

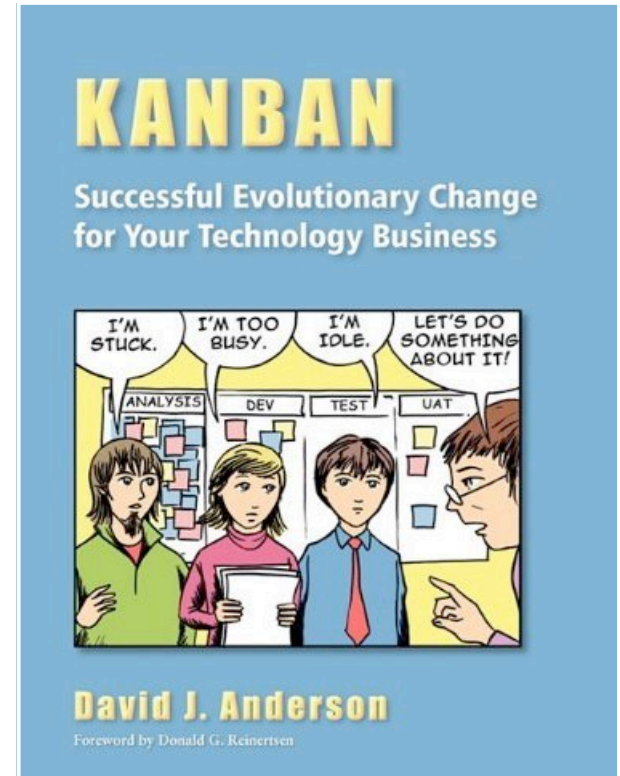
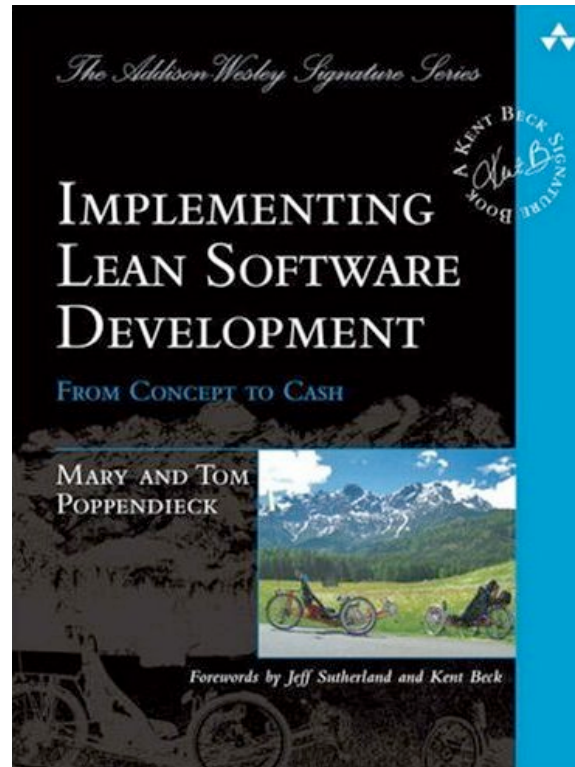
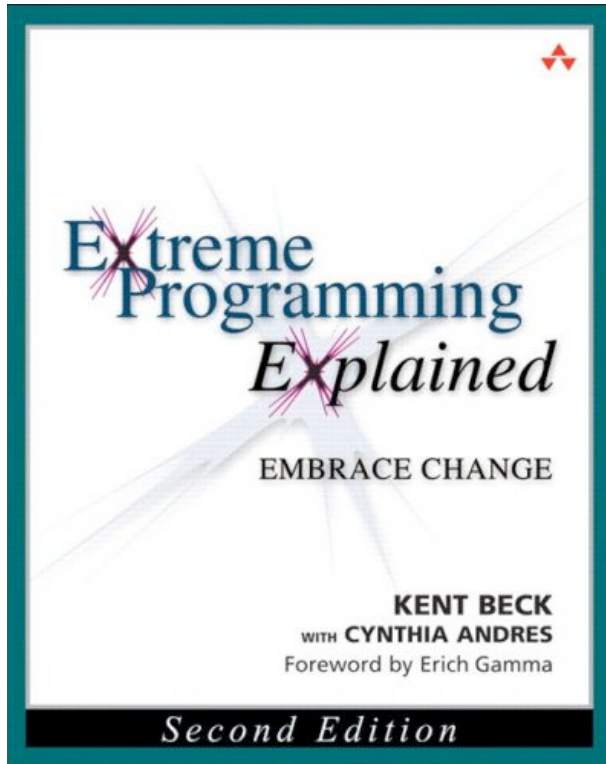
November, 2011  
Joshua Kerievsky  
Industrial Logic, Inc.  
Twitter: @JoshuaKerievsky

# Lean Startup

Why It Rocks Far More Than Agile Development



**#LeanStartup**



# **The Four Steps to the Epiphany**

***Successful Strategies for  
Products that Win***



**Steven Gary Blank**

Customer  
Development

Extreme  
Programming



Lean

Kanban

A disciplined, scientific  
and capital efficient  
method for discovering  
and building products and  
services that people love.



Mitch Kapor  
Founder, Lotus  
Development  
Corp.

“This is the book  
whose lessons I want  
**every entrepreneur**  
**to absorb and apply.**  
I know of no better  
guide to **improve the**  
**odds** of a startup's  
success.”

“Don't let the word  
startup in the title  
confuse you. This is  
a **cookbook for**  
**entrepreneurs in**  
**organizations of all**  
**sizes.”**



Roy Bahat  
President, IGN  
Entertainment





Scott Cook  
Co-Founder,  
Intuit

“Eric reveals the **rigorous process** that trumps luck in the invention of new products and new businesses. We've made this a centerpiece of how teams work in my company... it works!”

NeedFeed helps people  
make better buying decisions  
using their social network.

---

[ABOUT](#) | [WORKING AT NEEDFEED](#) | [JOBS](#) | [FAQ](#)

---

**Request Beta Invite**

Enter your email to be added to our Beta invite list.

Request

# Happiness?

Build

Hypothesis

Plan

Experiment



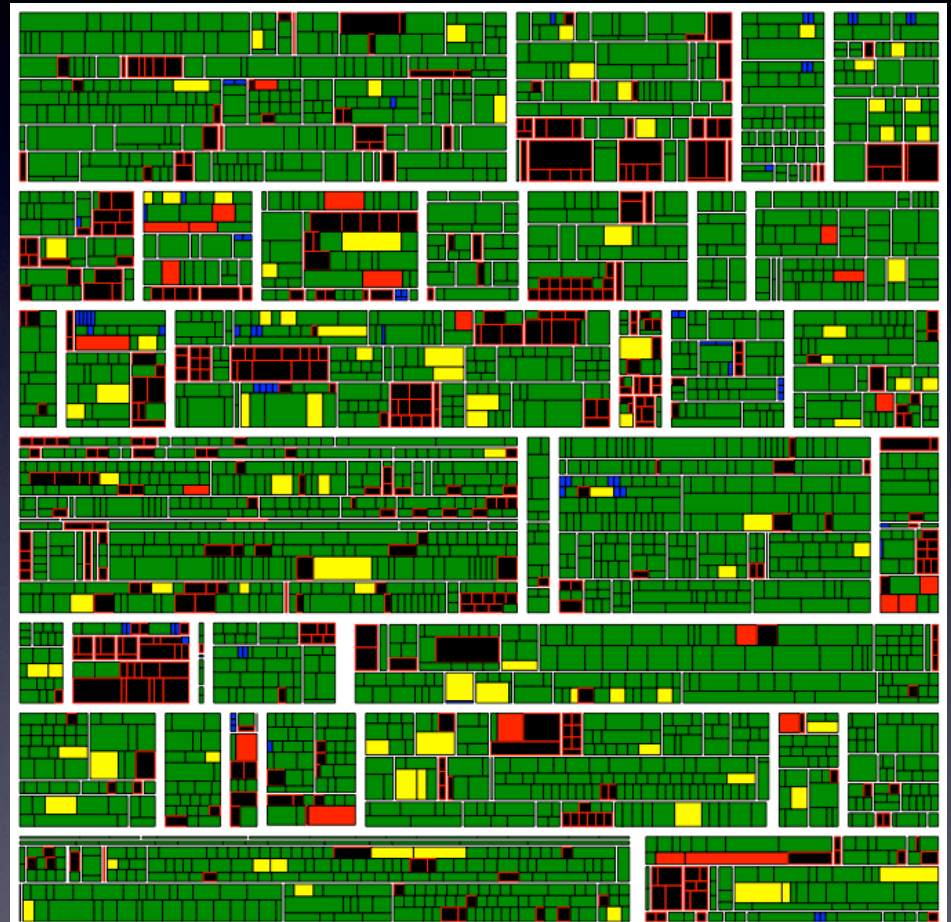
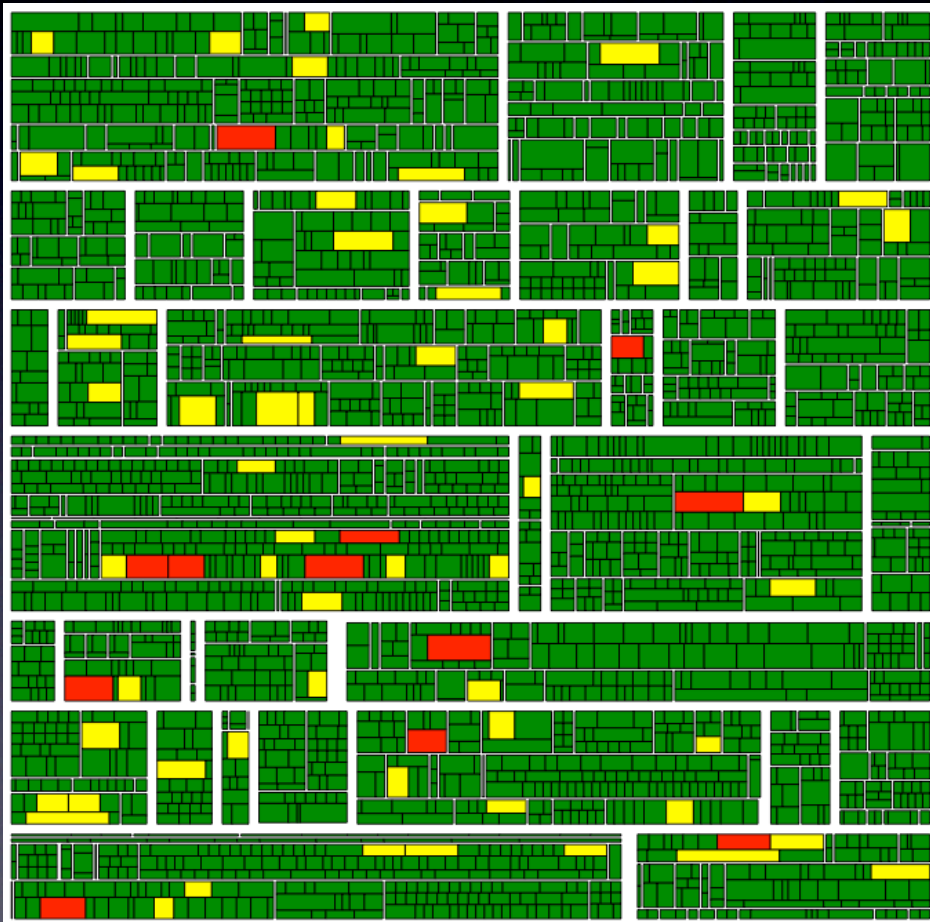
“Learning to see waste and then systematically eliminate it has allowed lean companies such as Toyota to dominate entire industries.”

Eric Ries

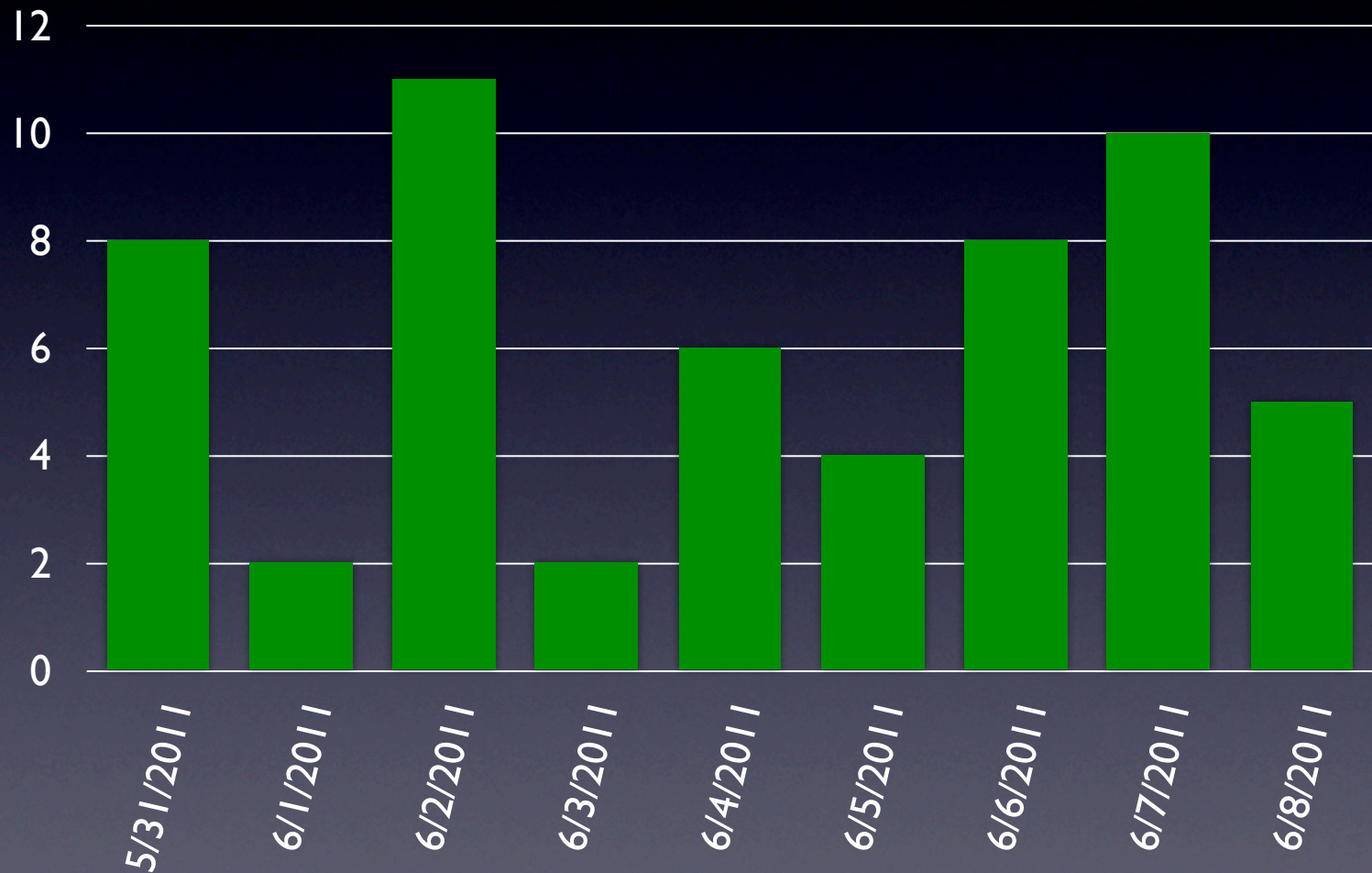
“Agile methods remove waste...yet those [Agile] methods had led me down a road in which the majority of my team’s efforts were wasted.”

- Eric Ries

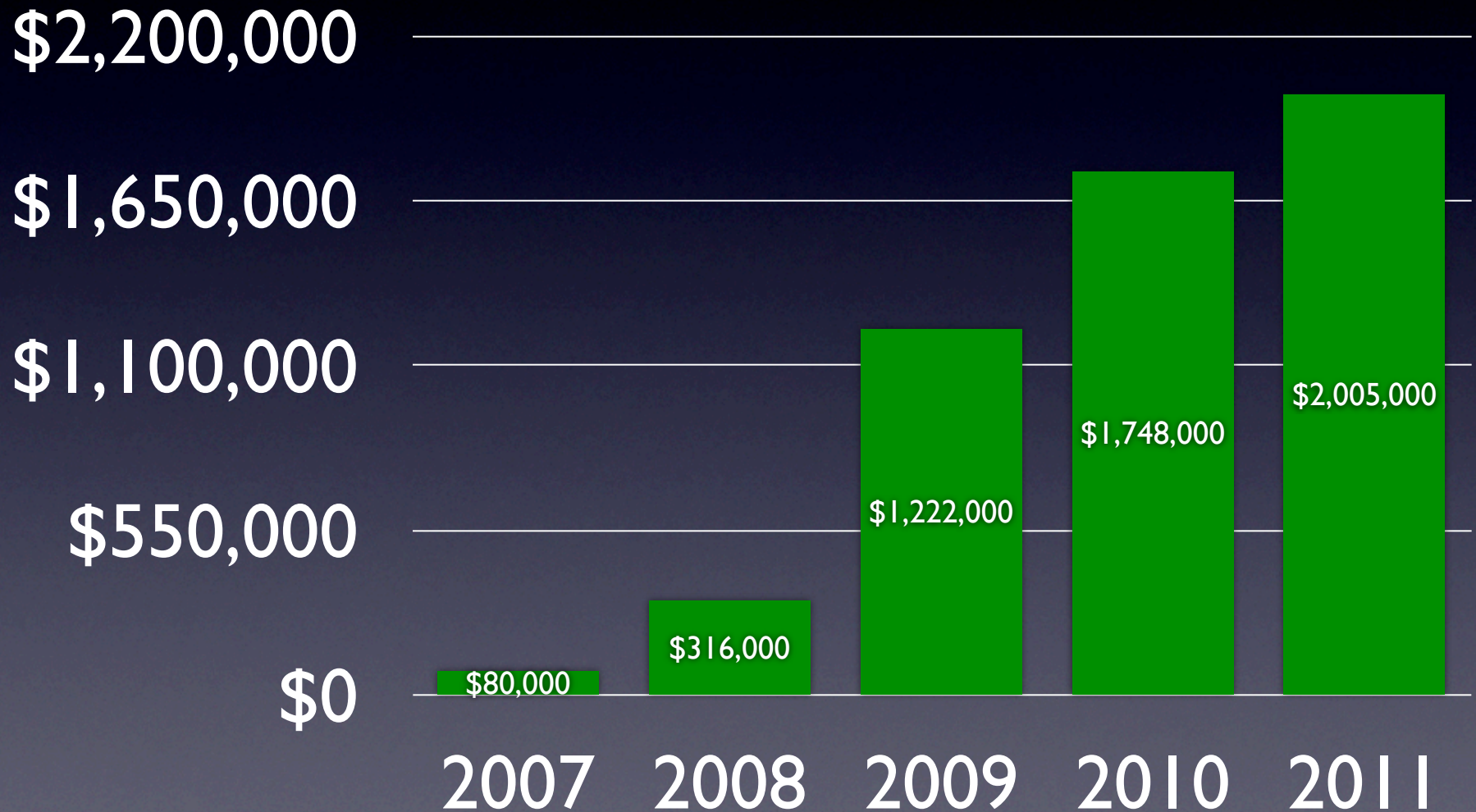
# Complexity & Coverage



# Continuous Deployment

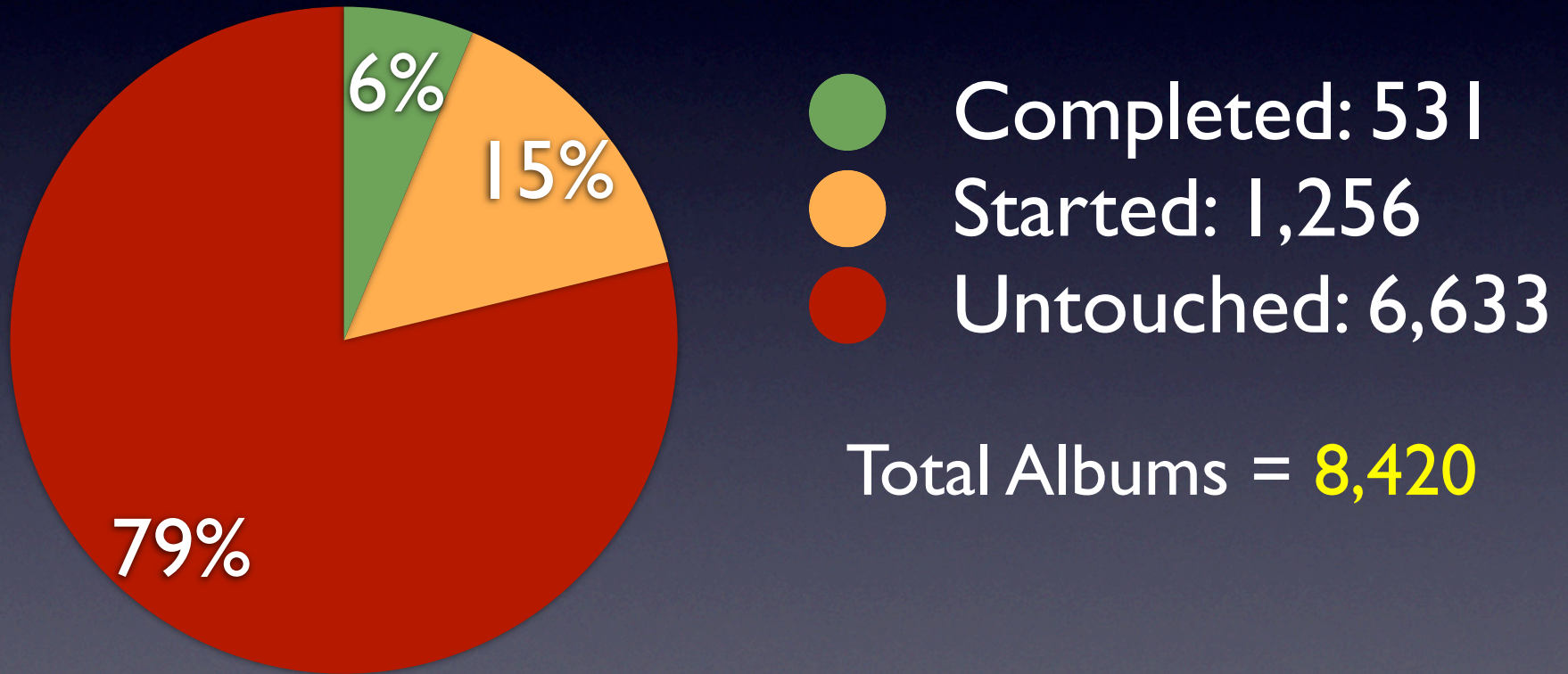


# Sales





# Album Usage Large Purchase



“A startup is a human institution designed to create a new product or service under conditions of extreme **uncertainty**.”

- Eric Ries

“A startup is a human institution designed to create a new product or service under conditions of extreme programming.”

- **Not** Eric Ries

# Done: Agile Style

...



...

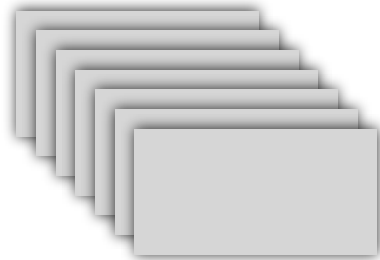


ACCEPTED

# Done: Lean Startup Style

...

ACCEPTED



VALIDATED



# Validated Learning

- Learning validated via data, surveys, interviews.
- Demonstrated by positive improvements in startup's core metrics.



**@SeanEllis**

Sean Ellis

In an early stage startup, success is not in a "to do" list, it's in a "to learn" list.

6 Jan via [Mobile Web](#) ☆ [Favorite](#) ↻ [Retweet](#) ↩ [Reply](#)

Retweeted by [trishmapinto](#) and 84 others



“In a startup no facts  
exist inside the  
building, only opinions.”

-**Steve Blank**



“Get outside  
of the building.”

-Steve Blank



EMAIL SIGN UP  
 DWR BLOG | DESIGN NOTES  
 REQUEST A CATALOG  
 DWR 3D ROOM PLANNER  
 TRADE & CONTRACT

ABOUT DWR | DWR LOCATIONS | 800.944.2233 | LIVE CHAT

MY ACCOUNT | CUSTOMER SERVICE | CART (0 items) \$0.00

NEW | LIVING | DINING | BEDROOM | OUTDOOR | WORKSPACE | STORAGE | LIGHTING | RUGS | ACCESSORIES | DESIGNERS | SALE

# THE YARD SALE

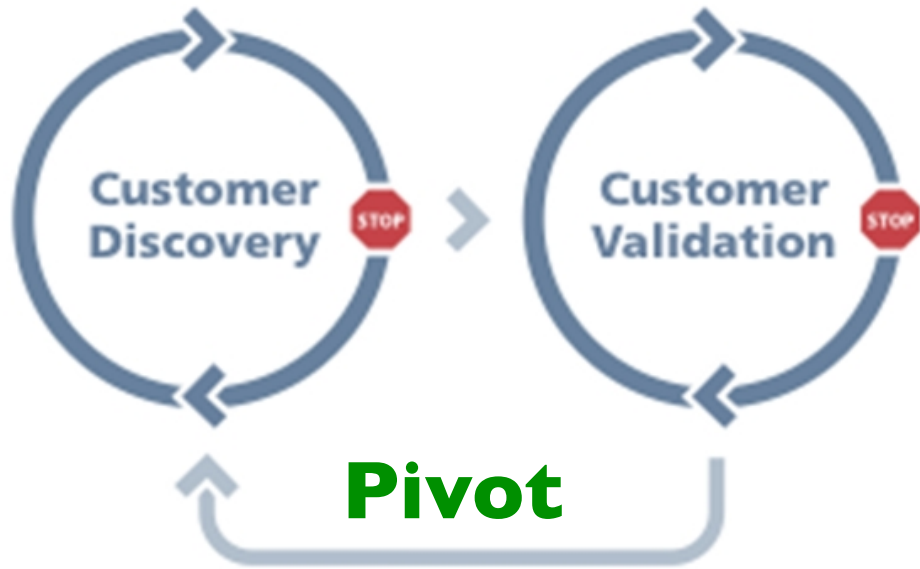
## SAVE UP TO 70%

Items are selling out and quantities are very limited.

▶ SHOP THE YARD SALE



## Iteration



The **Search** for a Business

## Execution



The **Growth** of a Business

**Product/Market Fit**

# What Is A Pivot?

“A special kind of **change** designed to **test** a new fundamental **hypothesis** about the product, business model or engine of growth.” - Eric Ries

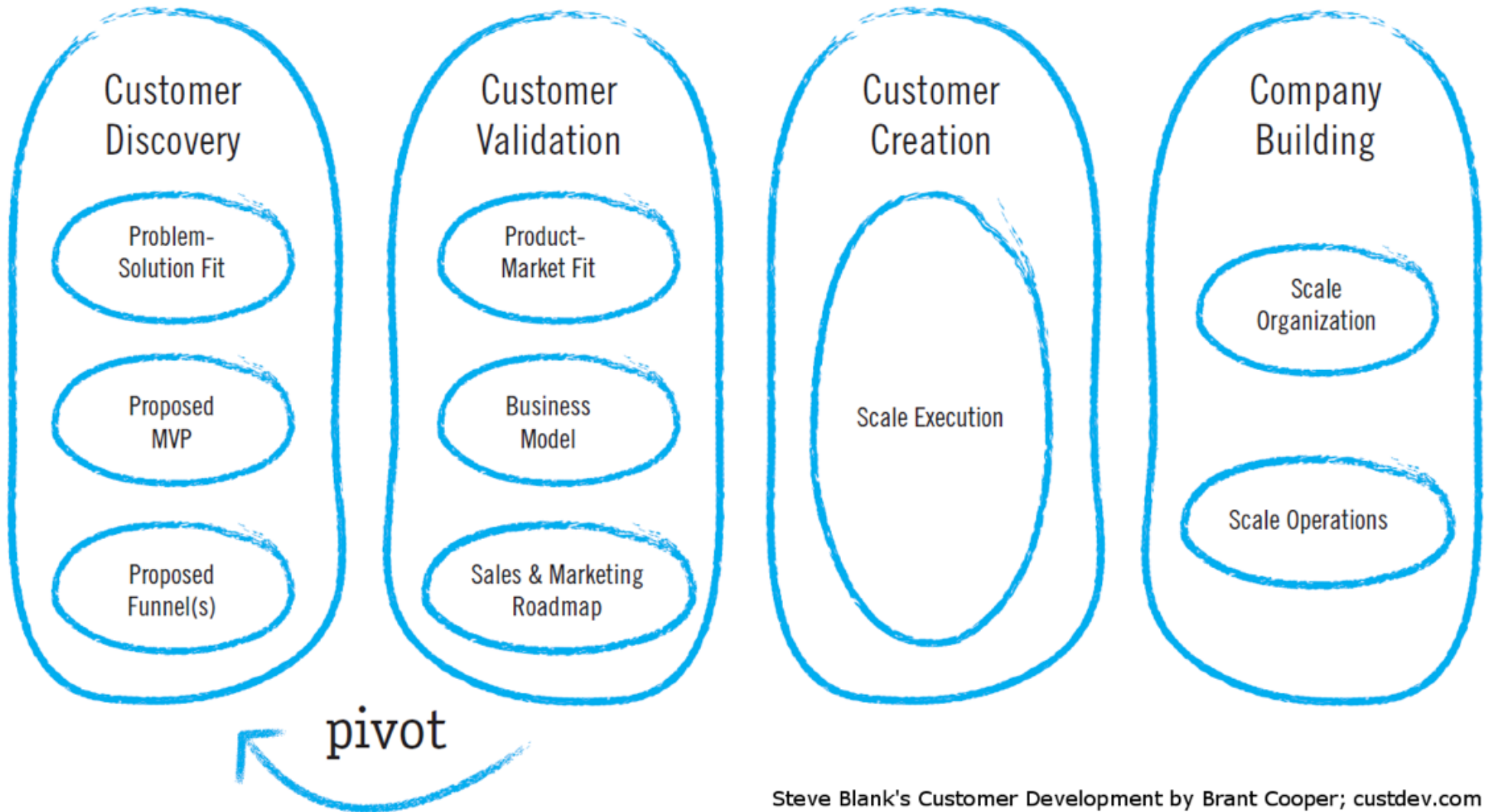


*"I'm not leaving you. I'm pivoting to another man."*

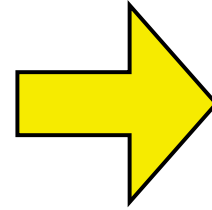
# Eric's Catalog of Pivots

- Zoom-In Pivot
- Zoom-out Pivot
- Customer Segment Pivot
- Customer Need Pivot
- Platform Pivot
- ...

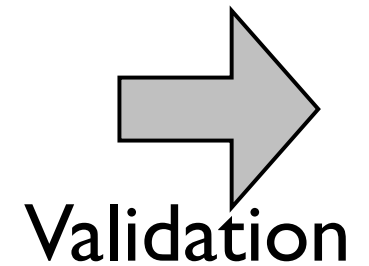
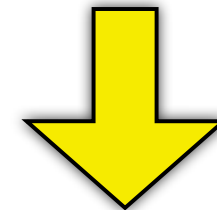
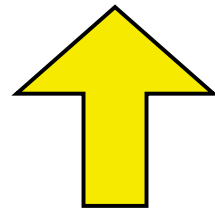
# Customer Development



**3. Test Product  
Concept**

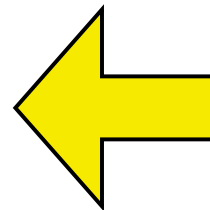


**4. Verify**



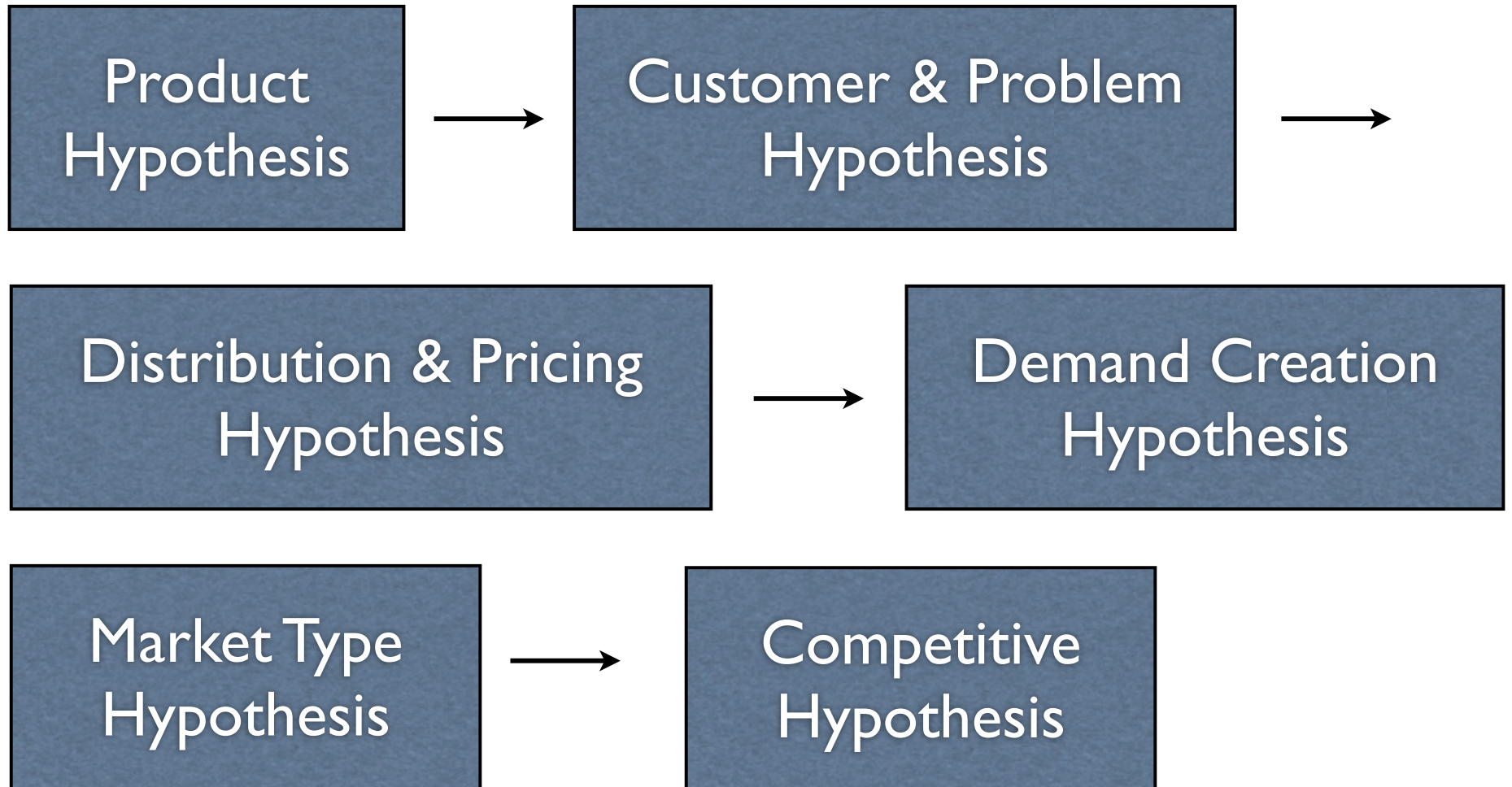
**2. Test  
Problem  
Hypothesis**

**1. State  
Hypothesis**





# State Product Hypothesis



Steve Blank:


“In addition to checking your assumptions about customer problems and your solution, you need to...”


“...validate your hypotheses concerning how customers actually spend their day, spend their money and get their job done.”



# YOUR VOTE IS POWERFUL

use it between elections with votizen

 Sign up with Facebook

 Sign up with Twitter

## 1 Claim your profile

We're able to verify you as a voter & **give you your voting history** in most states. What does your record look like?

[SEE a VOTER PROFILE >](#)

## 2 Write your officials

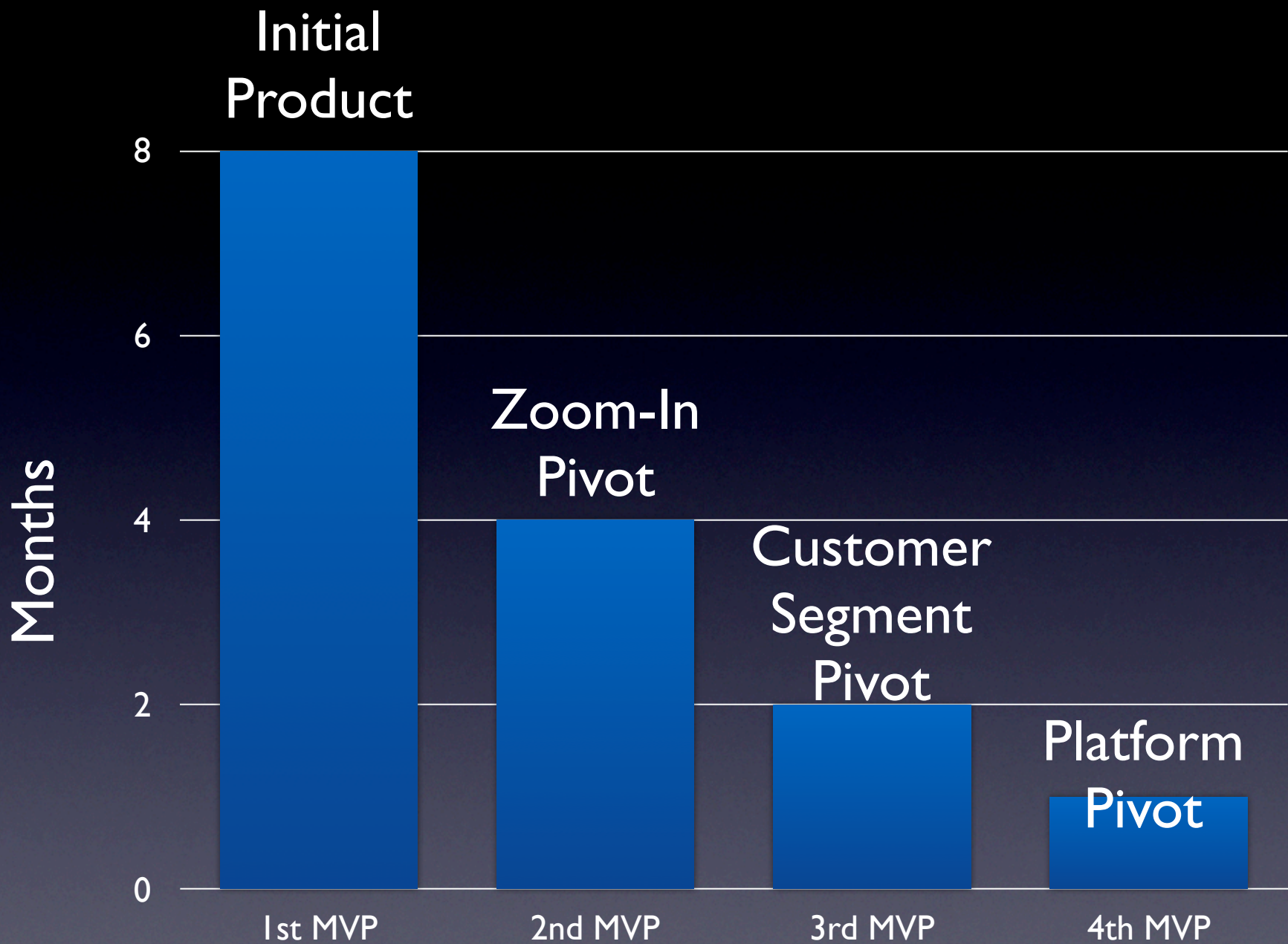
We **find & connect you with your officials**, and deliver your message backed by the power of your standing as a voter.

[SEE an OFFICIAL'S PAGE >](#)

## 3 Gather your Supporters

**Organize thousands of voters** so your message is delivered to all officials in Washington or your state capitol.

[SEE an OPEN LETTER >](#)



	Before Pivot 4	After Pivot 4
Engine of growth	Paid	Viral
Registration rate	42%	51%
Activation	83%	92%
Retention	21%	28%
Referral	54%	64%
Revenue	1%	11%
Lifetime value	Minimal	\$0.20/message

Source: The Lean Startup by Eric Ries

# AARRR: Metrics for Pirates

by Dave McClure

- **A**cquisition: users come to the product/site
- **A**ctivation: users “use” the product/site
- **R**etention: users use the site/product multiple times
- **R**eferral: users like product enough to refer others
- **R**evenue: users conduct some monetization behavior



## 40,000 FT VIEW

The Lean Startup movement is taking hold in companies both new and established to help entrepreneurs and managers do one important thing: make better, faster business decisions.

*“My belief is that these lean startups will achieve dramatically lower development costs, faster time to market, and higher quality products in the years to come.” - Eric Ries*

[See the Case Studies »](#)



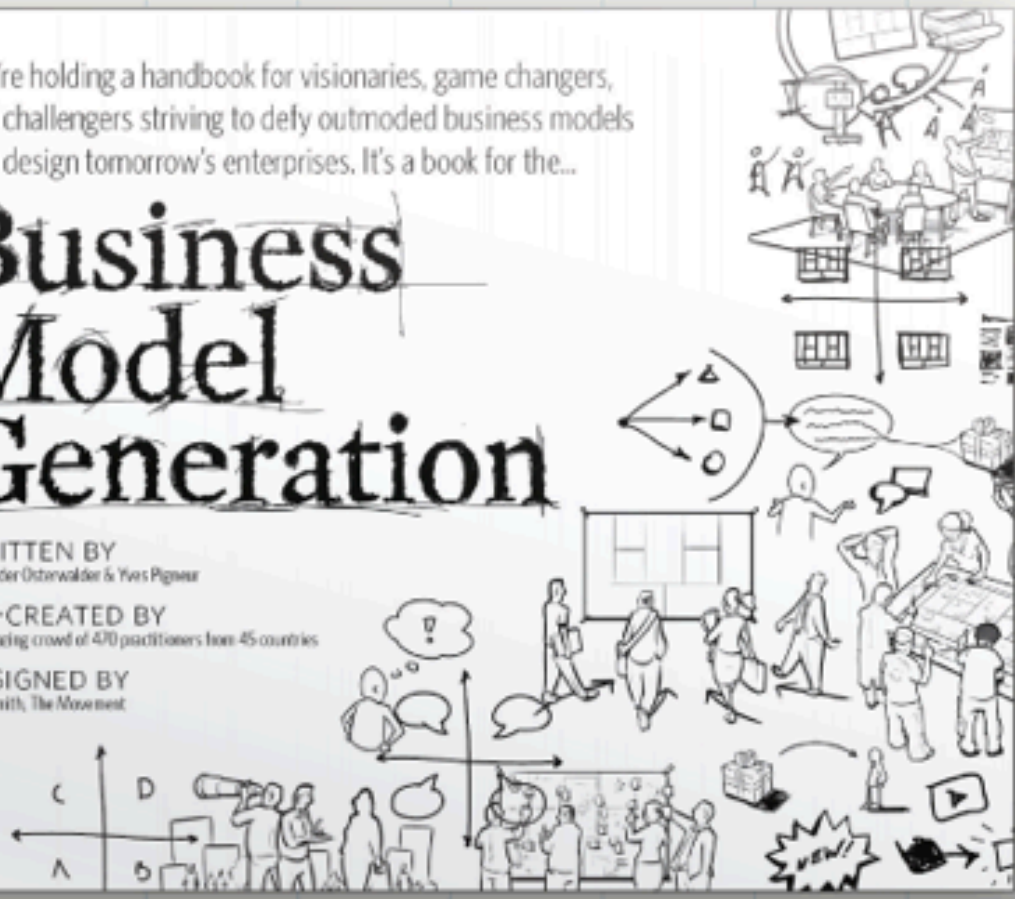
You're holding a handbook for visionaries, game changers, and challengers striving to defy outmoded business models and design tomorrow's enterprises. It's a book for the...

# Business Model Generation

WRITTEN BY  
Alexander Osterwalder & Yves Pigneur

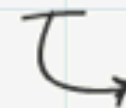
CO-CREATED BY  
An amazing crowd of 470 practitioners from 45 countries

DESIGNED BY  
Alan Smith, The Movement



Systematically understand, design & differentiate your business model.

A handbook for visionaries, game changers and challengers. The global best-selling phenomenon, available in 18 languages and counting. [Learn More](#)



Order now












# The Business Model Canvas

Designed for:

Designed by:

On:

Iteration:

<h3>Key Partners</h3>  <p>Who are our Key Partners?          Who are our key suppliers?          Which Key Resources are we acquiring from partners?          Which Key Activities do partners perform?</p> <p><small>Key Partnerships</small>          Supplier Partnerships          Distribution Partnerships          Co-opetition          Strategic Alliances</p>	<h3>Key Activities</h3>  <p>What Key Activities do our Value Propositions require?          Our Distribution Channels?          Customer Relationships?          Revenue streams?</p> <p><small>Key Activities</small>          Production          Distribution          Platform Development          Problem Solving          Infrastructure Development</p>	<h3>Value Propositions</h3>  <p>What value do we deliver to the customer?          Which one of our customer's problems are we helping to solve?          What bundles of products and services are we offering to each Customer Segment?          Which customer needs are we satisfying?</p> <p><small>Value Propositions</small>          New Products          Performance          Customization          Design          Cost          Convenience          Accessability          Risk Reduction          Time Savings          Convenience</p>	<h3>Customer Relationships</h3>  <p>What type of relationship does each of our Customer Segments expect us to establish and maintain with them?          Which ones have we established?          How are they integrated with the rest of our business model?          How costly are they?</p> <p><small>Customer Relationships</small>          Personal Assistant          Self-Service          Co-creation          Communities          Concierge          Dedicated Personal Assistant          Self-Service          Self-Service          Self-Service          Self-Service</p>	<h3>Customer Segments</h3>  <p>For whom are we creating value?          Who are our most important customers?</p> <p><small>Customer Segments</small>          Mass          Niche          Segments          Segments          Segments</p>		
<h3>Key Resources</h3>  <p>What Key Resources do our Value Propositions require?          Our Distribution Channels?          Customer Relationships?          Revenue Streams?</p> <p><small>Key Resources</small>          Physical          Intellectual          Financial          Human          Channels          Infrastructure</p>		<h3>Channels</h3>  <p>Through which Channels do our Customer Segments want to be reached?          How are we reaching them now?          How are our Channels integrated?          Which ones work best?          Which ones are most cost-efficient?          How are we integrating them with customer routines?</p> <p><small>Channels</small>          Direct          Indirect          Direct          Indirect          Direct          Indirect          Direct          Indirect          Direct          Indirect</p>			<h3>Cost Structure</h3>  <p>What are the most important costs inherent in our business model?          Which Key Resources are most expensive?          Which Key Activities are most expensive?</p> <p><small>Cost Structure</small>          Variable Costs          Fixed Costs          Variable Costs          Fixed Costs          Variable Costs          Fixed Costs          Variable Costs          Fixed Costs          Variable Costs          Fixed Costs</p>	<h3>Revenue Streams</h3>  <p>For what value are our customers really willing to pay?          For what do they currently pay?          How are they currently paying?          How would they prefer to pay?          How much does each Revenue Stream contribute to overall revenues?</p> <p><small>Revenue Streams</small>          Transactional          Subscription          Transactional          Subscription          Transactional          Subscription          Transactional          Subscription          Transactional          Subscription</p>



# Love Metrics

Delivered customer benefit?

Customers actively using it?

Customers tellings  
others about it?

**intuit.**





# Split (A/B) Testing

- Different versions of a product offered to customers at the same time
- Refine understanding of what customers want
- Measure productivity via validated learning, not production of features.

MailChimp

# Dashboard

create campaign ▾

my templates

my drafts ▾

## Recent Campaigns

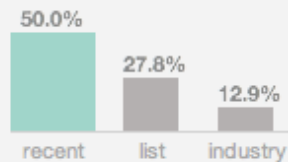
◀ [redacted] Resend June 3 - 2011 ▶

8 messages sent

6/3/11 12:34PM

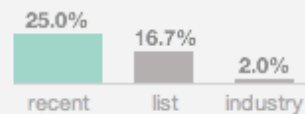
### Opens

4



### Clicks

2



opened	4	50.0%
bounced	0	0.0%
unopened	4	50.0%
click rate	25.0%	2 clicks

MailChimp

# Dashboard

create campaign ▾

my templates

my drafts ▾

## Recent Campaigns

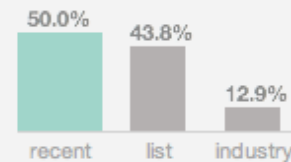
◀ [redacted] VIP - Resend (June 3) ▶

6 messages sent

6/3/11 11:52AM

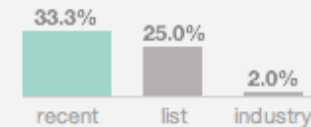
### Opens

3



### Clicks

2



opened	3	50.0%
bounced	0	0.0%
unopened	3	50.0%
click rate	33.3%	2 clicks

“These courses are a huge help for me; I switched into coding from audio production, so anything that covers gaps in my knowledge is great.”



Where Would You  
Click To Go To Page 8?



[Add To Playlist...](#)

4

## Connoisseurs of Code

### Wine lovers have an extensive vocabulary to describe wines.

acetic - acidity - aged - appley - astringent - austere - berrylike - big - bitter - bouquet - brawny - buttery - chewy - citrusy - cloudy - cloying - complex - creamy - crisp - delicate - dry - earthy - elegant - fat - filtered - fined - flat - fleshy - fruity - full-bodied - grapefruity - grapey - grassy - hard - herbaceous - leafy - lean - lingering - lively - lush - maderized - malolactic fermentation - meager - meaty - mouth-filling - nutty - oaky - oily - overripe - oxidized - peppery - perfumed - plump - ponderous - powerful - pruney - puckery - raisiny - refined - rich - rim - ripe - robust - rough - round - rustic...



### Now programmers have a growing vocabulary to help us become Connoisseurs of Code.



alternative classes with different interfaces - black sheep - conditional complexity - combinatorial explosion - data class - dead code - duplicated code - comment - feature envy - inappropriate intimacy - indecent exposure - lazy class - long method - long parameter list - large class - oddball solution - primitive obsession - refused bequest - solution sprawl - speculative generality - temporary field...

0 posts | [Post Your Thoughts](#)

Rate this Page (0 Ratings)





[Add To Playlist...](#)

4

## Connoisseurs of Code

### Wine lovers have an extensive vocabulary to describe wines.

acetic - acidity - aged - appley - astringent - austere - berrylike - big - bitter - bouquet - brawny - buttery - chewy - citrusy - cloudy - cloying - complex - creamy - crisp - delicate - dry - earthy - elegant - fat - filtered - fined - flat - fleshy - fruity - full-bodied - grapefruity - grapey - grassy - hard - herbaceous - leafy - lean - lingering - lively - lush - maderized - malolactic fermentation - meager - meaty - mouth-filling - nutty - oaky - oily - overripe - oxidized - peppery - perfumed - plump - ponderous - powerful - pruney - puckery - raisiny - refined - rich - rim - ripe - robust - rough - round - rustic...



### Now programmers have a growing vocabulary to help us become Connoisseurs of Code.



alternative classes with different interfaces - black sheep - conditional complexity - combinatorial explosion - data class - dead code - duplicated code - comment - feature envy - inappropriate intimacy - indecent exposure - lazy class - long method - long parameter list - large class - oddball solution - primitive obsession - refused bequest - solution sprawl - speculative generality - temporary field...

0 posts | [Post Your Thoughts](#)

Rate this Page ☆☆☆☆☆ (0 Ratings)





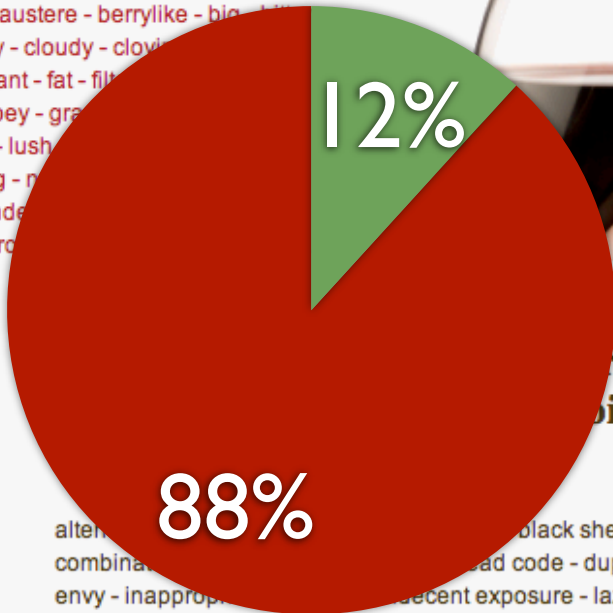
[Add To Playlist...](#)

4

### Connoisseurs of Code

**Wine lovers have an extensive vocabulary to describe wines.**

acetic - acidity - aged - appley - astringent - austere - berrylike - big - bitter  
- bouquet - brawny - buttery - chewy - citrusy - cloudy - clove  
creamy - crisp - delicate - dry - earthy - elegant - fat - full  
fleshy - fruity - full-bodied - grapefruity - grapey - grassy  
herbaceous - leafy - lean - lingering - lively - lush  
fermentation - meager - meaty - mouth-filling - nutty  
oxidized - peppery - perfumed - plump - ponderous  
puckery - raisiny - refined - rich - rim - ripe - roasty



**growing vocabulary  
Connoisseurs of Code.**

alternating - black sheep - conditional complexity -  
combination - dead code - duplicated code - comment - feature  
envy - inappropriate - recent exposure - lazy class - long method - long  
parameter list - large class - oddball solution - primitive obsession - refused bequest -  
solution sprawl - speculative generality - temporary field...

- Correct (2)
- Incorrect (15)

0 posts | [Post Your Thoughts](#)

Rate this Page ☆☆☆☆ (0 Ratings)



You are on page 20.  
Where would you  
click to quickly jump  
to page 40?



[Add To Playlist...](#)

20

## Dead Code

**All of us would dispose of a pet goldfish if it met an unhappy end. Yet how many of us dispose of code that is no longer being used?**



Code that is no longer used in a system or related system is **Dead Code**.

Far too much dead code exists in software systems.

Its presence often leads to:

- *Increased Complexity*. Dead code increases the size of a code base, which makes a system harder to comprehend, maintain and extend.
- *Accidental Changes*. Since it can be easy to mistake dead code for real code, programmers can waste time by accidentally extending or improving dead code without realizing it.
- *More Dead Code*. Once it becomes acceptable to leave dead code in a system, new dead code has a way of accumulating.

Just as great writing contains no unnecessary sentences, great software ought to contain no dead code.

9 posts | [Post Your Thoughts](#) | [Show Discussion](#)

Rate this Page (7 Ratings)





[Add To Playlist...](#)



## Dead Code

**All of us would dispose of a pet goldfish if it met an unhappy end. Yet how many of us dispose of code that is no longer being used?**



Code that is no longer used in a system or related system is **Dead Code**.

Far too much dead code exists in software systems.

Its presence often leads to:

- *Increased Complexity*. Dead code increases the size of a code base, which makes a system harder to comprehend, maintain and extend.
- *Accidental Changes*. Since it can be easy to mistake dead code for real code, programmers can waste time by accidentally extending or improving dead code without realizing it.
- *More Dead Code*. Once it becomes acceptable to leave dead code in a system, new dead code has a way of accumulating.

Just as great writing contains no unnecessary sentences, great software ought to contain no dead code.

9 posts | [Post Your Thoughts](#) | [Show Discussion](#)

Rate this Page (7 Ratings)



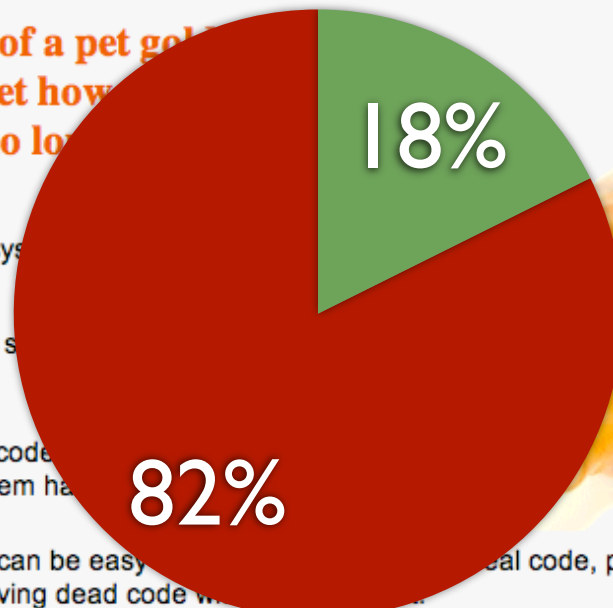


[Add To Playlist...](#)



### Dead Code

All of us would dispose of a pet goldfish that met an unhappy end. Yet how many of us would dispose of code that is no longer used in a system?



Code that is no longer used in a system is called **Dead Code**.

Far too much dead code exists in systems.

Its presence often leads to:

- *Increased Complexity.* Dead code increases the size of the code base, which makes a system harder to maintain and extend.
- *Accidental Changes.* Since it can be easy to accidentally delete or modify dead code, programmers can waste time by accidentally extending or improving dead code when they intended to do something else.
- *More Dead Code.* Once it becomes acceptable to leave dead code in a system, new dead code has a way of accumulating.

Just as great writing contains no unnecessary sentences, great software ought to contain no dead code.

● Correct (3)  
● Incorrect (14)

9 posts | [Post Your Thoughts](#) | [Show Discussion](#)

Rate this Page ★★★★★ (7 Ratings)





You are on page 20.  
Where would you  
click to jump ahead  
several pages?



[Add To Playlist...](#)

20

## Dead Code

**All of us would dispose of a pet goldfish if it met an unhappy end. Yet how many of us dispose of code that is no longer being used?**



Code that is no longer used in a system or related system is **Dead Code**.

Far too much dead code exists in software systems.

Its presence often leads to:

- *Increased Complexity*. Dead code increases the size of a code base, which makes a system harder to comprehend, maintain and extend.
- *Accidental Changes*. Since it can be easy to mistake dead code for real code, programmers can waste time by accidentally extending or improving dead code without realizing it.
- *More Dead Code*. Once it becomes acceptable to leave dead code in a system, new dead code has a way of accumulating.

Just as great writing contains no unnecessary sentences, great software ought to contain no dead code.

9 posts | [Post Your Thoughts](#) | [Show Discussion](#)

Rate this Page ★★★★★ (7 Ratings)





[Add To Playlist...](#)



## Dead Code

**All of us would dispose of a pet goldfish if it met an unhappy end. Yet how many of us dispose of code that is no longer being used?**



Code that is no longer used in a system or related system is **Dead Code**.

Far too much dead code exists in software systems.

Its presence often leads to:

- *Increased Complexity*. Dead code increases the size of a code base, which makes a system harder to comprehend, maintain and extend.
- *Accidental Changes*. Since it can be easy to mistake dead code for real code, programmers can waste time by accidentally extending or improving dead code without realizing it.
- *More Dead Code*. Once it becomes acceptable to leave dead code in a system, new dead code has a way of accumulating.

Just as great writing contains no unnecessary sentences, great software ought to contain no dead code.

9 posts | [Post Your Thoughts](#) | [Show Discussion](#)

Rate this Page ★★★★★ (7 Ratings)



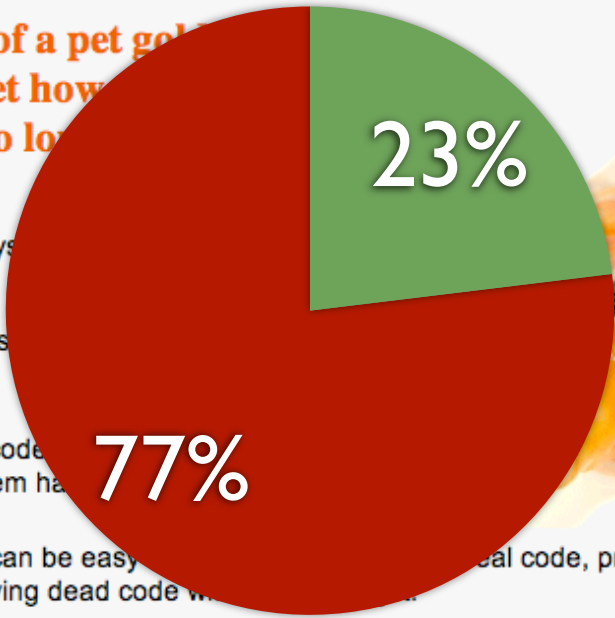


Add To Playlist...



### Dead Code

All of us would dispose of a pet goldfish that met an unhappy end. Yet how do we dispose of code that is no longer used in a system?



Code that is no longer used in a system is called **Dead Code**.

Far too much dead code exists in systems.

Its presence often leads to:

- *Increased Complexity.* Dead code increases the size of the code base, which makes a system harder to maintain and extend.
- *Accidental Changes.* Since it can be easy to delete dead code, programmers can waste time by accidentally extending or improving dead code when they meant to delete it.
- *More Dead Code.* Once it becomes acceptable to leave dead code in a system, new dead code has a way of accumulating.

Just as great writing contains no unnecessary sentences, great software ought to contain no dead code.

● Correct (3)  
● Incorrect (10)

9 posts | Post Your Thoughts | Show Discussion

Rate this Page ★★★★★ (7 Ratings)



You are on page 31.  
Where would you  
click to quickly jump  
ahead several pages?



[Add To Playlist...](#)

## Dead Code

**All of us would dispose of a pet goldfish if it met an unhappy end. Yet how many of us dispose of code that is no longer being used?**



Code that is no longer used in a system or related system is **Dead Code**.

Far too much dead code exists in software systems.

Its presence often leads to:

- *Increased Complexity*. Dead code increases the size of a code base, which makes a system harder to comprehend, maintain and extend.
- *Accidental Changes*. Since it can be easy to mistake dead code for real code, programmers can waste time by accidentally extending or improving dead code without realizing it.
- *More Dead Code*. Once it becomes acceptable to leave dead code in a system, new dead code has a way of accumulating.

Just as great writing contains no unnecessary sentences, great software ought to contain no dead code.

9 posts | [Post Your Thoughts](#) | [Show Discussion](#)

Rate this Page (7 Ratings)





[Add To Playlist...](#)

## Dead Code

**All of us would dispose of a pet goldfish if it met an unhappy end. Yet how many of us dispose of code that is no longer being used?**



Code that is no longer used in a system or related system is **Dead Code**.

Far too much dead code exists in software systems.

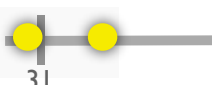
Its presence often leads to:

- *Increased Complexity*. Dead code increases the size of a code base, which makes a system harder to comprehend, maintain and extend.
- *Accidental Changes*. Since it can be easy to mistake dead code for real code, programmers can waste time by accidentally extending or improving dead code without realizing it.
- *More Dead Code*. Once it becomes acceptable to leave dead code in a system, new dead code has a way of accumulating.

Just as great writing contains no unnecessary sentences, great software ought to contain no dead code.

9 posts | [Post Your Thoughts](#) | [Show Discussion](#)

Rate this Page (7 Ratings)

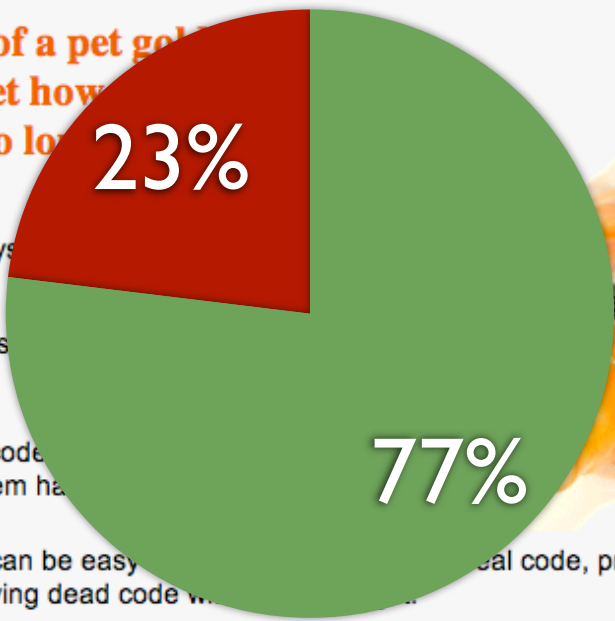




[Add To Playlist...](#)

### Dead Code

All of us would dispose of a pet goldfish that met an unhappy end. Yet how do we dispose of code that is no longer used in a system?



Code that is no longer used in a system is called **Dead Code**.

Far too much dead code exists in systems.

Its presence often leads to:

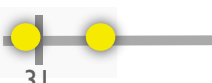
- *Increased Complexity.* Dead code increases the size of the code base, which makes a system harder to maintain and extend.
- *Accidental Changes.* Since it can be easy to accidentally delete real code, programmers can waste time by accidentally extending or improving dead code when they meant to delete it.
- *More Dead Code.* Once it becomes acceptable to leave dead code in a system, new dead code has a way of accumulating.

Just as great writing contains no unnecessary sentences, great software ought to contain no dead code.

● Correct (10)  
● Incorrect (3)

9 posts | [Post Your Thoughts](#) | [Show Discussion](#)

Rate this Page ★★★★★ (7 Ratings)



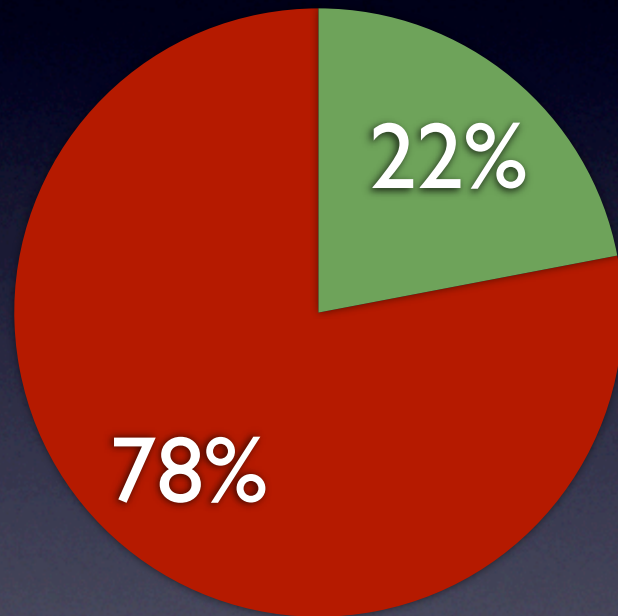


Laura Klein  
Lean UX  
Expert

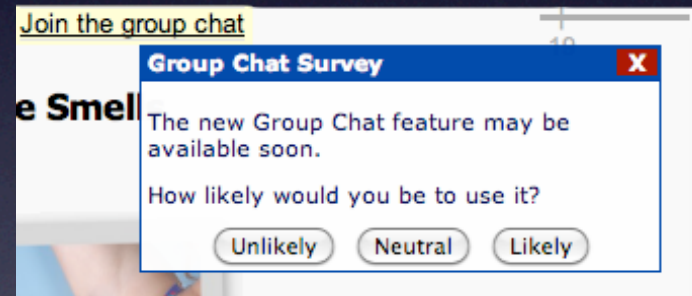


# Feature Fake

16 users from 4 countries  
are online right now.  
Join the chat.



- Used
- Not Used

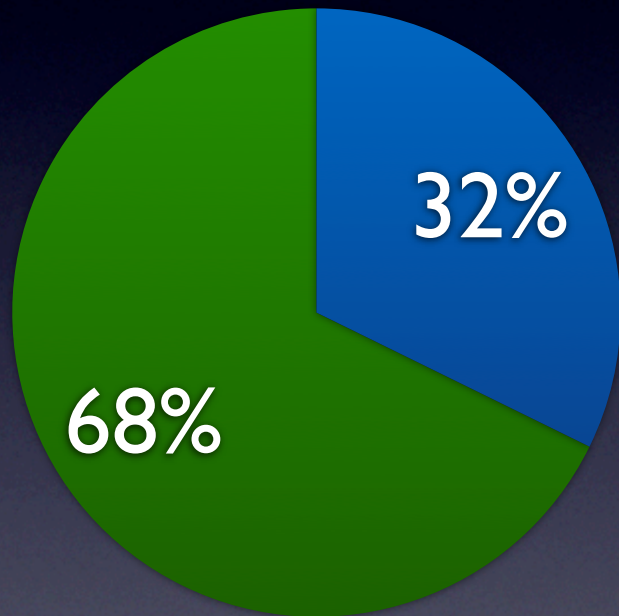


Unlikely	Neutral	Likely
1	7	6

# Personas, IL Website

<p align="center"><b>Influencers</b> Sell IL to the Approvers</p>	<p align="center"><b>Approvers</b> Approve the Proposal</p>
<p align="center"><b>@ Big Companies</b></p> <p>Who they are:</p> <ul style="list-style-type: none"> <li>• senior engineers // high level product managers</li> <li>• IL graduates – engineers or product managers depending on album</li> <li>• In-company trainers</li> </ul> <p>How they discover IL:</p> <ul style="list-style-type: none"> <li>• Know Josh / IL from conference, book, etc.</li> <li>• Took a course or workshop</li> <li>• Personal referral</li> </ul>	<p>Who they are:</p> <ul style="list-style-type: none"> <li>• senior level executives</li> <li>• Heads of Education / Training / L&amp;D</li> <li>• Head of business improvement</li> <li>• CTO / VP Product</li> </ul> <p>How they discover IL:</p> <ul style="list-style-type: none"> <li>• Influencers referrals</li> <li>• Search engines, SEO</li> </ul>
<p align="center"><b>Individual Purchasers</b> for Small Companies / Self-Study</p>	<p align="center"><b>Resellers</b> Companies and Individuals</p>
<p>Who they are:</p> <ul style="list-style-type: none"> <li>• senior employees at a startup/ small company</li> <li>• independent software engineers / job seekers</li> </ul> <p>How they discover IL:</p> <ul style="list-style-type: none"> <li>• Heard about it at a conference or referral</li> <li>• graduates – took a class at a former company</li> <li>• SEO -- searching specifically for agile, transition, workshops, extreme programming training</li> </ul>	<p>Who they are:</p> <ul style="list-style-type: none"> <li>• training consultants</li> <li>• scrum master trainers</li> </ul> <p>How they discover IL:</p> <ul style="list-style-type: none"> <li>• a conference, or referral from agile community</li> <li>• SEO -- searching specifically for agile, transition, workshops, extreme programming training</li> <li>• graduates – took a class at a former company</li> </ul>

# Measure Usage

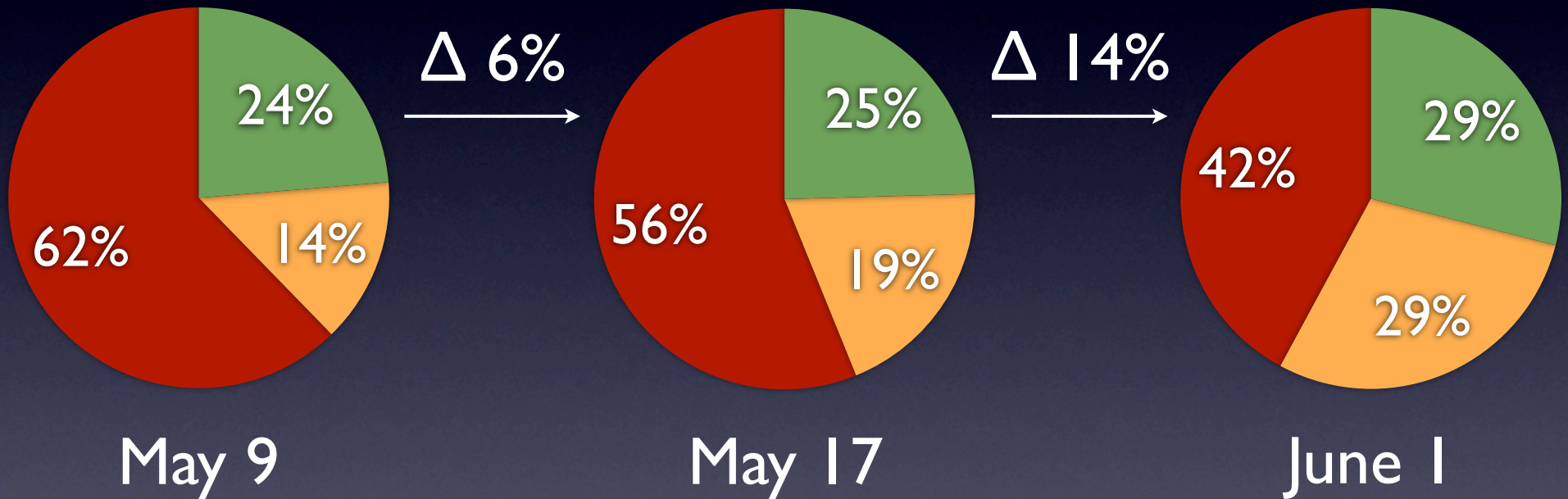


- Used
- Not Used

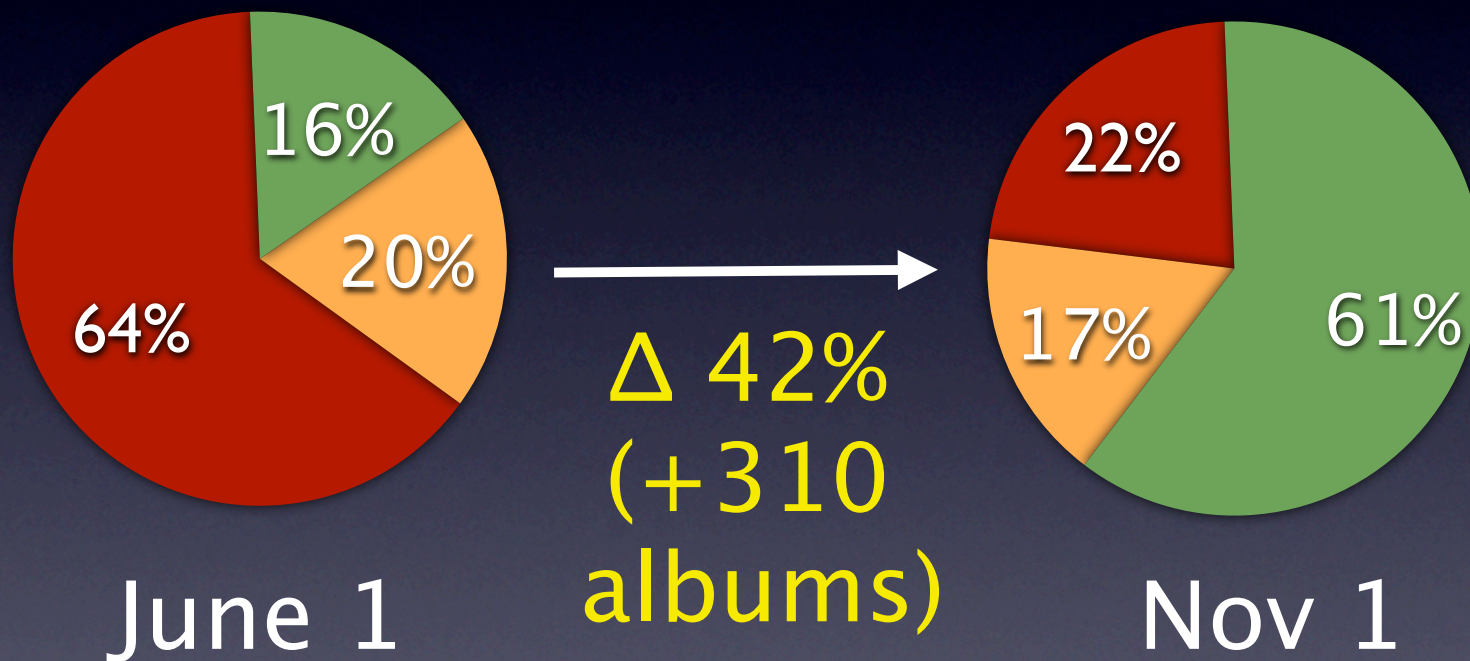
A screenshot of a music player interface. On the left is the album cover for 'Composing User Stories', which features a quill pen and musical notes. To the right of the cover is a blue 'Resume!' button. Below the button, the following text is displayed: 'Album: Composing User Stories', 'Track: Users and Roles', and 'Page: Favored or Disfavored?'.

Logins with Resume	Resumes	Non-Resumes
4965	1767 (36%)	3198 (64%)

# Usage Improvements



# 990 Albums Assigned to 160 Employees



# Agile

# Lean Startup

Product Roadmap

Business Model Canvas

Product Vision

Product Market Fit

Release Planning

Minimum Viable Product

Sprint

Learn/Measure/Build

On-Site Customer

“Get Out Of The Building”

# Agile

# Lean Startup

User Story

Hypothesis

Backlog

To Learn List

Customer Feedback

Customer Validation

Acceptance Test

Split Test

Continuous Integration

Continuous Deployment

Velocity/Burndown/etc.

AARRR

**Agile**

**Lean Startup**

Led By Successful  
Consultants

Led by Successful  
Entrepreneurs



“Get outside  
of the building.”

-Steve Blank

Thank You