IT 4.0 – The Challenge and The Opportunity

Value and Data Driven, Query Centric, Functional, Self Service …

Dave Thomas

www.davethomas.net

Bedarra Research Labs YOW! / GOTO Conferences DepthFirst Carleton University Canada Queensland University of Technology Australia

Lodestone Foundation

IT 4.0 Outline

Challenges

- IT 1.0 - 3.0 Rigidity and Lost Opportunity
- Business 4.0
- Technology 4.0

Opportunities

- Simplfied Flow - Value Driven
- Accelerated Delivery
- Data Driven - Query Oriented
- New Modularity - Microservices and Actors
- Self Service - Ultimate Pairs

Jurassic Software? – Life in The Tar Bits

1.0 Mainframe with COBOL and 4GLs, Waterfall

2. PC and Unix – Desktop GUI, Client–Server, C/C++, Relational and SQL Stored Procedures

3. OO, 3 Tier, Java/C#, Middleware, App Server, Agile, Web 1.0

Escalating Costs of Ownership

Recruiting Retaining Talent

OpenSource Management

Certification vs Competence

Maintenance vs Development

License Complexity Costs

Licenses Expense Complexity

MORE APIs

API Instability

MORE Tools

MORE Accidental Complexity

MORE Languages

MORE SW HW Platforms

Increased Code Bloat

Platform Framework Tool Churn

Business 4.0

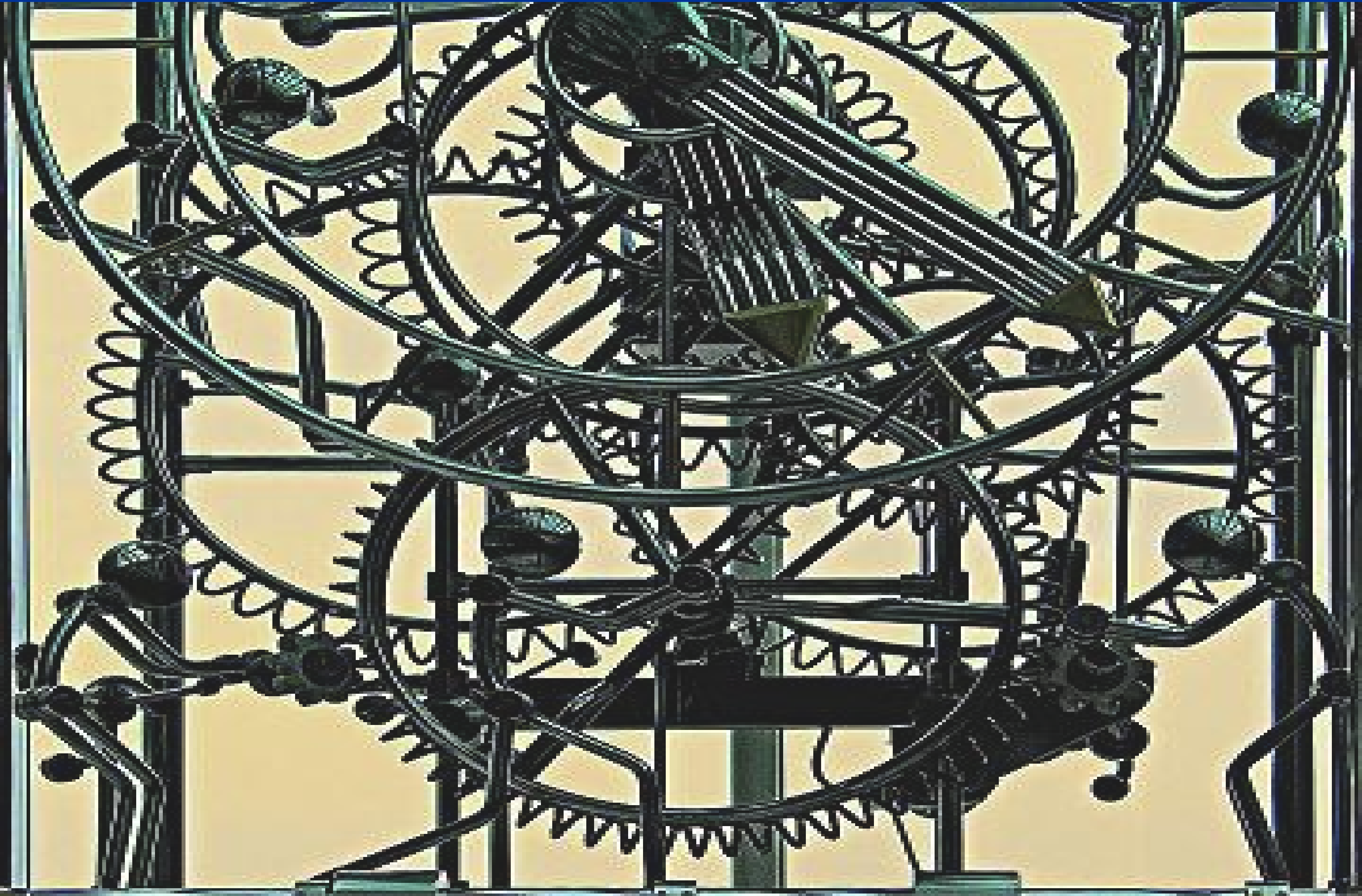
- Innovation delivered through Agility (beyond Rigidity)
- Buiness Model Value Driven
- Data Driven Real-time Business
- Continous Adapation and Delivery

Better, Faster, Cheaper – A New Road?

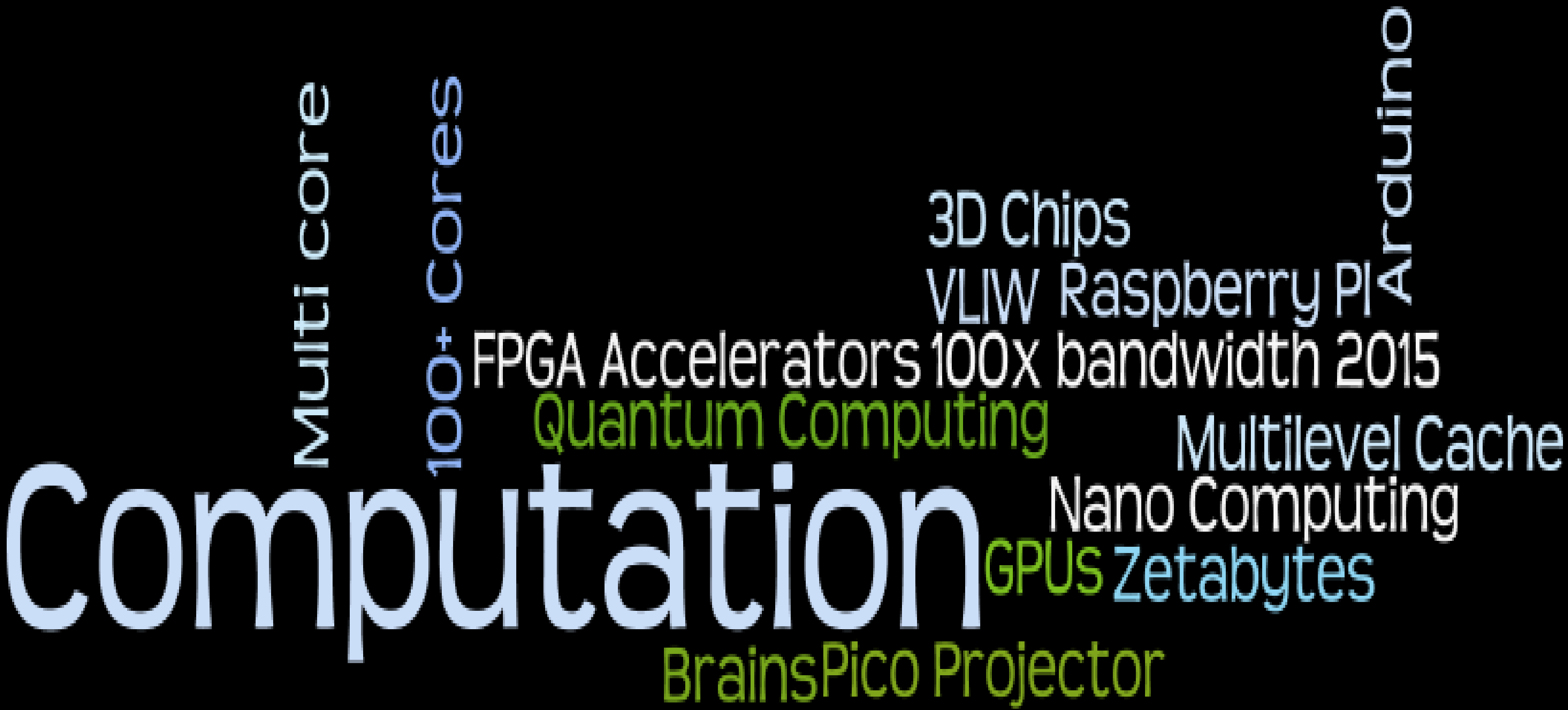
We must improve our way software delivery to meet the challenge.

- Focus on Value and Flow– Target resources and innovations to where they will make a difference.
- Refactor our organization - to enable more concurrency and reduce cycle time without reducing quality. Leverage what works and not be constrained by current best practices. If it is slow it has to go!
- Explore and Innovate – we need to envision alternatives and evaluate them quickly before betting too much on any approach. We need to fail vast to maximize ROI and time.
- Advance and Automate Development – use alternative techniques to communicate, design, estimate, build, test and deploy.

Technology 4.0



Hardware - Warp Speed in Parallel Universes



Let the Hardware Do The Work!

$25,000 buys a computer 1 TB RAM with 40 TB disk and 32 cores!

$200 buys 1000 4 GB cpus on Amazon for 1 hour!

- Automated Build and Test is mandatory
- All interesting data is in memory! DB is an oxymoron
- Inexpensive Data Conversion/Translation
- Data Compression and Encryption is “free” on multi-core
- Speed and Memory enable Simpler Algorithms
- Enable End User Computing at Scale

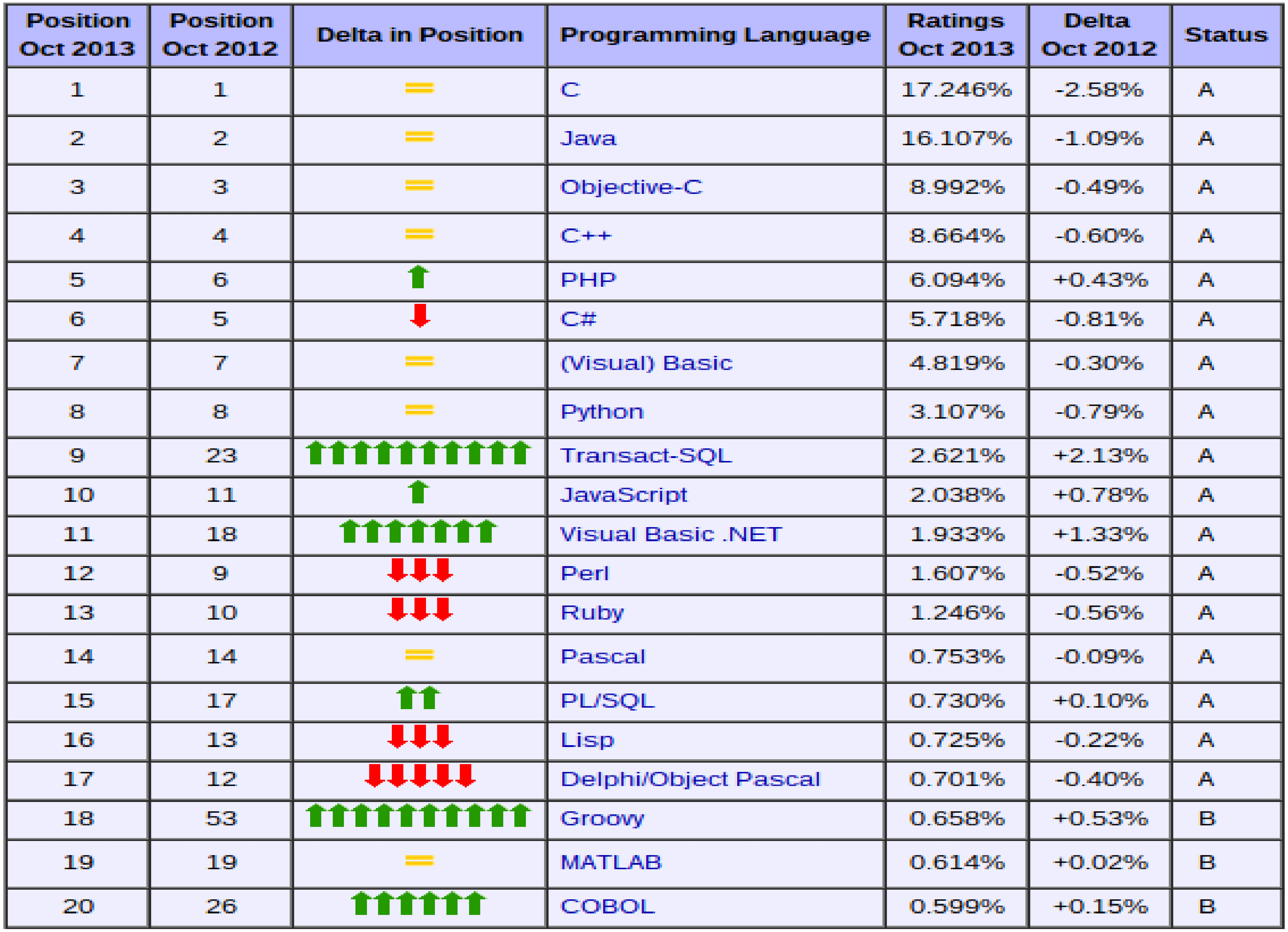
If only the Software will let us have *mechanical sympathy*?

Every Thing Occasionally Disconnected

Mobile
Pen
Web Next
Sencha Touch
Radio
Video
Audio
Native App
Compass
Speech
Web App
Android
Hardware Addons
GPS
NFC
Metro
Occasionally disconnected
Phone Gap
Games
Accelerometer
IOS

Voice
Affordances
Gestures
Walls
Context Aware
Cyborgs
Clothes
Ears
Nano Machines
Goggles
Robots
3D Printing
Smart Things
Sensors
Eyes

So Many Languages to Choose From?



Position Oct 2013	Position Oct 2012	Delta in Position	Programming Language	Ratings Oct 2013	Delta Oct 2012	Status
1	1	=	C	17.246%	-2.58%	A
2	2	=	Java	16.107%	-1.09%	A
3	3	=	Objective-C	8.992%	-0.49%	A
4	4	=	C++	8.664%	-0.60%	A
5	6	⬆	PHP	6.094%	+0.43%	A
6	5	⬇	C#	5.718%	-0.81%	A
7	7	=	(Visual) Basic	4.819%	-0.30%	A
8	8	=	Python	3.107%	-0.79%	A
9	23	⬆⬆⬆⬆⬆⬆⬆⬆⬆⬆	Transact-SQL	2.621%	+2.13%	A
10	11	⬆	JavaScript	2.038%	+0.78%	A
11	18	⬆⬆⬆⬆⬆⬆⬆	Visual Basic .NET	1.933%	+1.33%	A
12	9	⬇⬇⬇	Perl	1.607%	-0.52%	A
13	10	⬇⬇⬇	Ruby	1.246%	-0.56%	A
14	14	=	Pascal	0.753%	-0.09%	A
15	17	⬆⬆	PL/SQL	0.730%	+0.10%	A
16	13	⬇⬇⬇	Lisp	0.725%	-0.22%	A
17	12	⬇⬇⬇⬇⬇	Delphi/Object Pascal	0.701%	-0.40%	A
18	53	⬆⬆⬆⬆⬆⬆⬆⬆⬆⬆	Groovy	0.658%	+0.53%	B
19	19	=	MATLAB	0.614%	+0.02%	B
20	26	⬆⬆⬆⬆⬆⬆	COBOL	0.599%	+0.15%	B

How many Classes & Packages in Java?

Total No of Classes

Java1.02 –> 250 Java1.1 –> 500 Java(2-4) –> 2300 Java5 –> 3500

Java6 –> 3793 Java7 –> 4024 Java8–> ????

Total No of Packages

Java6 –> 203 Java7 –> 209 Java8 -> ???

How many frameworks? ….

How many serialization formats? ...

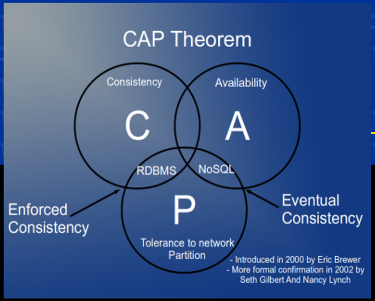
Of Course you know 4.0 Alien Languages !

RACKET
EVENT
ACTORS
JRUBY
PROGRAMMING
JAVA
ERLANG
LANGUAGES
HASKELL
ONCURRENT COLLECTIONS
DART
DECLARATIVE
CLOJURESCRIPT
COFFEESCRIPT
JAVASCRIPT
SPEC
PROCESSING
C++11
CLOJURE AGENTS
SCHEME
IMMUTABLE
VECTOR
ARRAY
LUA
GREMLIN
FUNCTIONAL
RAILS
SCALA
LINQ
LIVELY
DSL
LINQ

Planets Immersed in the Clouds

Elastic Computing
PAAS
SAAS
Chef
Force
Heroku
Accumulo
Fault Tolerance
Private
Replicas
Map Reduce
AppScale
Security
Hypervisor
Trust
OpenStack
Privacy
JSON
App Engine
On Demand
Joyent
App Dynamics
Beanstock
REST
Machine Learning
Protocol Buffer
Walled Garden
Eucalyptus
Glu
Region
Sharding
Cloud Computing
ATOM
Virtualization
Public
RackSpace
Turk
vCloud
IAAS
Big Table
API
Puppet
Azure

The many moons of NoSQL



- Introduced in 2000 by Eric Brewer
- More formal confirmation in 2002 by Seth Gilbert And Nancy Lynch

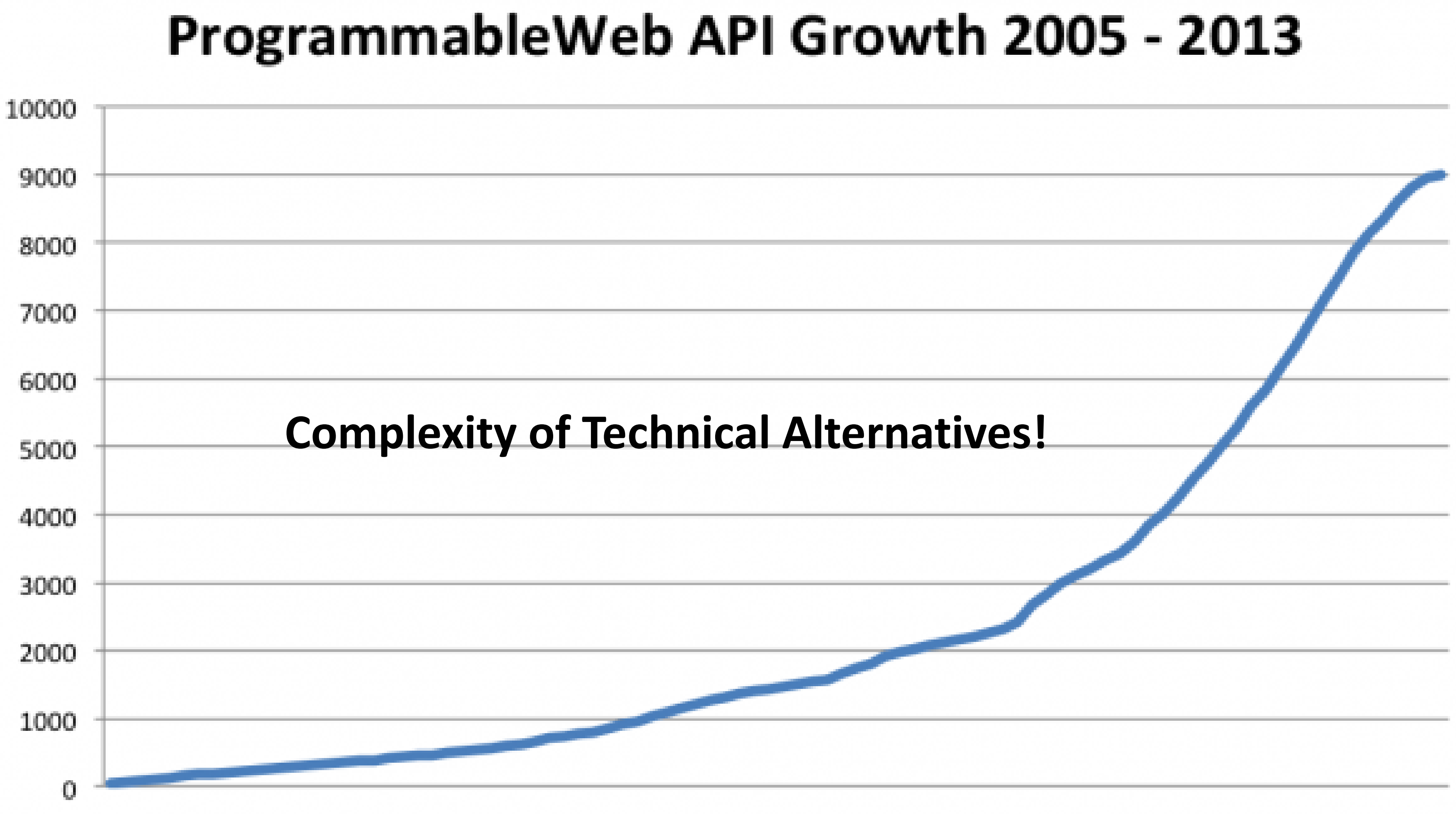
Lost in the Big Data Galaxy

Web Next? Open Force versus Walled App Empires

Atom
TUIO
DoJo
BSON
REST
Raphael
Web
Modules
Reactive
LINQ
Nginx
DOM
SCSS
OWL
JavaScript
Web Workers
Infoviz
RabbitMQ
Backbone.js
Browser Extension
Data Binding
Web Next
Web Sockets
Moustache
Processing.js
Lamp
Apache
508 Compliance
Spring
RDF
Ecmascript Next
Firebug
Linked Data
d3
Local Storage
Rails
Protoviz
WebGL
Node.js
Ember.js
Sinatra
SVG
Canvas
HTML5
CSS
New Relic
HTTP
RSS
Next
Audio
Jquery
JSON
SEO
Yaws
J2EE

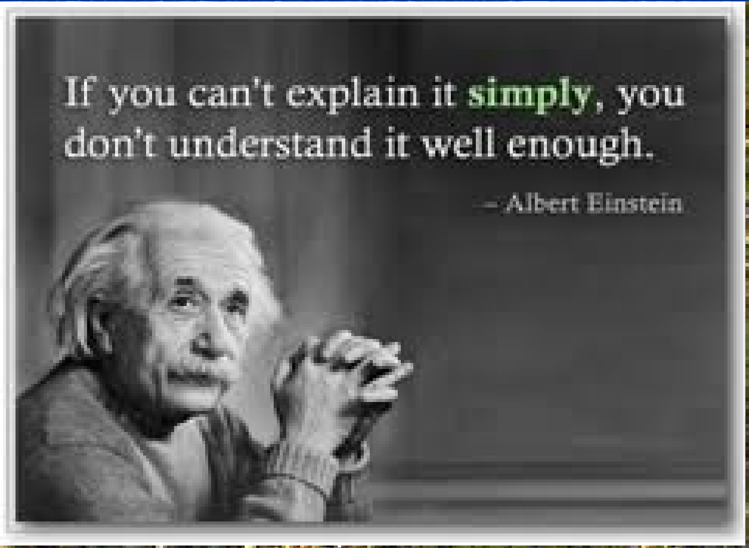
API "Field of Dreams"

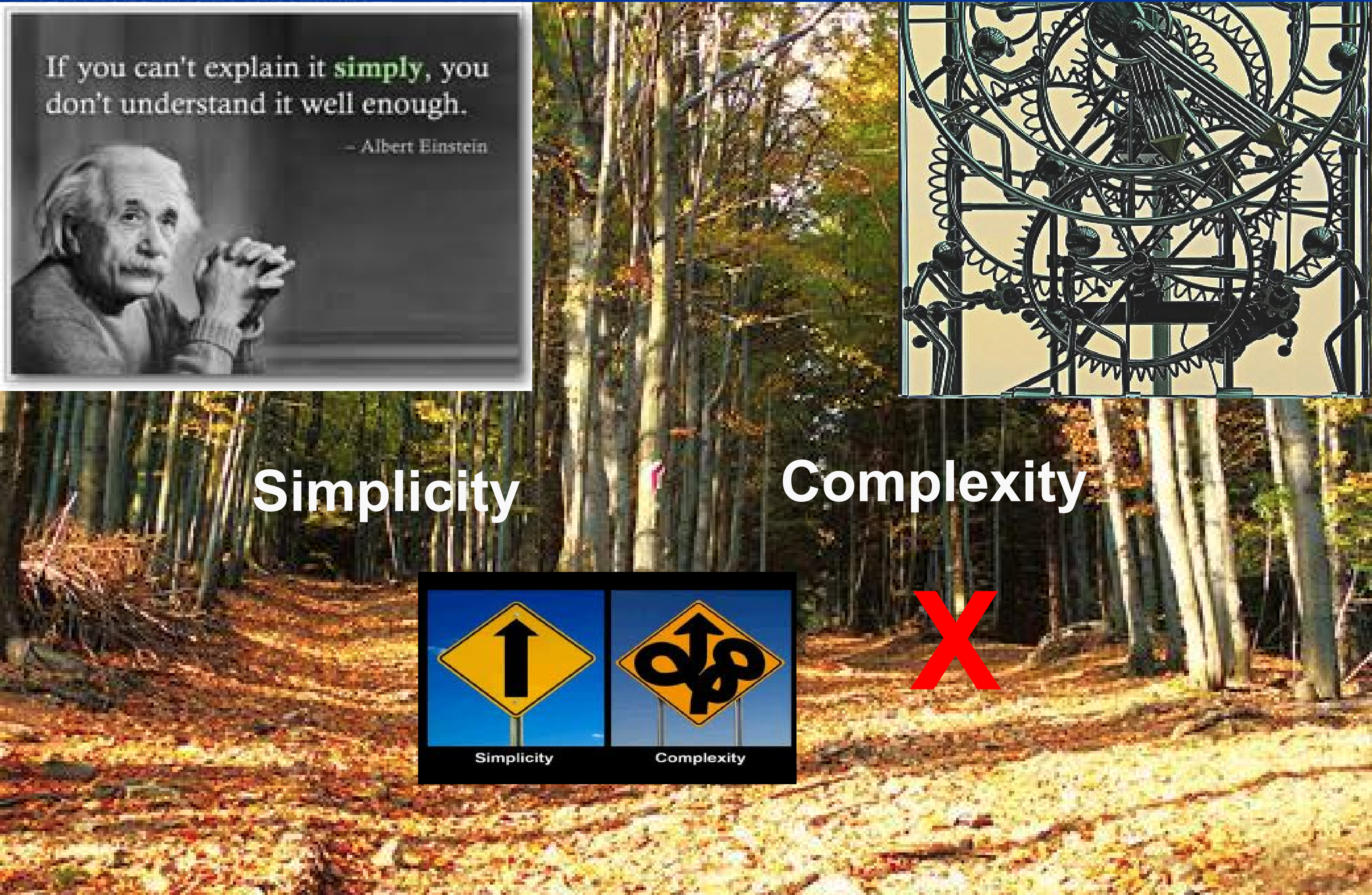
Give them APIs and pray the Applications will



Complexity of Technical Alternatives!

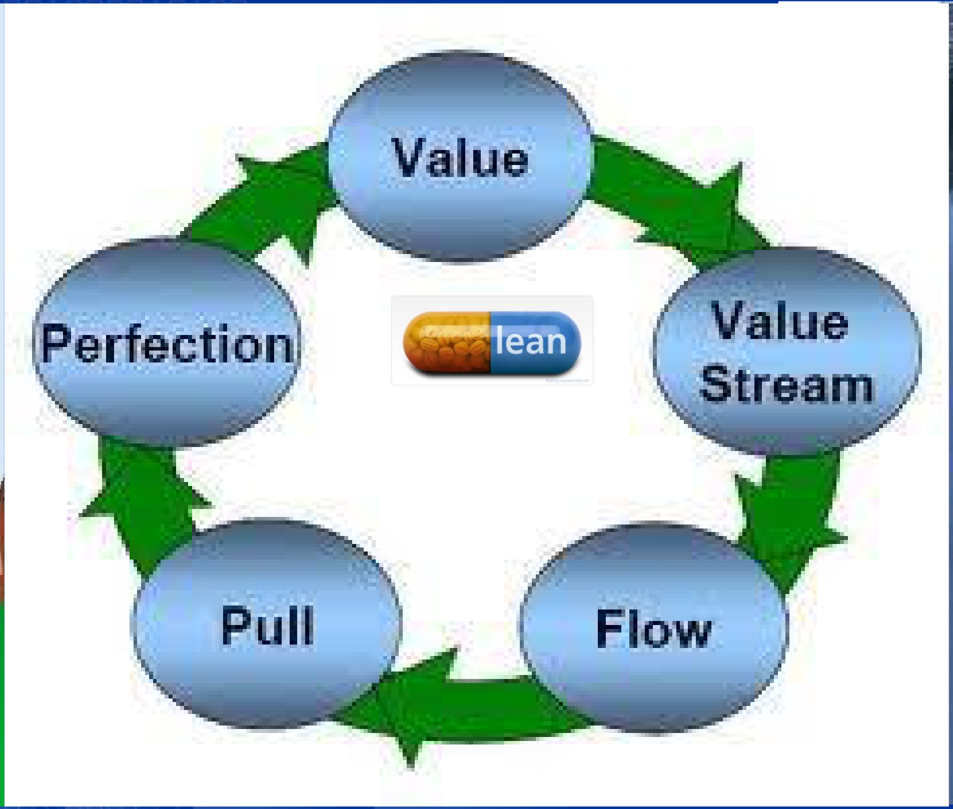
Simplicity! - The Road Not Taken?

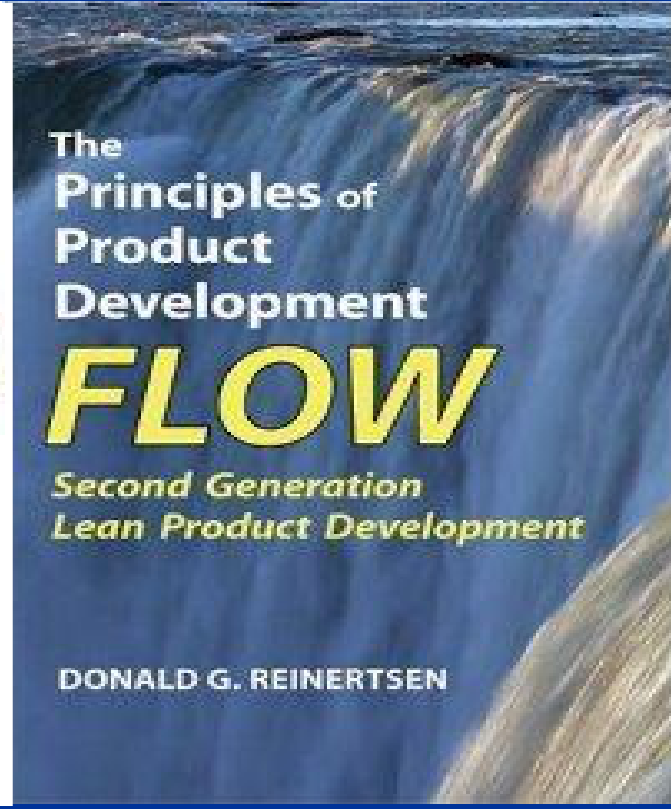

If you can't explain it simply, you don't understand it well enough.
– Albert Einstein



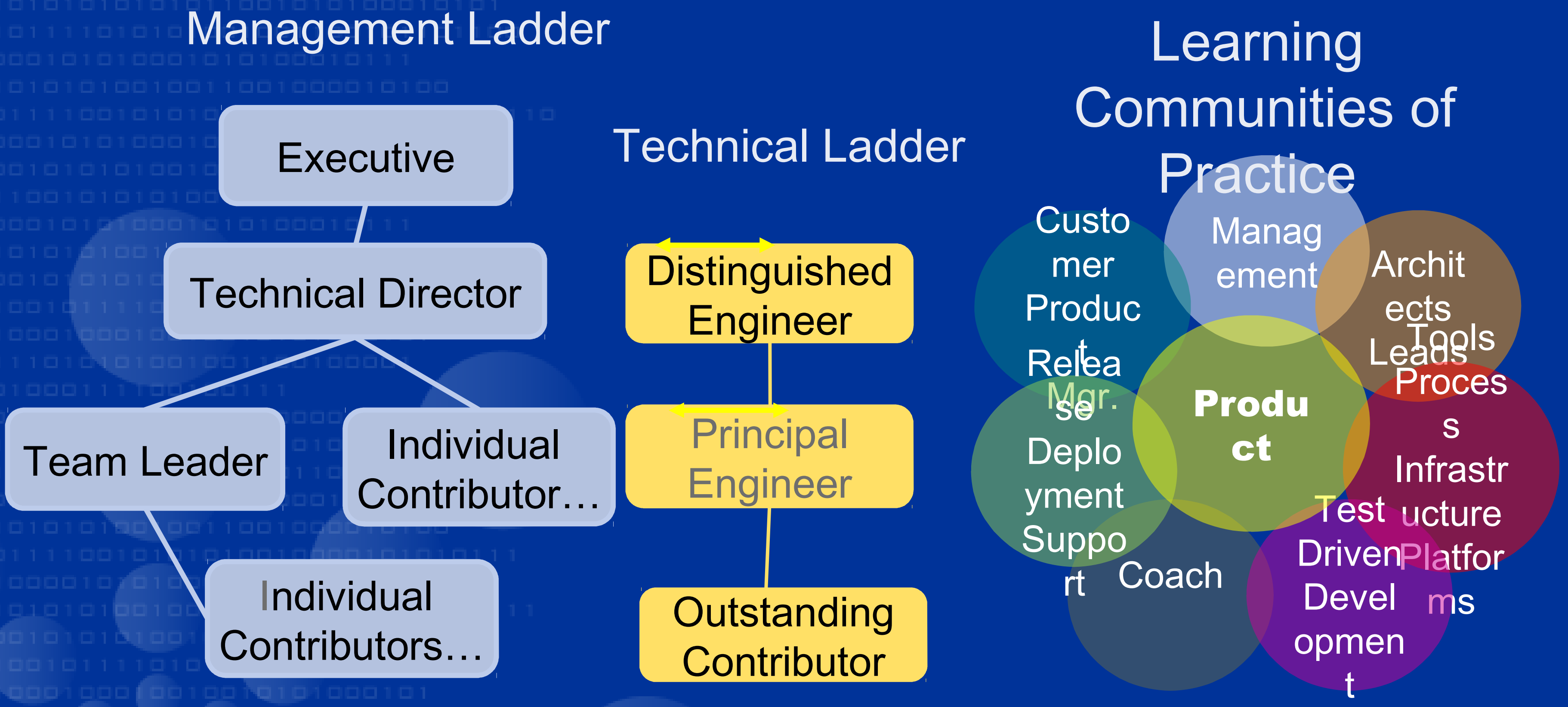
Lean – Ouch! Thinking and Leadership?

Value Driven Flow




The Principles of Product Development
FLOW
Second Generation Lean Product Development
DONALD G. REINERTSEN

Lean Software Organization
Technical Ladders, Playing Coaches and Communities



Align Compensation with Work Products and Goals

Projects are from Mars - Products are from Venus

Projects are about apps and resource; Products are about Features

IT sees feature teams == project teams; Products require component and feature teams;

Projects App is all we need, reuse is optional and unlikely; Product Architecture and Reuse is Essential

Project focus is my App ; Products all about Interfaces, Dependencies

Project apps can evolve : Products need Design to Last

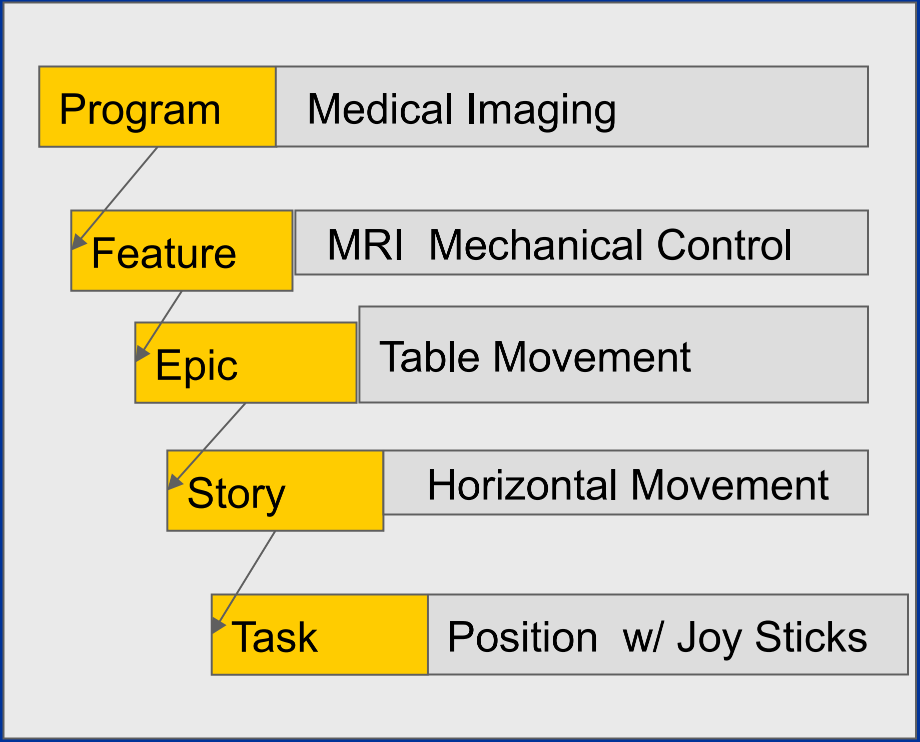
Projects resources pooled ;Products own their $, backlogs, teams …

Projects delivered incrementally ; Products need a schedule and deliverables for major functionality

Eliminate Projects! – Manage to Your Capacity

	Program	Feature	Team
	P1	F1	Blue
		F2	Blue
		F3	Red
		F4	Red
		F5	Red
		F6	Red
	P2	F7	Yellow
		F8	Green
		F9	Green
		F10	Purple
		F11	Purple
	P3	F12	White
		F13	White
		F14	White
		F15	White
	Component	F16	Orange
		F17	Orange
		F18	Orange

Company Backlog	▨
Program Backlogs	⬜🟦▨
Team Backlogs	🟦🟥🟨🟩🟪🟧



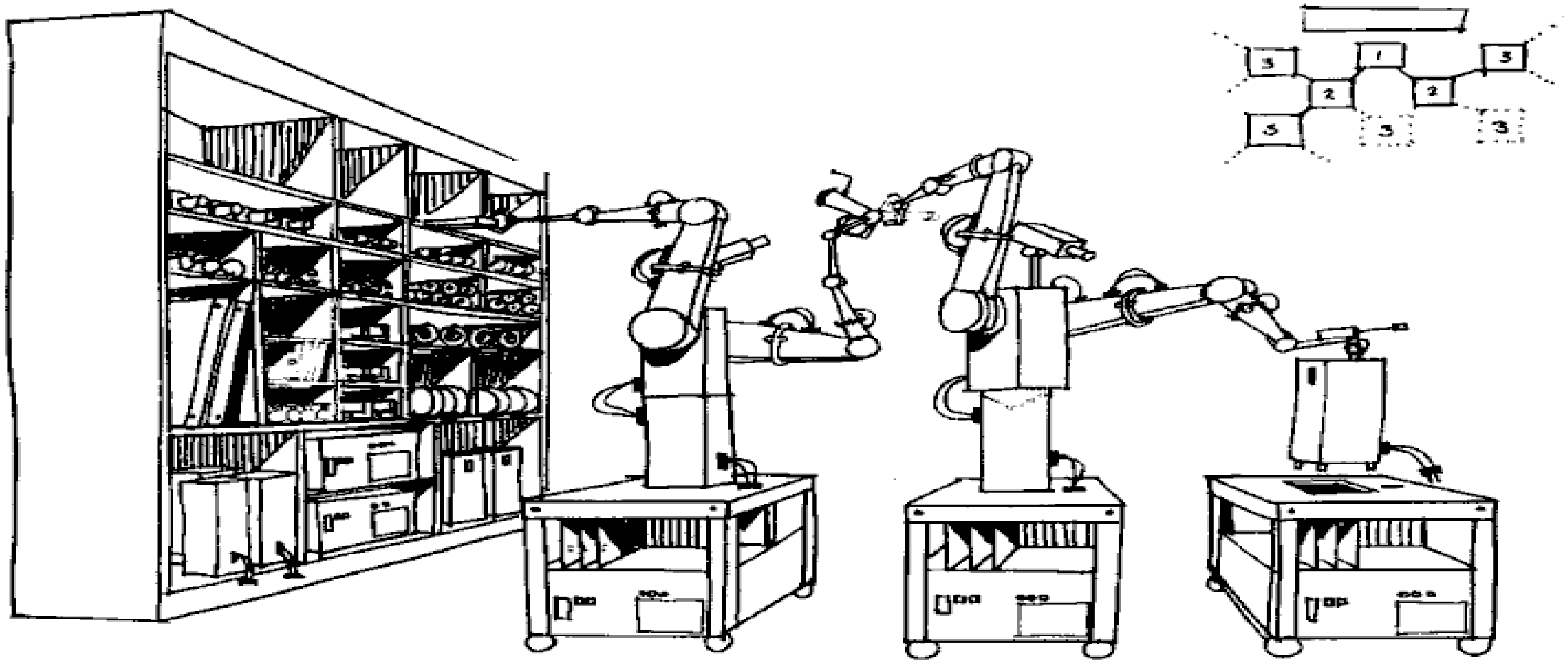
Rapid Application Development Process

Code and Deploy
Kanban
Mocking
Design
Clean Code
Micro Service Architecture
No Build
Refactoring
High Performance Teams
Many Languages
Lean
Reduce Waste
Maximize Flow
Value Chain
Producer
Scrum
Emergent Architecture
Dependency Injection
Software Analytics
Kata
Unit Component Acceptance Test
Lean StartUp
Best Practice
DSM
Cumulative Flow Diagram
DevOps
Continuous Delivery
Envisioning
XP
Acceptance Criteria
One Language
Programmer Anarchy
Single Page App

The More Things Change? … IT Stays the Same!

Create – Read –Update – Delete (CRUD)

Business Analytics, Mobile, Cloud Computing, Multi-player Games …



IT is still! = CRUD + Workflow + Data

Input and Outputs - Forms, Reports and Data Transformations

Workflow – Form Flow and Transformation (Boxes and Arrows)

Event/Actions – Event-State Tables/Sourcing

Rules/Actions – Decision Tables (Rule Engines)

Complex Calculations – Constraints/Dataflow (Spreadsheets)

Data and Relationships – Data Models (ER= OMT)

Objects are in the technology not in the domain!

Data Intensive Computing

All roads lead to some form of Functional CRUD!

Applied Functional Programming (aka Super CRUD)

SQL + Functions + Streams – e.g. Greenplum …

NoSQL Databases – Dictionaries on Steroids (Big Table, CouchDB…)

Map Reduce Data Parallel

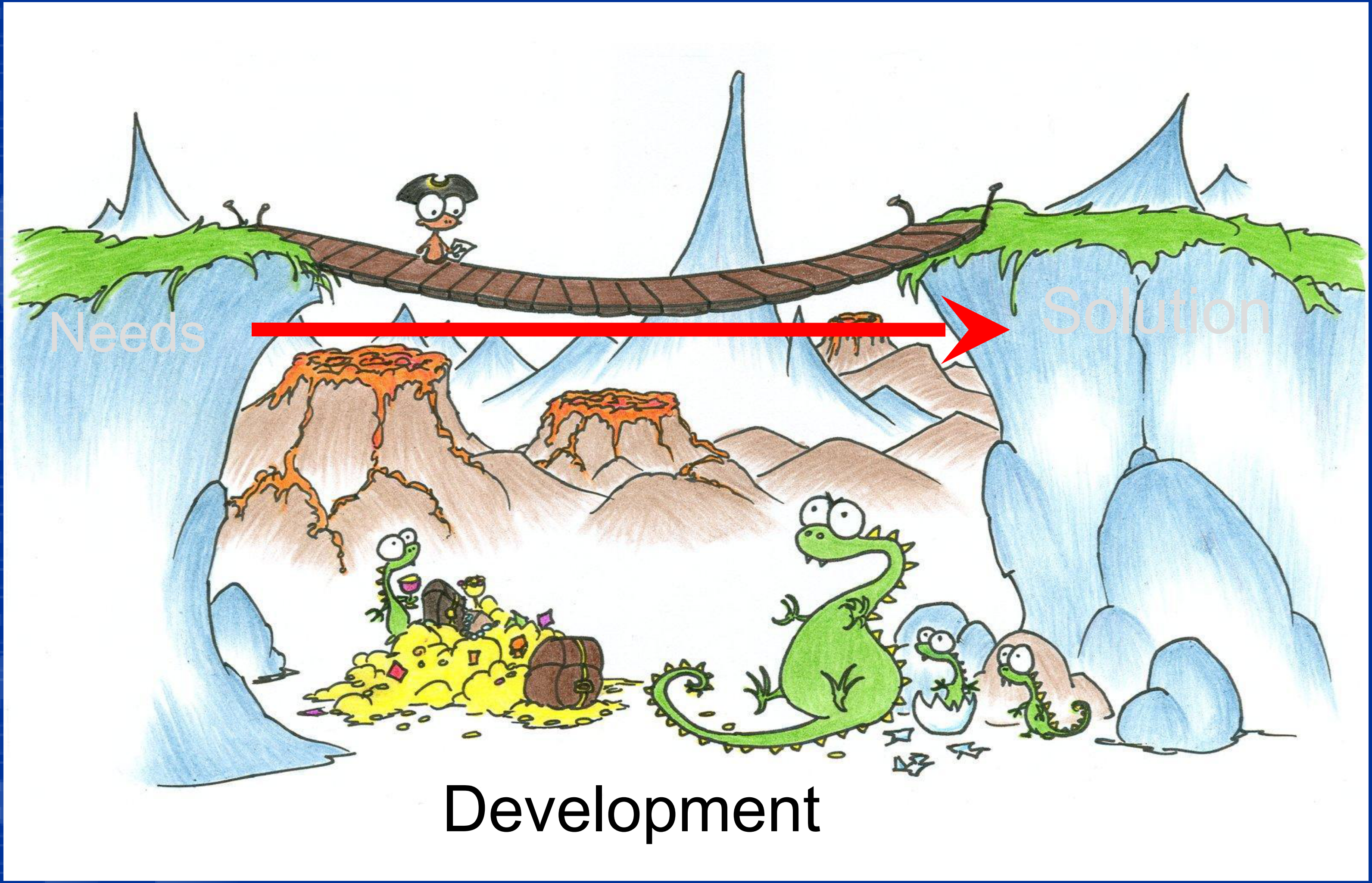
Hybrid JVM, CLR/LINQ functional languages F#, Scala, Clojure

Vector Functional Programming

Reactive Programming Rx...

Faster, Better, Cheaper

How can we reduce the software cycle time?



Answer - Ship Less Code! Make it Easier to Change!

KLOCS (1000 lines of code) Kill! => Be More Expressive

Dependencies Strangle => Micro Service Aarchitecture

Avoid Frameworks inject dependencies => Less Objects, more FP?

Data Driven Always More Flexible Than Code Driven => Descriptions, Tables and Queries

Automate Everything!

Enable DIY Programming => Let the business think and compute

Automate Everything!

- Simpfied Requirements Capture - Delta analysis
- Automated Testing
- Automated Build
- Automated Deploy
- Automated Test Construction?
  - - Randomized Testing
- Automated Program Construction ?
  - Programming By Example
  - Machine Learning

Use Less Objects and Less Code !

Object Refactoring harder than Changing Data and Functions

Many Apps have few if any domain objects!

Table Driven Programming

Rules	Decision Table
Calculation	Spreadsheet
Data Validation	Domain and Range Table
Mapping	Lookup Table
Flow	Data, Work Flow, Message
Events, Matches	State Table
Process, Reports	Input-Output Table
Acceptance Criteria	BDD
Domain Models	Entity-Attribute Dictionary

Table Oriented Programming

A picture is 1000 words, a table 200 and a diagram 50

Advantages

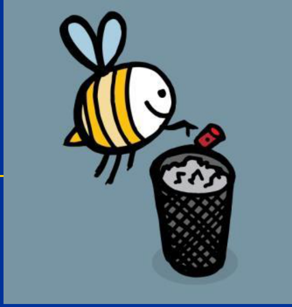
- Easily understood by Business, BA, Dev and QA
- Easy to create, refactor and extend using Excel
- Modularity through structured tables
- Consistency /Completeness Checking
- Easy to version and Diff
- Efficient Automated Data Driven implementation
- Data Driven means changes can be “hot deployed” to a running application

Applications

Insurance, Banking, Taxation, Healthcare, ATC, Real-time…

TOP Programmers Wanted!

Simplify - Reduce Integration Time and $$$

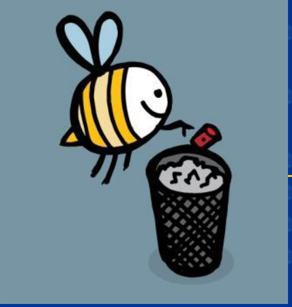


- ATOM/RSS feeds on our legacy/partner systems – journal files, events …

- Use ODBC as a simple interface to complex server systems

- REST and JSONify your services, Provide a scriptable service, Use Self Described Data e.g. LinkedData

- Use a simple MashUp tool to deliver a integrated application view



Script to Save Time and $$$

- More and more applications are *disposable* at least in
  - make it work and get it out there,
  - scale it later if you need to
- Script Softly for productivity
  - Ruby, Python, PHP, Groovy, JavaScript, Clojure…
- Leverage cloud services (map reduce, cloud DB..CRM)
- Leverage core internal and external services via REST/ODBC

Service-Oriented Computing Infrastructure: Cloud -The Software Enabler

The Emergence of A Simpler Application Infrastructure

- Examples - On Demand, Software As A Service such as Amazon S3, EC2, SimpleDB, Google App Engine, Sales Force …
- Simpler limited "thin" service API (< 50) closer to underlying platform which provides support for scalable, distributed, secure computing
- Independence on mainstream vendor Underware and Middleware
- Google Linux, VMware Virtual Machine, MS Azure Hypervisor V

Application Development Benefits

- Small Service API (thin to none class library & frameworks)
- Limited Choice Reduces Decisions and Support
- Leverages Other Apps through Services
- Total App Responsibility from envisioning to production i.e. App Team caries the beeper

Code and Deploy – Testing Considered Harmful!

We all know that testing costs a lot and takes time, mocking is hard especially when working in a changing complex environment. Lets not bother!

What it takes

1. Modular micro service architecture
   - instant deployment and tear down
   - loose data coupling
   - well defined SLA
2. Simple Functionality in each Deployment
3. Stringent SLA Monitoring for Deviant
4. Let it fail architecture (Erlang versus execptions)
5. Replication

Applications – Telecom, Finance, eCommerce ...

What Is a Micro Service?

A service provide an interface to a specific subset of functionality/data in an enterprise.

- Versioned Services can be concurrently deployed to enable new and older apps.
- Services are not frameworks; they are components with value only interfaces.
- Services are realized as components with consumable APIs.
- One or more services can often be supported by a service team.

Technical Value Proposition

Services increase modularity, reduce coupling, increase technology and delivery choices.

- Services reduce large monolithic applications to a set of single function technology independent APIs which can be composed into business applications
- Services are loosely coupled hence can be incrementally be developed and deployed
- Services are easier to distribute, provision and monitor
- Services expose the tangible business architecture versus the internal technical applications architecture
- Services enable parallel Business App development and service definition and development (feature and component teams)
- Services are realized as components with consumable APIs.
- Services easier to deploy and test.
- One or more services and be owned and implemented by a team.

Enabling Loose Coupling

- All APIs are value based and where possible stateless
- Isolation of services in separate processes/machines
- Simple Pipes and Filters when possible
- RSS/ATOM feeds from events/updates/logs
- Occasionally Disconnected – replication and sync; event source..
- Simple efficient implementations using co-routines..
- Orchestration/Composition using Scripting Process
  - Messaging.. Node.js, Erlang, Actors
- FP thinking encourages value orientation and composition

Service Architecture and Design

Occasionally-Disconnected
Micro-Services
Interfaces Self-Described
ServceBus
Fault-Tolerance REST JSON Nodejs
Separation-of-concerns
Service-Test RSS Data-driven Component Erlang Lock-Free
Functioal-Web SLA
Actors
Pico-container
QuickCheck
Isolation
Radomzed-Testing Data-Flow DevOps Orchestration Scripting
Pipes Code-Deploy-Monitor
Table-driven Let-It-fail PBE
Messaging Eventual-Consistency
Reactive SBE Replication ODBC Protocols
Integration-patterns
Strong-generic Corourines
Flow-Oriented API
SOLID-principles
Provisioning

Micro Serivce - The Business Value Proposition

Business Apps can be more easily configured from Services

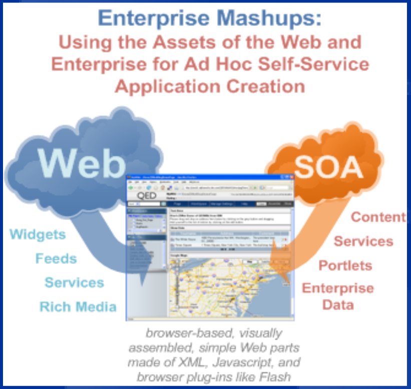
- Large Monolithic Systems are decomposed into simpler services which enable
- Services and their SLA published in a web catalog
- One or more services can be composed into an App with a tailored interface for a given market
- App delivery can be supported by simpler less technical end user tooling such as Wikis, Visual Connections
- Apps can combine services from insurance and banking etc.
- Bus Apps = Services (Features) + Services (Components)

DIY - Bus App Development

Do It Ourselves Programming –The Empowerment

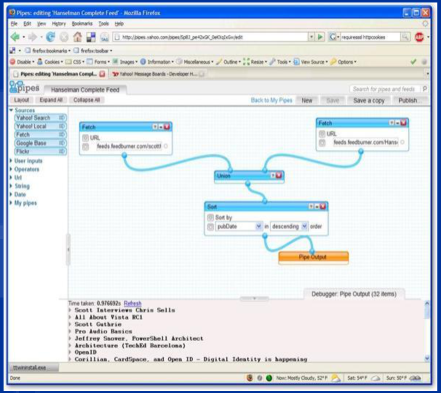
Business Driven Development

- Enterprise Mashups – The Real SOA?
- Applications Assembled from Feeds and Services



Enterprise Mashups:
Using the Assets of the Web and Enterprise for Ad Hoc Self-Service Application Creation
Web
SOA
Widgets
Feeds
Services
Rich Media
Content
Services
Portlets
Enterprise Data
browser-based, visually assembled, simple Web parts made of XML, Javascript, and browser plug-ins like Flash

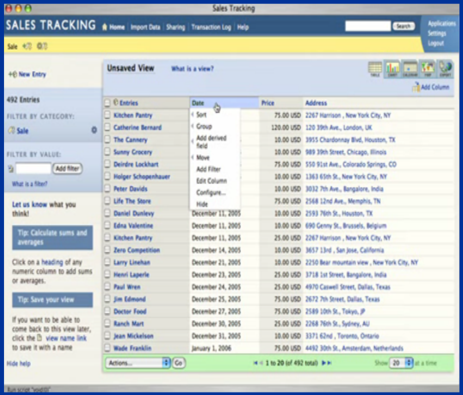
IBM QEDWiki



```
Time taken: 0.976692s Refresh
Scott Interviews Chris Sells
All About Vista RC1
Scott Guthrie
Pro Audio Basics
Jeffrey Snover, PowerShell Architect
Architecture (TechEd Barcelona)
OpenID
Corillian, CardSpace, and Open ID – Digital Identity is happening
```

Yahoo Pipes

Google Mashup Editor

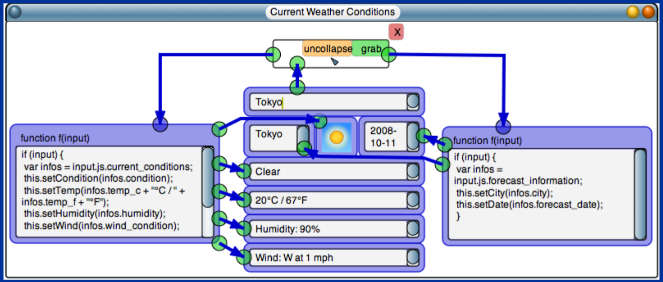


Click on a heading of any numeric column to add sums or averages.

If you want to be able to come back to this view later, click the view name link to save it with a name

Entries	Date	Price	Address
Kitchen Pantry		75.00 USD	2267 Harrison , New York City, NY
Catherine Bernard		120.00 USD	120 39th Ave., London, UK
The Cannery		10.00 USD	3955 Chardonnay Blvd, Houston, TX
Sunny Grocery		10.00 USD	989 39th Street, Chicago, Illinois
Deirdre Lockhart		75.00 USD	550 91st Ave., Colorado Springs, CO
Holger Schopenhauer		10.00 USD	1363 65th St., New York City, NY
Peter Davids		10.00 USD	3032 7th Ave., Bangalore, India
Life The Store		75.00 USD	2568 12nd Ave., Memphis, TN
Daniel Dunlevy	December 11, 2005	10.00 USD	2593 76th St., Houston, TX
Edna Valentine	December 11, 2005	10.00 USD	690 Genny St., Brussels, Belgium
Kitchen Pantry	December 11, 2005	25.00 USD	2267 Harrison , New York City, NY
Zero Competition	December 14, 2005	10.00 USD	3657 13rd , San Jose, California
Larry Linehan	December 21, 2005	10.00 USD	2250 Bear mountain view , New York City, NY
Henri Laperle	December 23, 2005	25.00 USD	3718 1st Street, Bangalore, India
Paul Wren	December 24, 2005	25.00 USD	4970 Caswell Street, Dallas, Texas
Jim Edmond	December 25, 2005	75.00 USD	2672 7th Street, Dallas, Texas
Doctor Food	December 27, 2005	75.00 USD	2589 10th St., Tokyo, JP
Ranch Mart	December 30, 2005	25.00 USD	2268 76th St., Sydney, AU
Jean Mickelson	December 31, 2005	10.00 USD	3371 62nd , Toronto, Ontario
Wade Franklin	January 1, 2006	75.00 USD	4492 30th St., Amsterdam, Netherlands

Dabble DB



```
function f(input)
if (input) {
 var infos = input.js.current_conditions;
 this.setCondition(infos.condition);
 this.setTemp(infos.temp_c + "°C / " +
infos.temp_f + "°F");
 this.setHumidity(infos.humidity);
 this.setWind(infos.wind_condition);
```

```
function f(input)
if (input) {
 var infos =
input.js.forecast_information;
 this.setCity(infos.city);
 this.setDate(infos.forecast_date);
 }
```

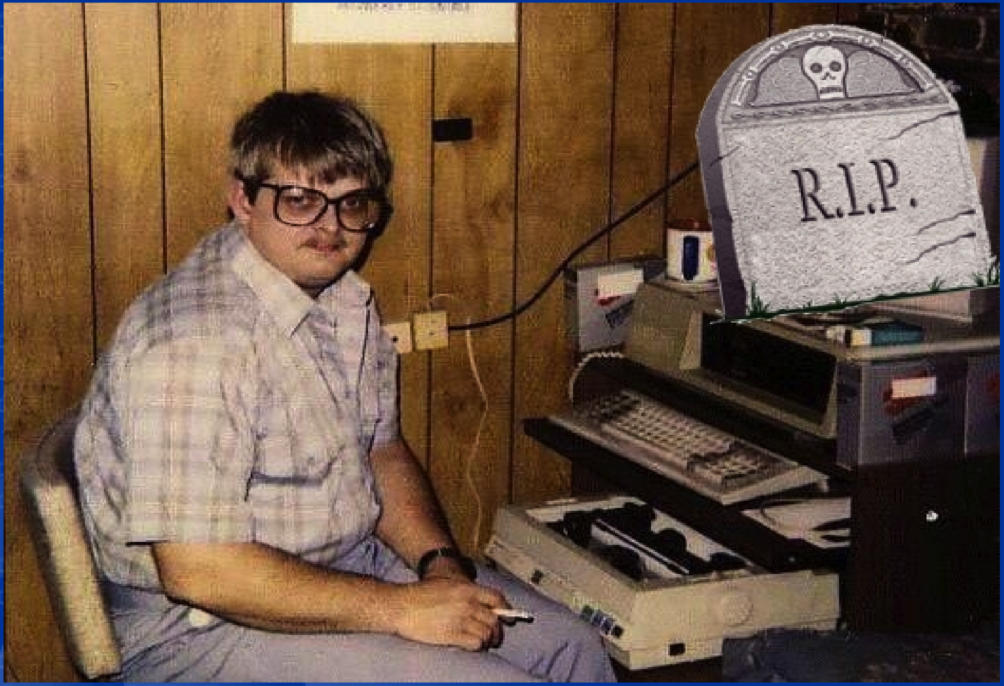
Lively Fabrik

Loose Coupling – Let's Hope It Sticks This Time

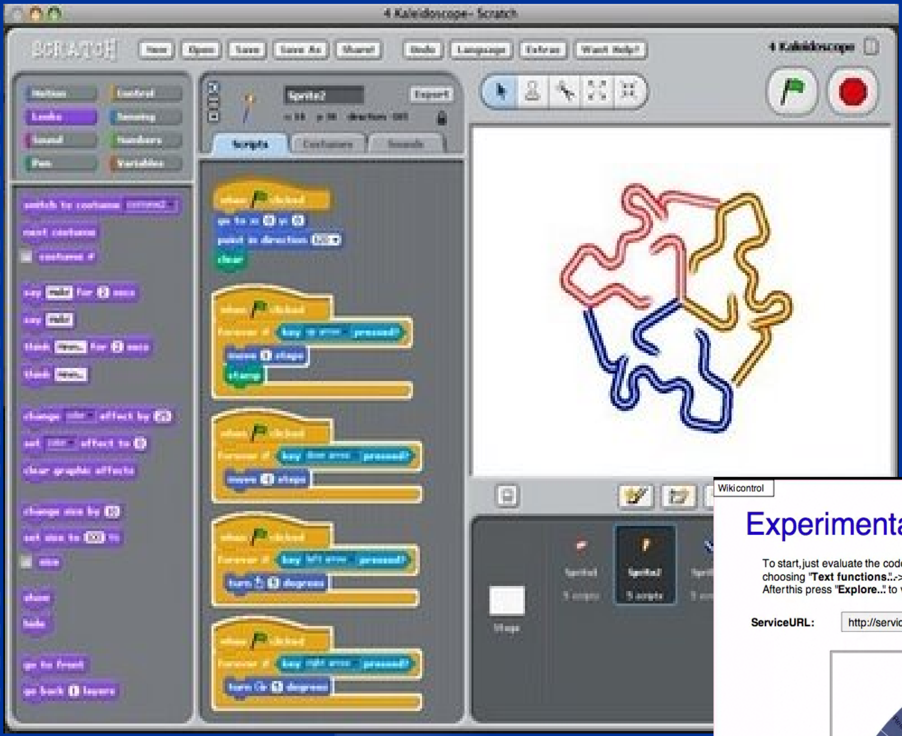
- Data Flow – Data Flow Computing … Maxor FPGA DF
- Structured Analysis and Design (SADT)
- Unix Pipes and Filters
- Flow Base Programming – J. Paul Morrison
- Hewitt Actors
- Spreadsheets
- Visual Programming - Labview
- Actors - Erlang
- Query/Collection Oriented Programming

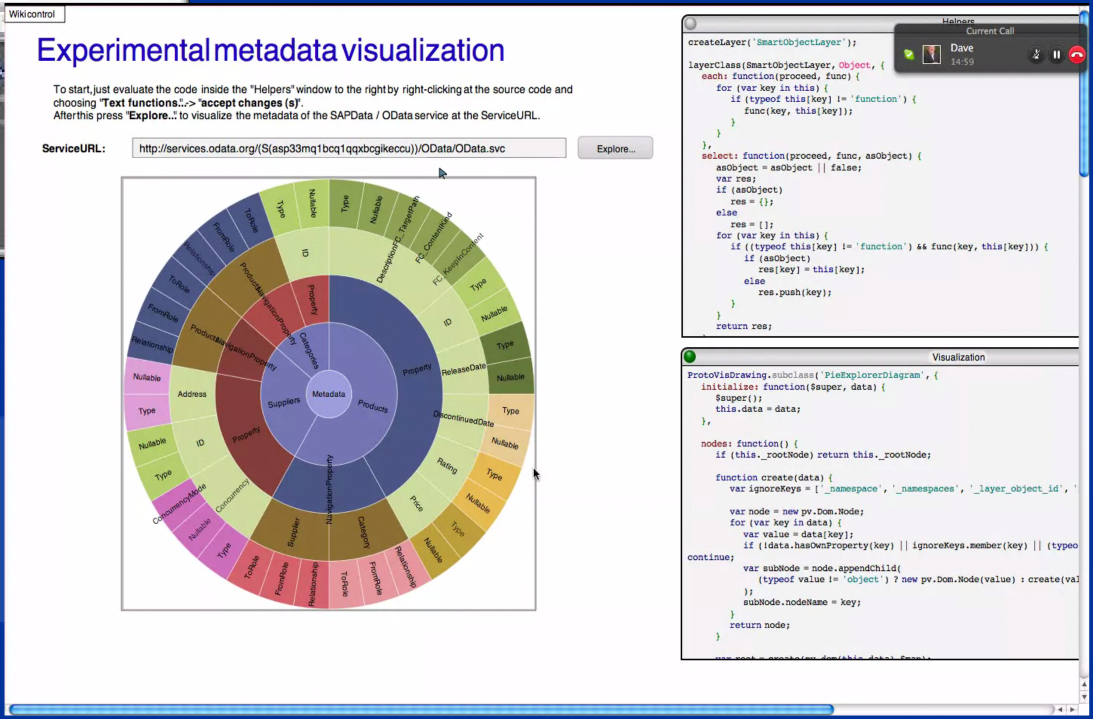
202x?

- Happy end users with DIY computing
- Best Practice = Lean + Real Models to Code By Business



We can learn from our Children!





Experimental metadata visualization

To start, just evaluate the code inside the "Helpers" window to the right by right-clicking at the source code and choosing "**Text functions...**-> "**accept changes (s)**".
After this press "**Explore...**" to visualize the metadata of the SAPData / OData service at the ServiceURL.

ServiceURL: http://services.odata.org/(S(asp33mq1bcq1qqxbcgikeccu))/OData/OData.svc

```
createLayer('SmartObjectLayer');

layerClass(SmartObjectLayer, Object, {
   each: function(proceed, func) {
      for (var key in this) {
         if (typeof this[key] != 'function') {
            func(key, this[key]);
         }
      }
   },
   select: function(proceed, func, asObject) {
      asObject = asObject || false;
      var res;
      if (asObject)
         res = {};
      else
         res = [];
      for (var key in this) {
         if ((typeof this[key] != 'function') && func(key, this[key])) {
            if (asObject)
               res[key] = this[key];
            else
               res.push(key);
         }
      }
      return res;
```

```
ProtoVisDrawing.subclass('PieExplorerDiagram', {
   initialize: function($super, data) {
      $super();
      this.data = data;
   },

   nodes: function() {
      if (this._rootNode) return this._rootNode;

      function create(data) {
         var ignoreKeys = ['_namespace', '_namespaces', '_layer_object_id', '

         var node = new pv.Dom.Node;
         for (var key in data) {
            var value = data[key];
            if (!data.hasOwnProperty(key) || ignoreKeys.member(key) || (typeo
continue;
            var subNode = node.appendChild(
               (typeof value != 'object') ? new pv.Dom.Node(value) : create(val
            );
            subNode.nodeName = key;
         }
         return node;
      }
```

Спасибо!

"The future has already arrived. It's just not evenly distributed yet."

William Gibson

Summary

- Focus on continuous delivery of value
- Maximize Flow
- Leadership and Skills Matter
- Favor targeted high value change over systemic change
- Build Products not Projects
- Respect the Individual and Organizational APIs
- Just Enough Design and Architecture
- Features and Components both essential
- Ensure every feature has an associated acceptance criteria
- Acceptance Tests >> API Test >> Unit Test
- Automate everything
- Use the right tool/practice for the right job