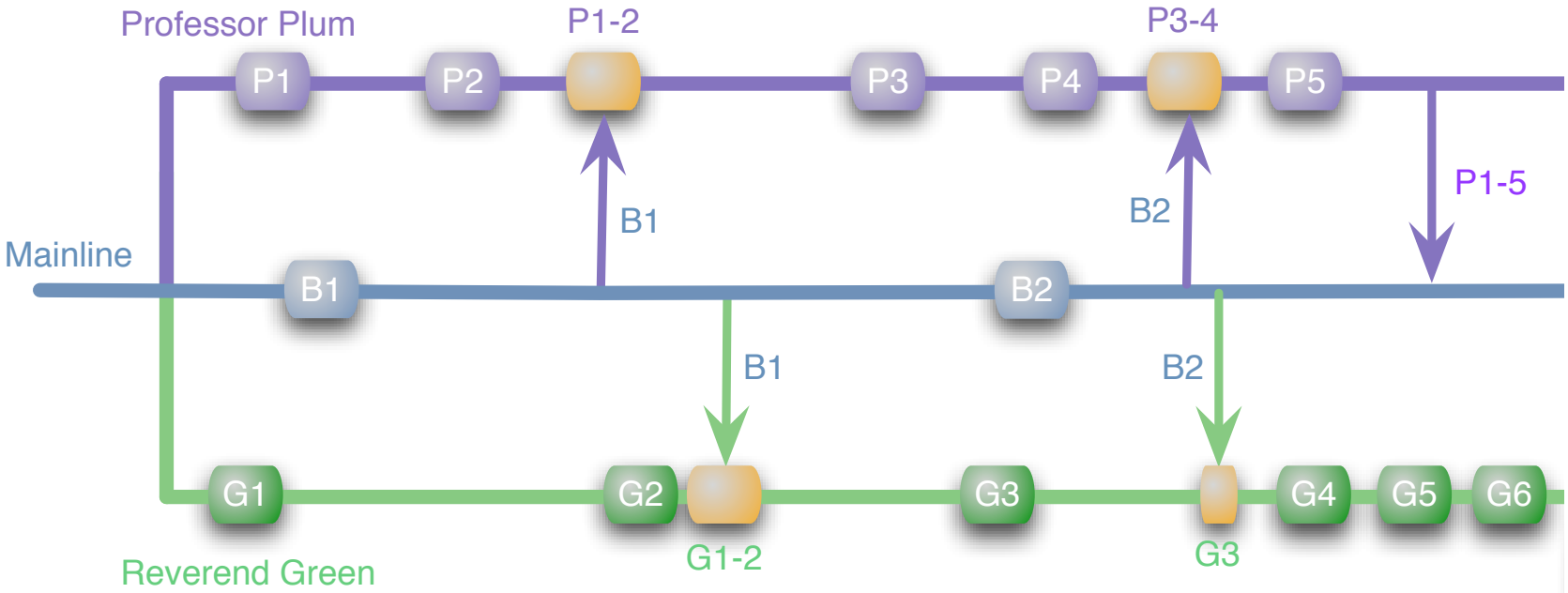
you get to pick the tools & practices!

now what?!?

example

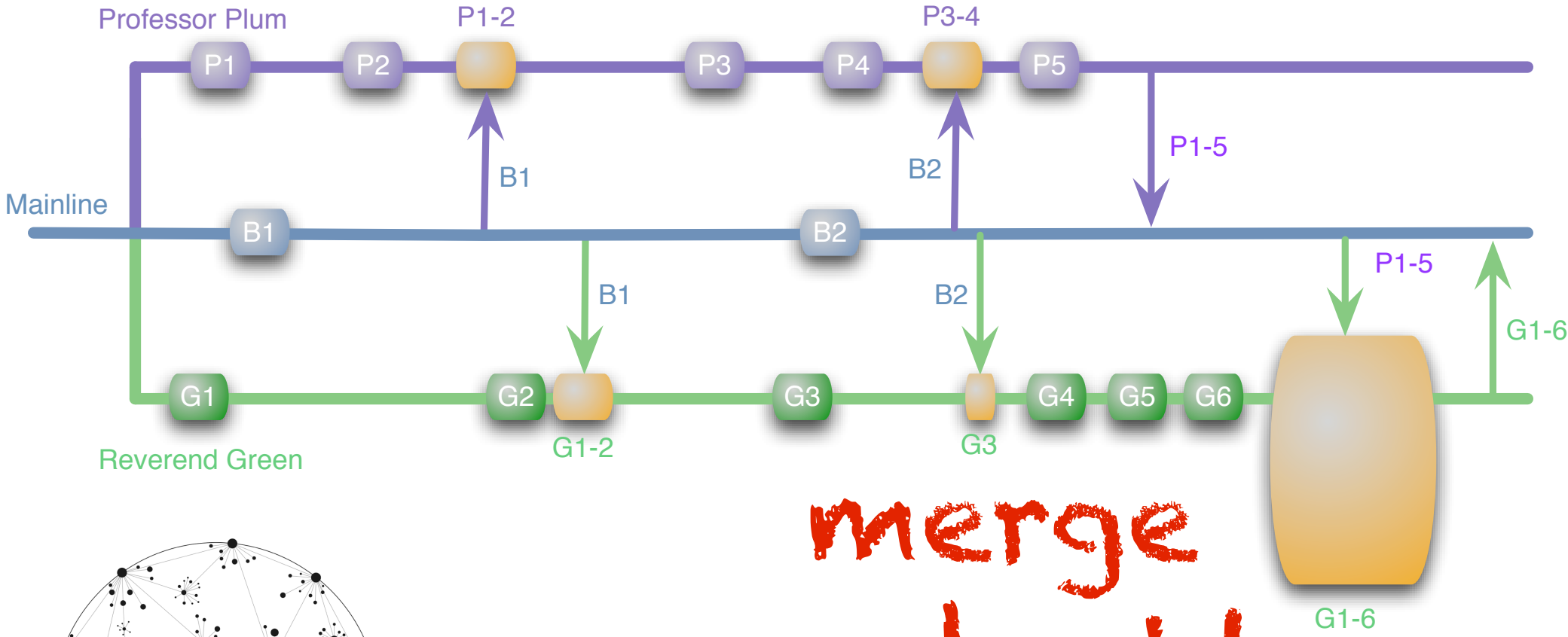


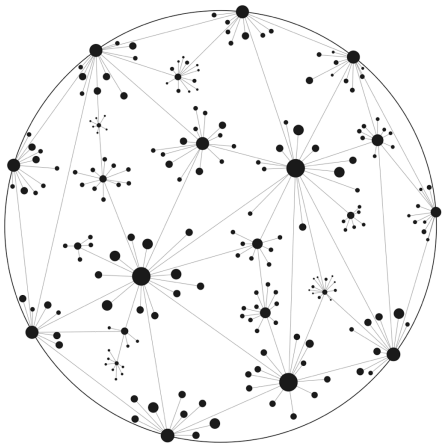
Feature Branching



Feature Branching

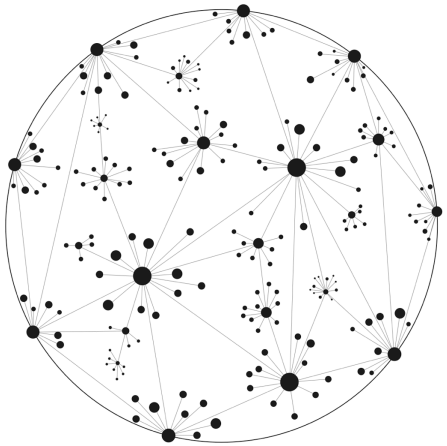
copy/paste reuse !!





merge ambush!

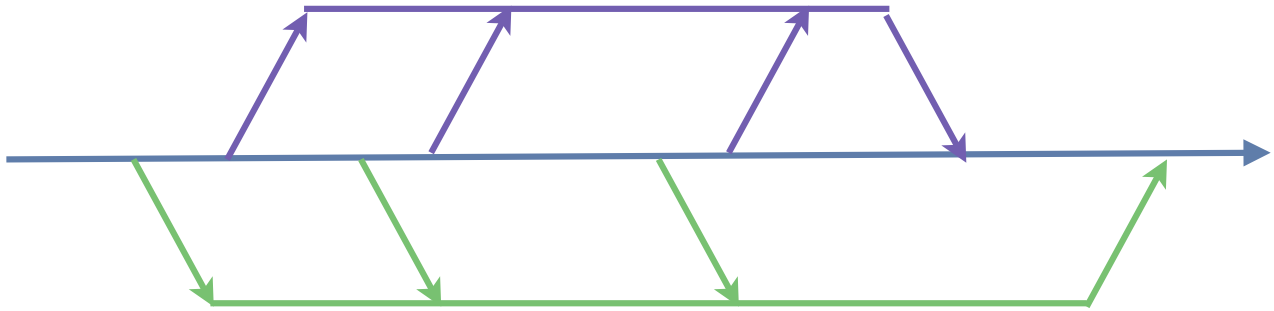
Feature Branching

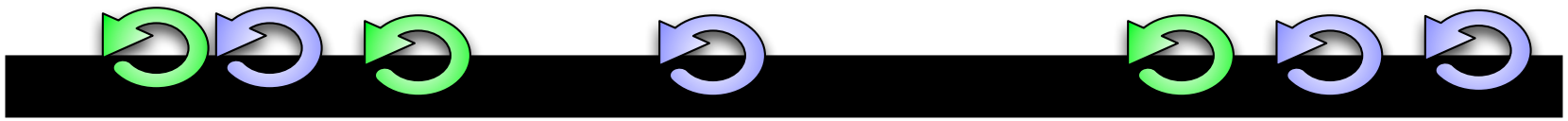


holistic approach to engineering practices

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

Feature Branch





Continuous Integration

everything interconnects

you can't separate process (continuous delivery) from architecture

tools offer a primrose path

- heavenly while on the road
- hellish to take a detour

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)."

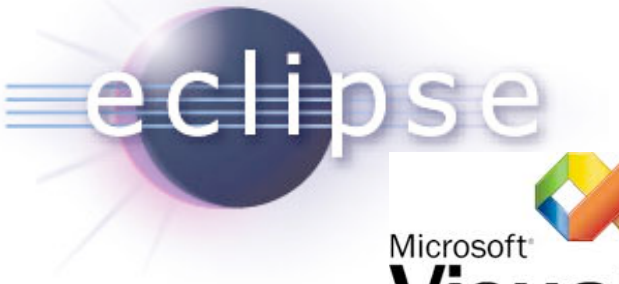
composable

Rake **Gant**

languages

contextual

PowerShell



frameworks

more context

more “out of the box”

better contextual intelligence

less flexibility

less ability to evolve

composable

less implicit behavior

better building blocks

greater eventual power

less initial power

more flexibility

Dietlzer's Law

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

80% 10% 10%

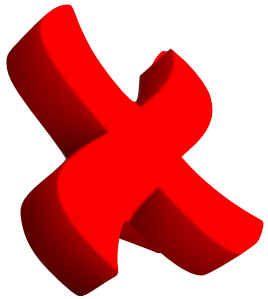
what the user wants

how do you choose?!?

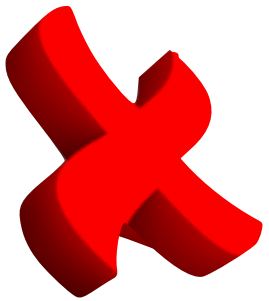
generic

opinionated

start with easiest...

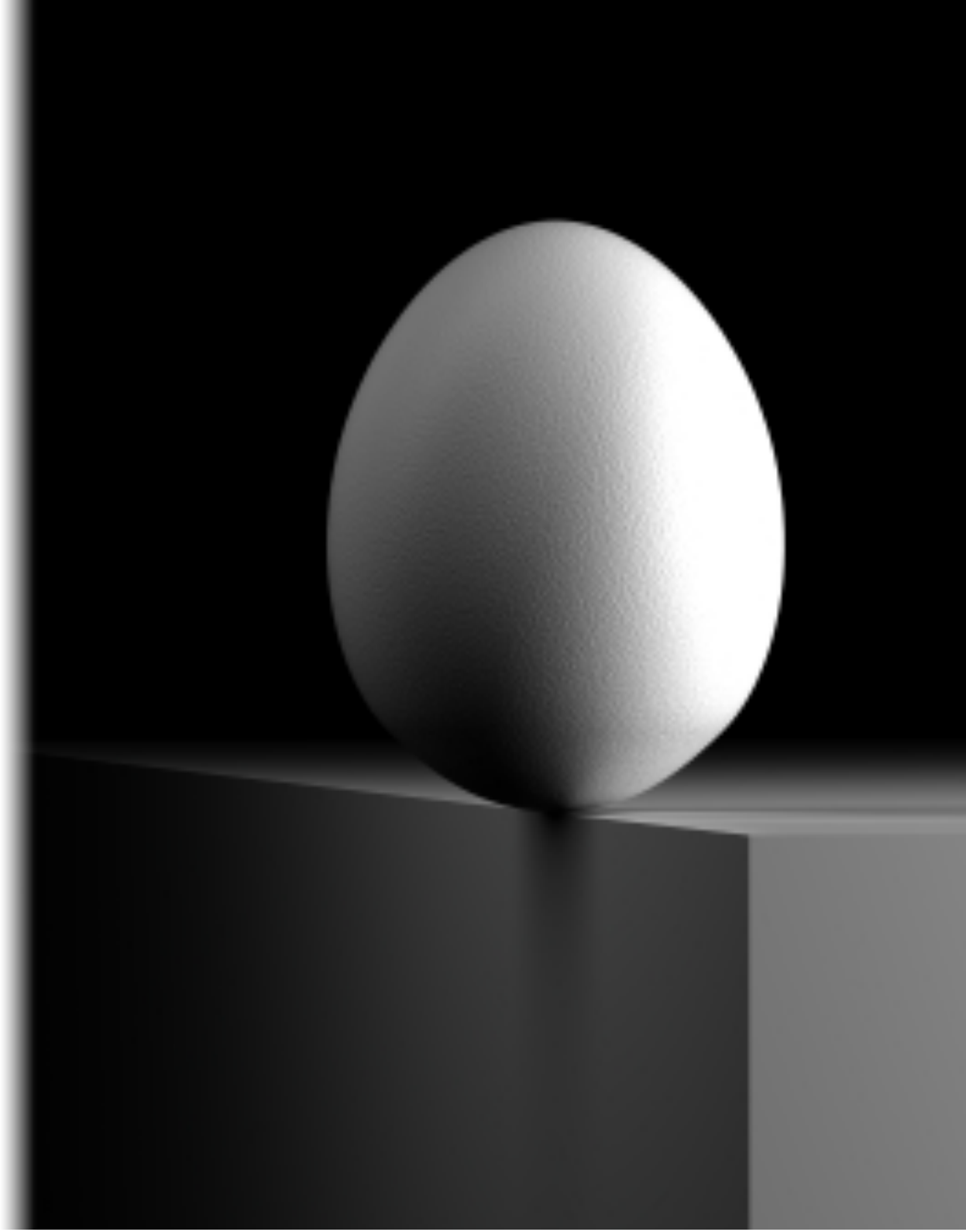


rigid



dogmatic

never
wonderful
again



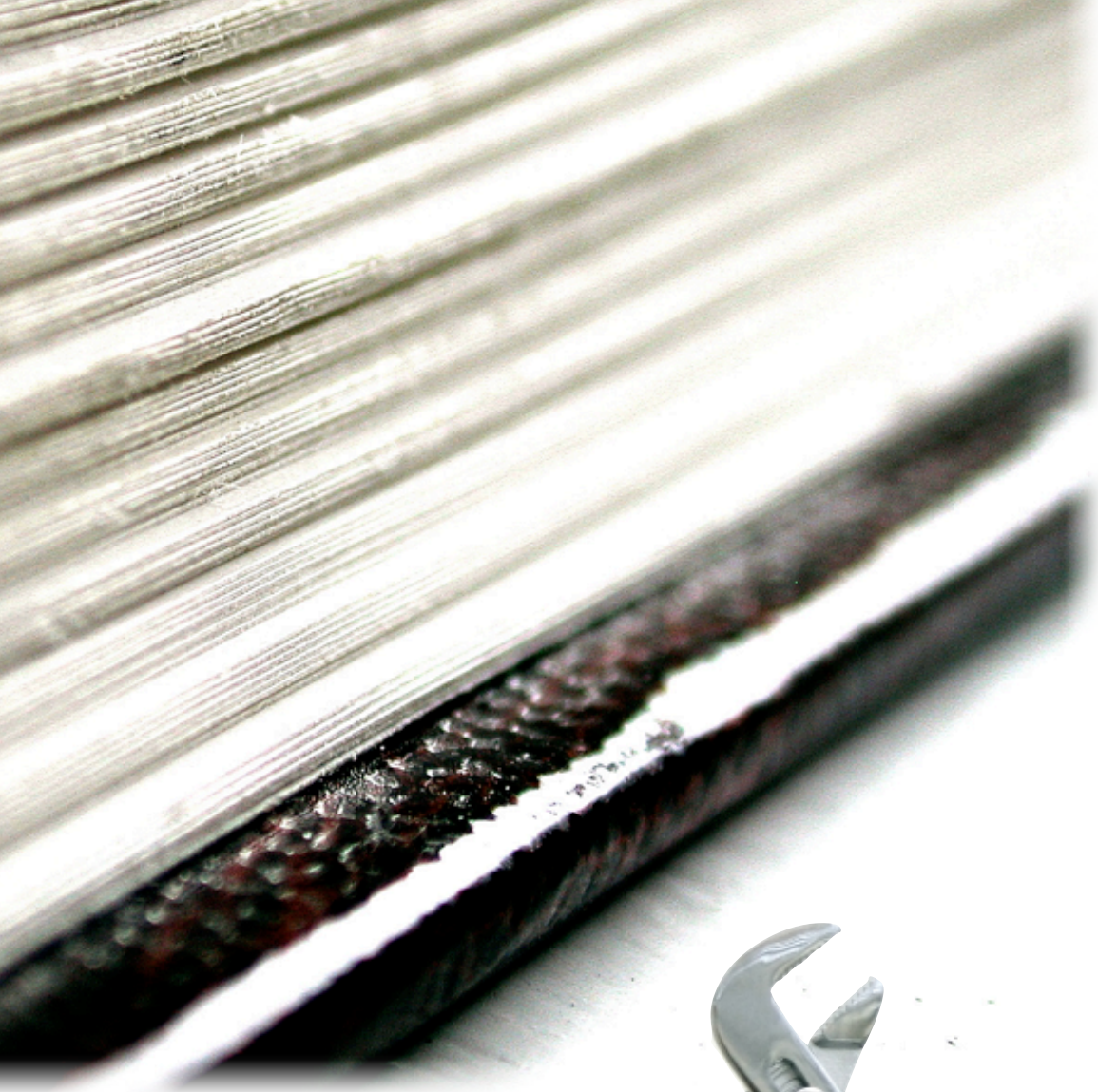
architect for change

pick the best tool for the job & the lifecycle station

reevaluate when friction appears

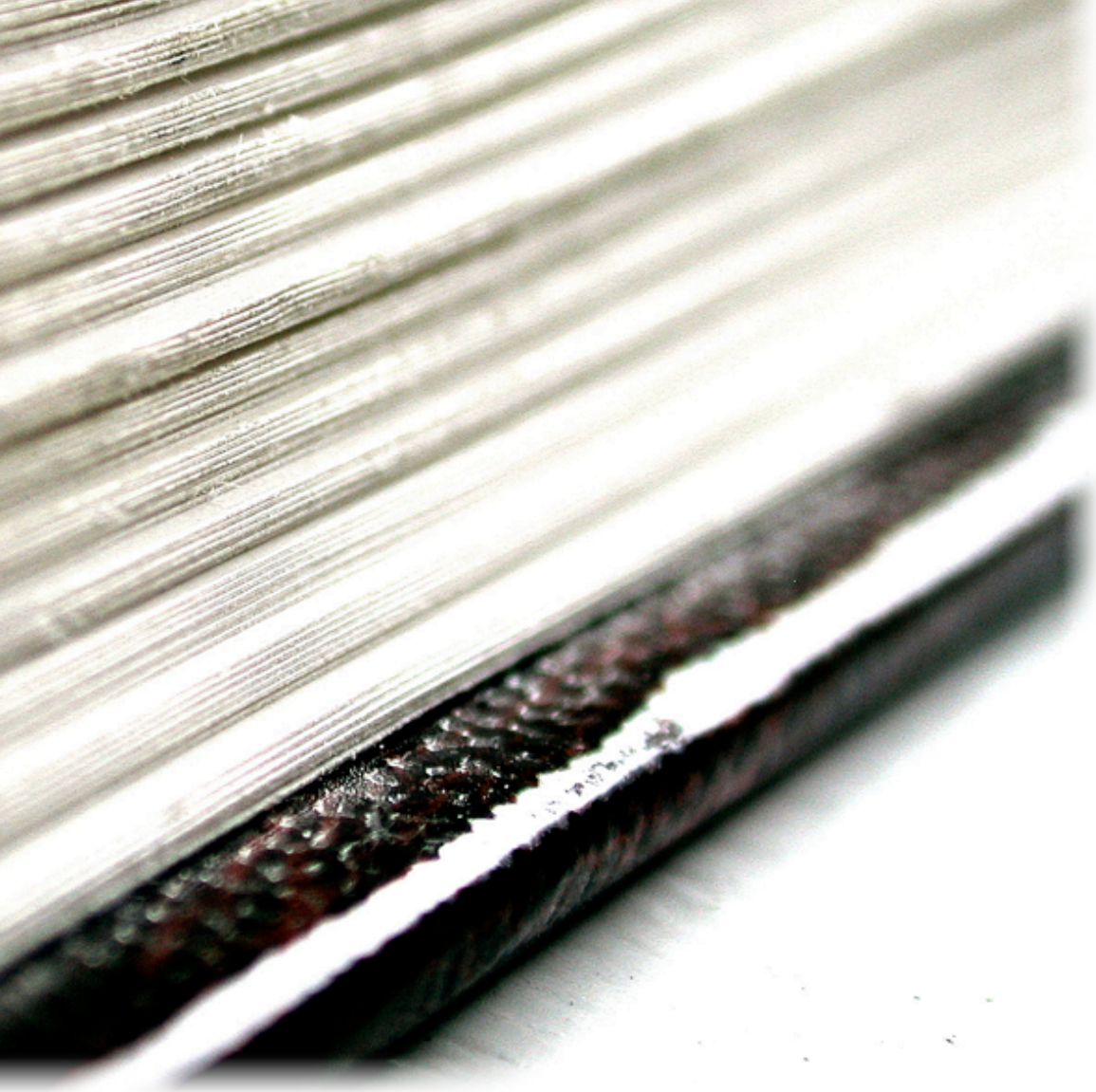
invest in suitable replacements

build anti-corruption layers



documentation

tools

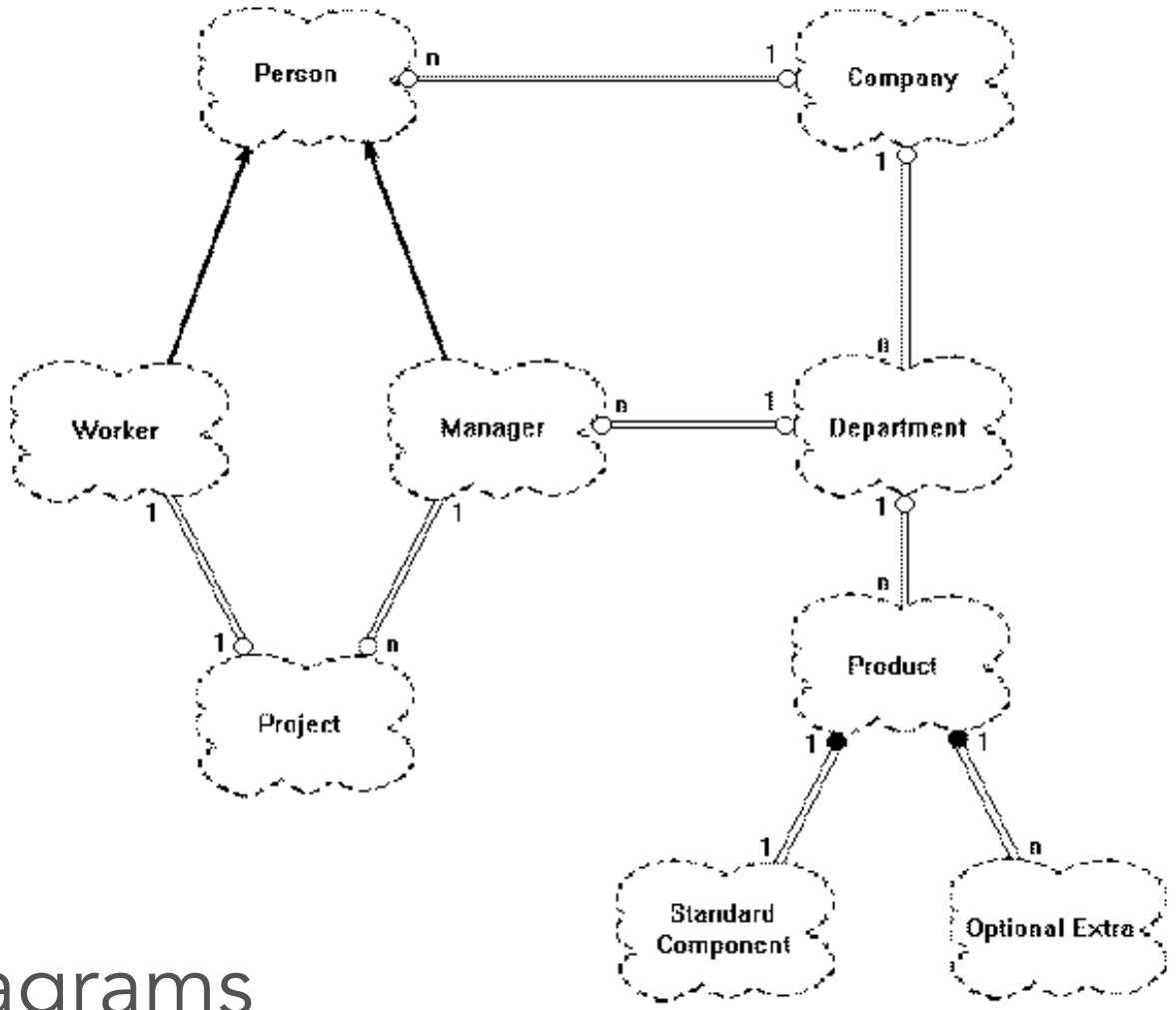


documentation

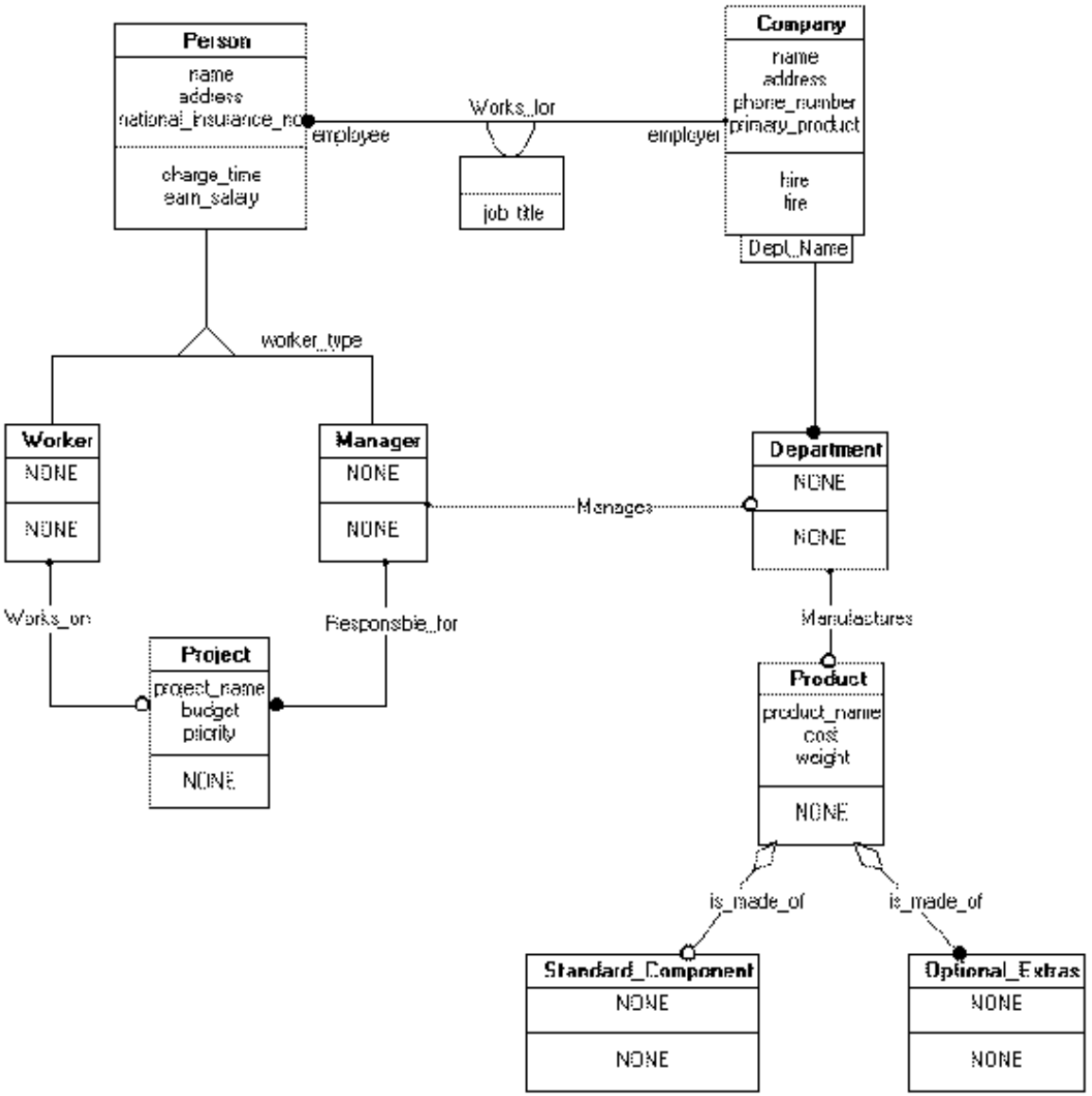
how do you document architecture?

way back when...

Booch diagrams

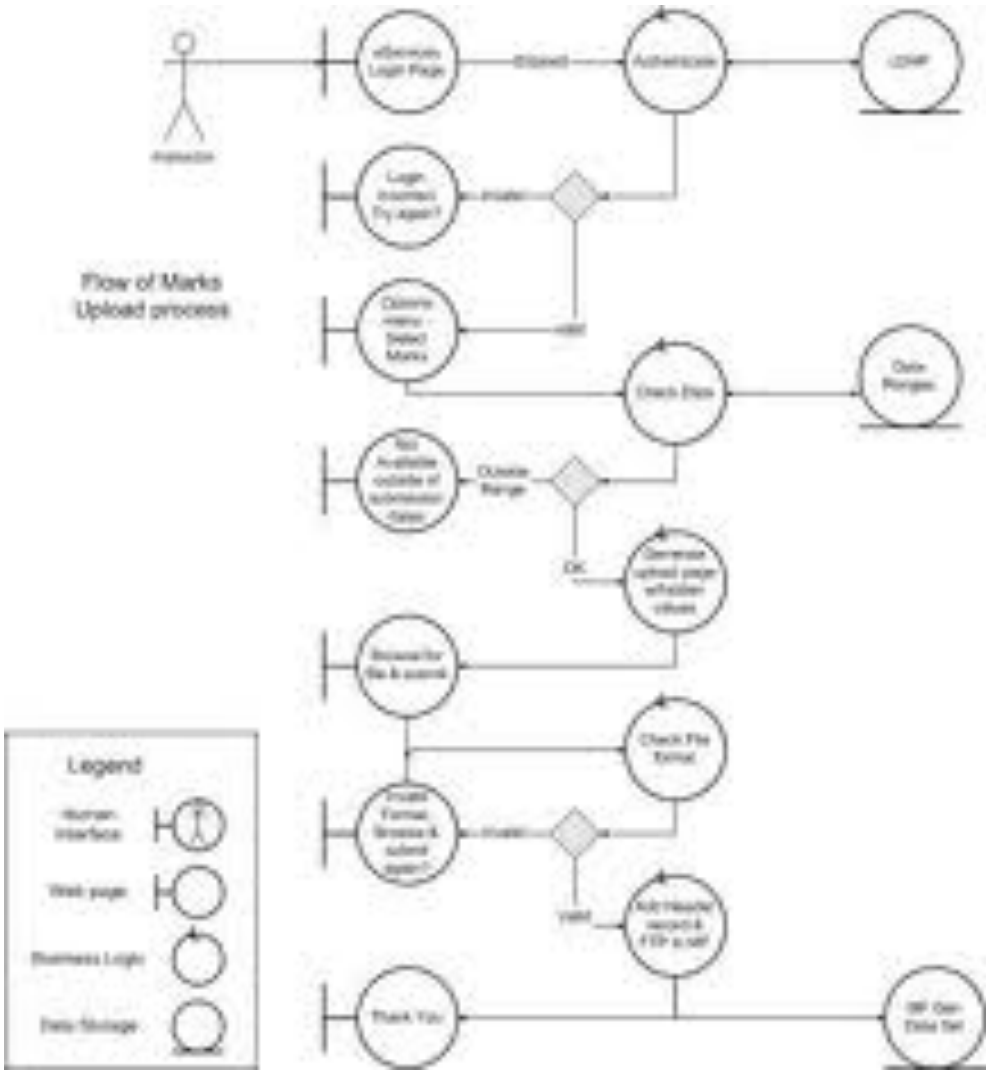


way back when...



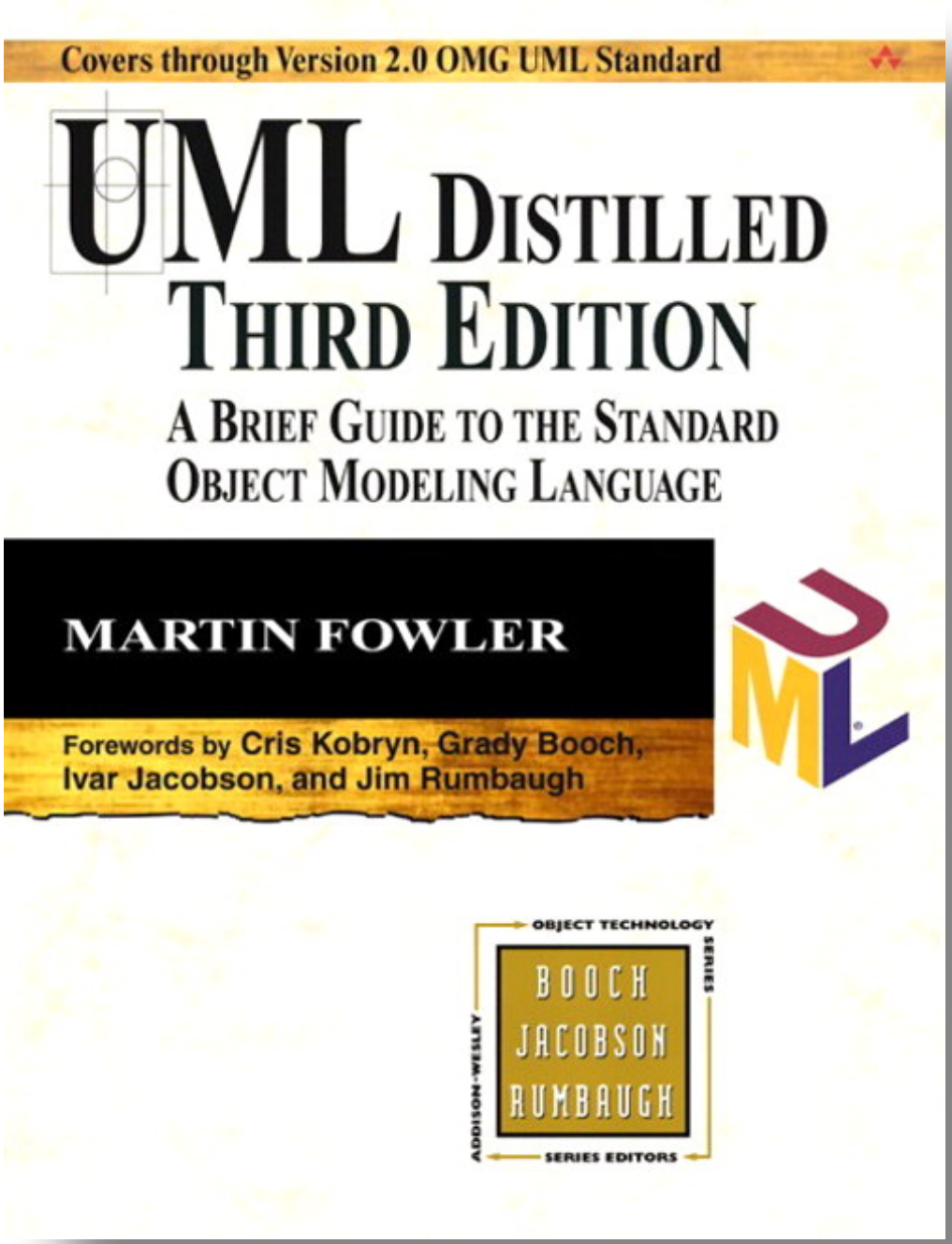
Rumbaugh diagrams

way back when...

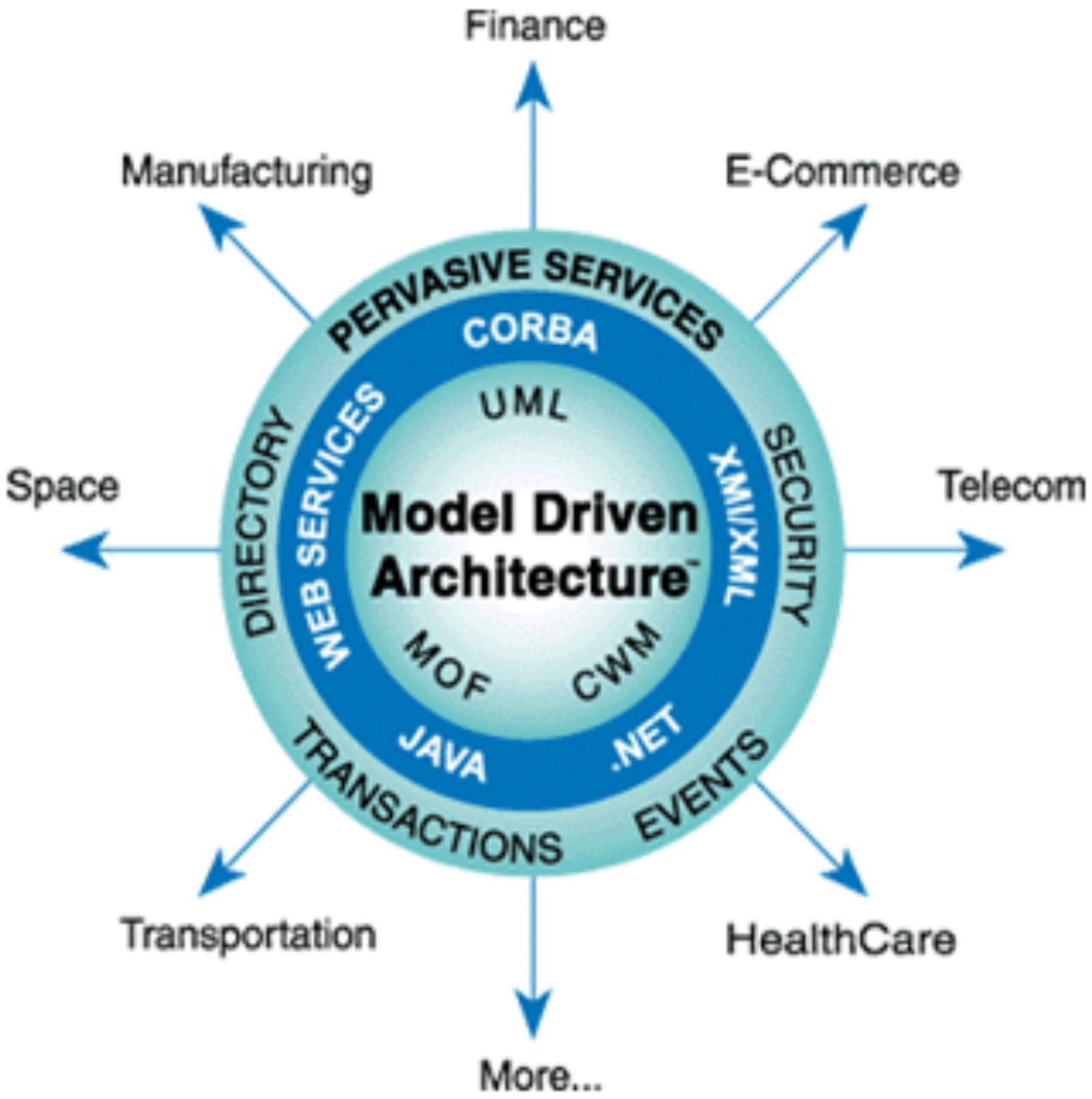


Jacobson use cases

U M L


Covers through Version 2.0 OMG UML Standard
UML DISTILLED
THIRD EDITION
A BRIEF GUIDE TO THE STANDARD
OBJECT MODELING LANGUAGE
MARTIN FOWLER
Forewords by Cris Kobryn, Grady Booch,
Ivar Jacobson, and Jim Rumbaugh
UML
OBJECT TECHNOLOGY SERIES
BOOCH
JACOBSON
RUMBAUGH
ADDISON-WESLEY
SERIES EDITORS

M D A


Finance
Manufacturing
E-Commerce
PERVASIVE SERVICES
CORBA
UML
DIRECTORY
WEB SERVICES
XMI/XML
SECURITY
Space
Model Driven Architecture
Telecom
MOF
CWM
JAVA
.NET
TRANSACTIONS
EVENTS
Transportation
HealthCare
More...


D O A

UML & the Goldilocks problem:

UML is too technical for non-technical people but not technical enough for technical people.

N M L

documentation downsides

infinite regress problem

who is it for?

effort vs benefit

fungibility / malleability

always (potentially) out of date

building an API

Java-doc level documentation...

...at the end of the project

refactoring

emergent design

building an application

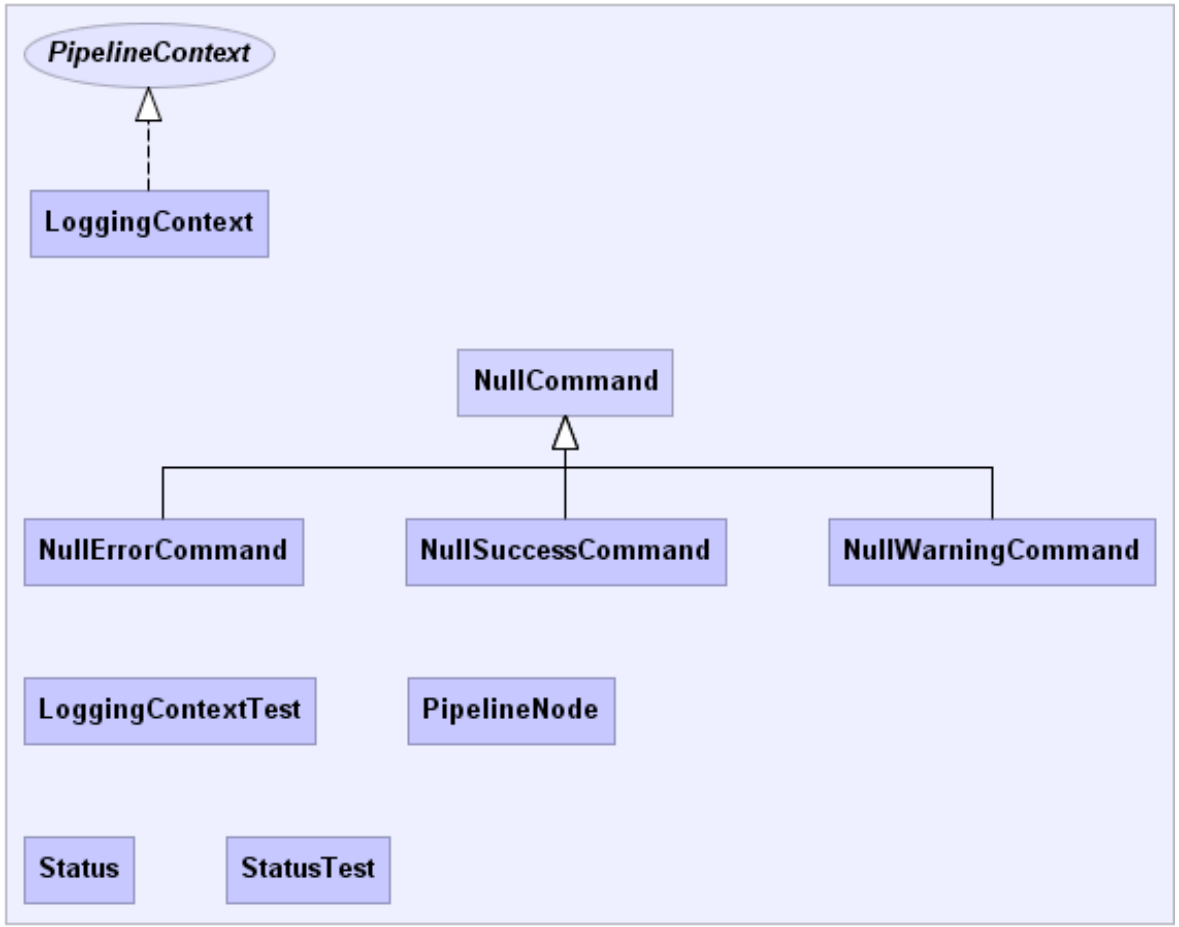
balance effort and reward

what are you documenting?

for what reason?

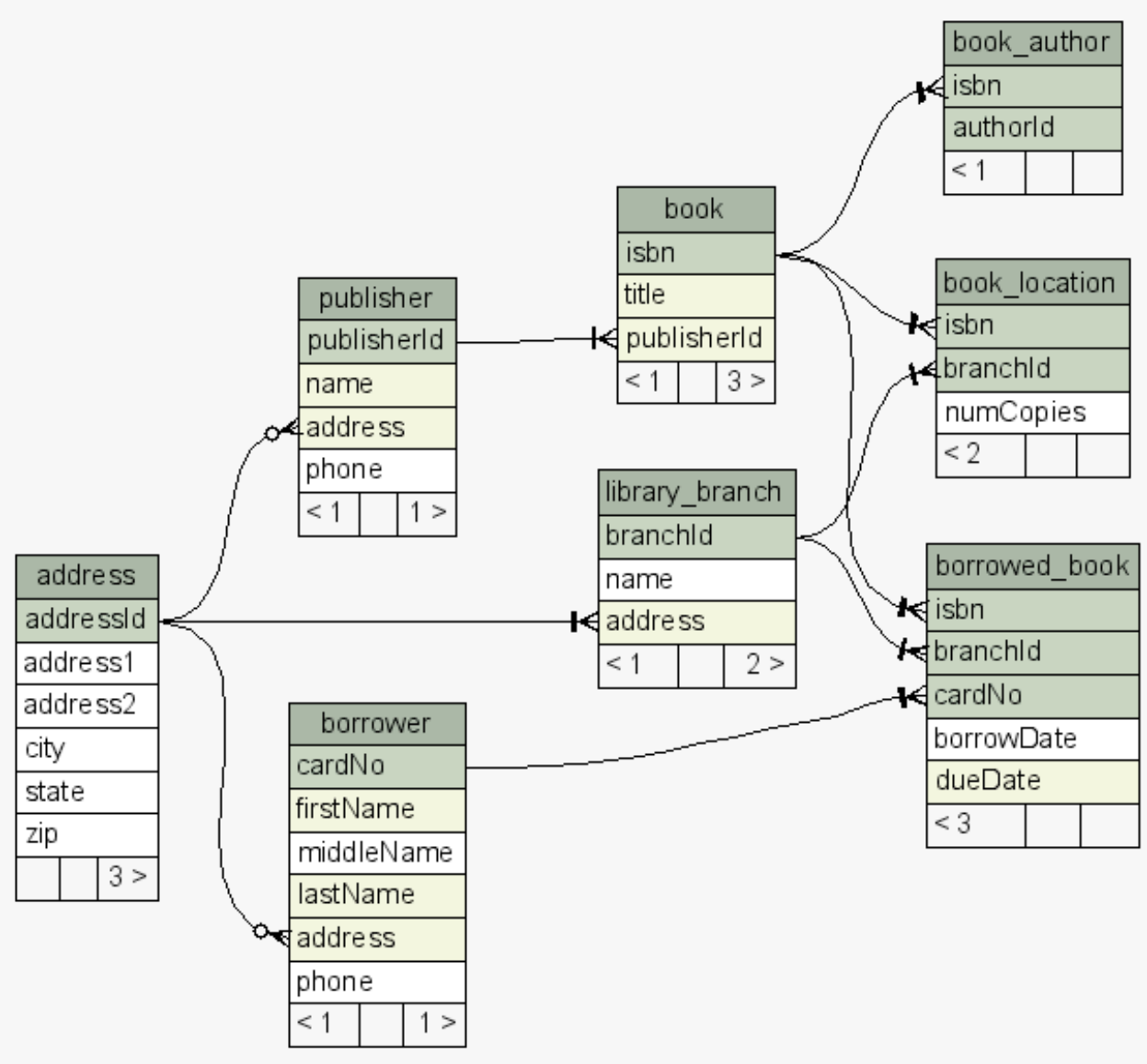
automate instead!

UML diagrams



Generated by yDoc Evaluation Version

database documentation



documentation tools

tools exist for all languages/version control

strike a balance between ceremony & essence

lightweight, in-situ design

capture in-situ design

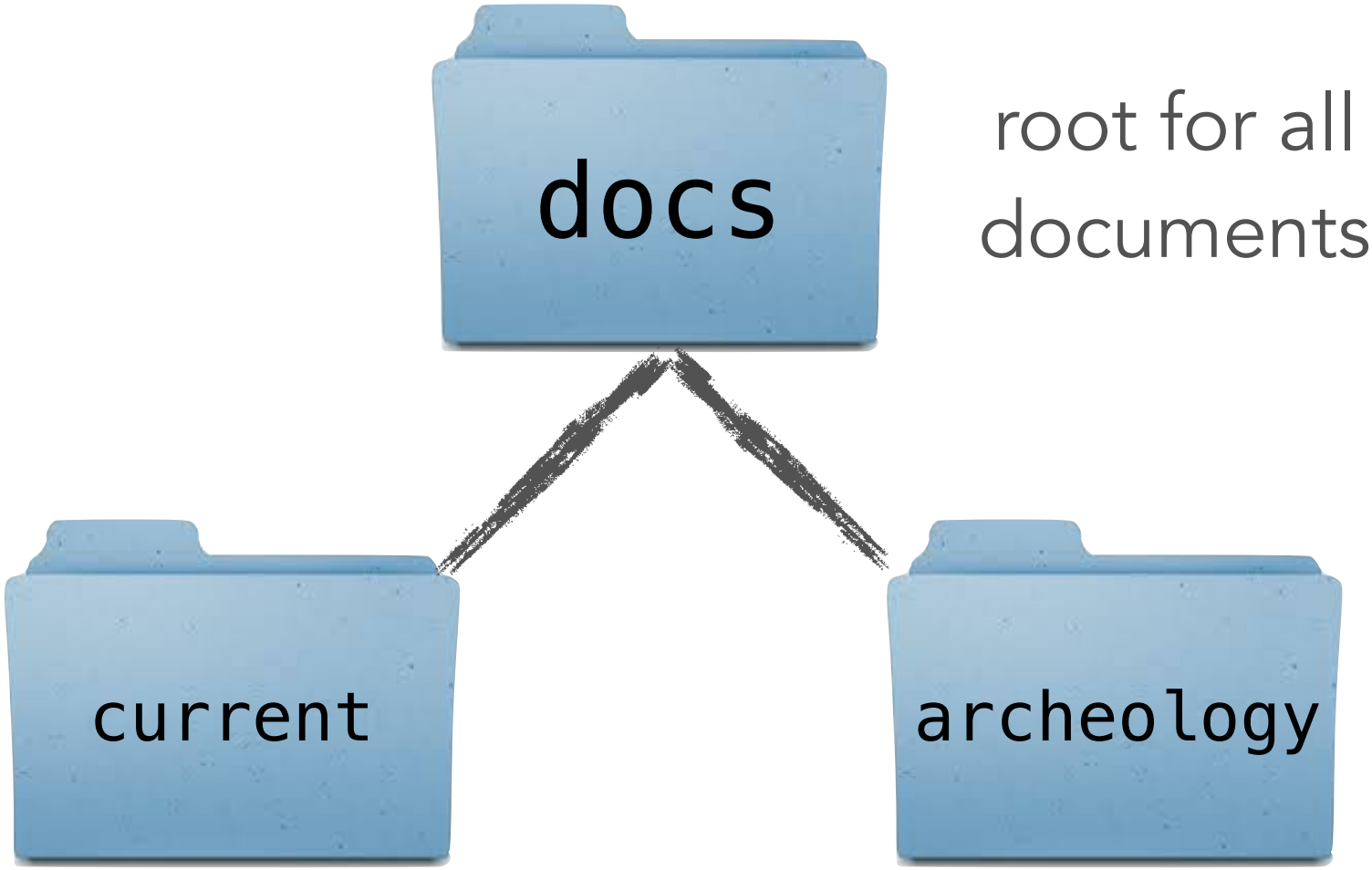
digital camera / smartphone

smart whiteboards

iPad hooked to a projector + note app

hand-crafted diagrams

archeology



root for all documents

useful enough to update

interesting historical artifacts

?’S

Mark Richards
Independent Consultant
Hands-on Enterprise / Integration Architect
Published Author / Conference Speaker

http://www.wmrichards.com
http://www.linkedin.com/pub/mark-richards/0/121/5b9

Published Books:
Java Message Service, 2nd Edition
97 Things Every Software Architect Should Know
Java Transaction Design Strategies

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