

#BugsZero

eliminating bugs by not creating them in the first place

Arlo Belshee

http://bit.ly/PromiscuousPairingPdf http://bit.ly/AgileEngineeringFluency http://bit.ly/NamingIsAProcess http://bit.ly/BugsZeroSlides (this talk)

Team Craftsman, Legacy Code Mender, and Rabblerouser

Tableau & Innovating Teams

@arlobelshee, github/arlobelshee, http://arlobelshee.com/

The #BugsZero Story: Why We Think We Care





Why we Actually Care

Testers feel like



Product owners feel like

© 2007 GCMGA Inc., All Rights Reserved http://aggie-horticulture.tamu.edu/galveston

10

e Sint

1

by some

That

Developers feel like



The architecture feels like



Operations feels like



Sales has to pretend it is



But sales knows it actually is



Customers feel



And execs feel





But I Don't Have Technical Debt



Good design + 30 years of careful changes

Quick Poll: How Many Bugs?

- Function named DoContentHit
- In service for 30+ years
- 26,000 lines
- Uses undocumented system calls
- Uses gotos liberally
- Monitors keyboard and mouse

Need to add support for touch and pen

Quick Poll: How Many Bugs?

- Function named DoContentHit
- In service for 33+ years
- 18,000 lines, x2
- Uses undocumented system calls
- Uses gotos liberally, including jumps between functions
- Monitors keyboard, mouse, touch and pen

• Need to port to new OS, with different input devices



Bugs Annoy Me I Decided to Stop Having Them

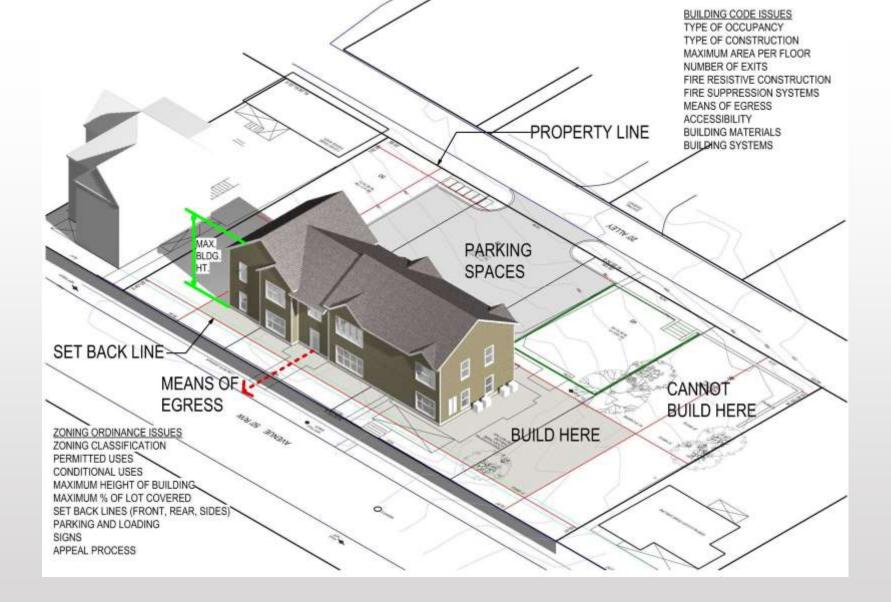
Bugs are Exciting Like Kitchen Fires



Fires Have Causes



Caused by a cow?



1,000,000 Fires Prevented

When people build a building, are they thinking about fire prevention?



How Do We Prevent Bugs?

To take an action for a fixed period of time, such that the probability of some undesirable outcome is permanently reduced.

"

Arlo

Definition of "Prevent"

Action + Context

Outcome

+ Context + Spread



Outcome

"

Anything that would **frustrate**, **confuse**, **or disappoint** some human, and is **potentially visible** to any human other than the one who created it. **y**

Arlo

Definition of "Bug"

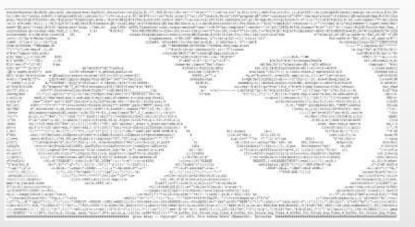
Actions that Can Cause Bugs

· Changing what a human receives

Coding

- Designing
- Changing what a human expects
 - Selling
 - Gaining insight about customer intent / need
 - Marketing
 - UI / UX
 - The world changing

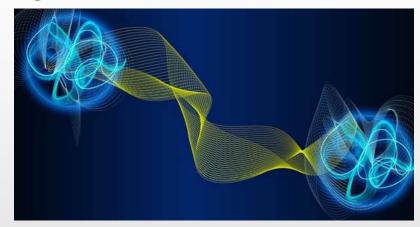
Contexts that Turn Those Actions Into Bugs



Unreadable code



Stuff developers don't know



Context sensitivity



Miscommunications between customer and developer

Situations that Spread Bugginess



Hurrying



Repeating Mistakes



Accepting Errors



High Risk of Change



People Notice Little Habit Drives Most Actions



Bugs Don't Come From What You Do

How do we change the habits that set our results?

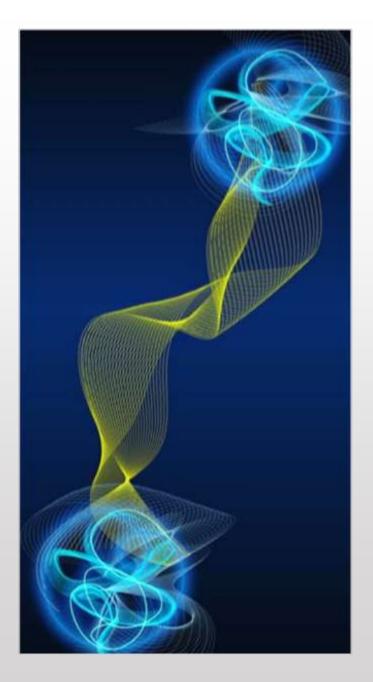
Unreadable Code

- Read by Refactoring Habits
- Insight Loop
- Core 6 Refactorings
 - Rename to store insights
 - Extract Method to divide operations
 - Intro Variable / Param / Field to divide data
 - Inline to bring things together

<pre>#(3*4))*/) [***#(4 1-p-0*3(Int.to3) (f(%(Module-QR:Su Transform'>cwalic Wm(STRING:=-224) *Program(public-)</pre>	<pre>n)):E->a['("'+e[e[+'")'):G- 4size/2)"+z]):C-*Console.We n'sEe"-"):p-O:p-1p-130:Lian ib-RK:Dim-n,n,i,c-3a-Objectin subput-method='teat'>>e(C+*' +153,a:=a-166,b:=a+*2**=== static-void=RX:Bystem.#(C+E)</pre>	rite":4D-"program-QB":4 .tabulate(127,p):5J.ms +=Chr(10):For-Each-c-' calitesplate")-match-' (50,c:=b+'5"+4+165,t: (%(lat-f(c-:Tht):Lat')-
	. n"@f-10@" "Method. 'n");let	
	r#DG-'to-the-bak'g-dish.'s'n5 'PROCEDURE-DEVISION.''DISPLA'	
	{o(((f-in)->&rray(n+1)-jo!	
	s-mx(H+[E]*[+["class-QR:Appli	
	"ZEZ"+4 E[1>+E[1"+4(+:-A"4	
	5*-#60*-0-5**-2-^*14*+0-	5-*-
	ckage-mal; import "fmt"; fund	- 23
	B[static-function-mX]	
	stic-ma:((1#15+*)	ava/lanç
	99999), y ="", z	-y, ==*# z-
87 800 0 0	x40esder+nam	e:⇔GR/Sectio
99992 8574601	Z#(E[%(. putp	scoenbly-t[].d ut]:beg]=2(4(f))
] au		exel]def(#10("gr:-
(c)) 'nZ('ca	ti"end	"n") '111.7'1
op-pop)forell		
11#1>#2144#1/#		
("MNCV", #w+2,128		
"B*") (0) , K{/_(8)/		
].scan(/[-,-:A-:		
[1-cei1(#/13));fc		
)*13, g, 1-13) -f (3) art'n startimov-e		
orlib)System.Cons		
DT #45 #588 #59E-		E11.#12e+1
(id*getelementpt:		(1)x-18]*8a,1 32
replace("-"", "Y-"		a.g):::+=""""+a +"
>(6-)) \$1,2);put5	52	(d);)for-c+!-ers
MGy==-1AFy>=?yl	1dCz	liditidiAutii")
[248] [E] 2.2=2.	[n] (b [r/78]Jb[r]JQ4
+(a.chazAt(11-13) 9847	11(1++++++2>0)
hethod]+N -u)\$256)]];E("DO,3<-#"];c:=c;[:H("PLEASE	READOL
1111.	#21("In+45'n"))));;Clo	set
1111	-**-C-5*+end-#sD*-C-A*	
	(*,*)(*)))))/***-**	
	D>n(Lec(n+62)\$92+35;D);a.h	392
-(05	90).f!d[k mot-s[1+1+k]s[]+	(k) } /
	s/87];Leep]0,u];b[0,u]="";x	
	<pre>[E[L]]):="":while=0<les(d):< pre=""></les(d):<></pre>	
	cast-1t-5 %92+6)):s+=s[len(s	
	(-0>a-else'-'*a)+'.');s=b)).t	
	.:," ^"389**6+44*6+00p25*,^" WENDDE(+c++(50+n2000)+b+" x4n	
	<pre>import # [C] [5-an-aChr (9) an-a"</pre>	
	MedEm-'s/(['x1eY] Y. (1,1	
ord847812]])))**	")#_buffer_for_future_bug_fi ight (c) 2013, 2014 Yusuke Er	

Context Sensitivity

- Dependency Elimination Principle
- Dependency-Breaking Patterns
- TDD alone doesn't help much
- Test by Refactoring Habits
 - Test as Spec
 - Acceptance Microtests
- Design by Refactoring Habits
 - Replace Supplier with Supplies
 - Breaking up God Classes iteratively
 - Follow fields to find multiple responsibilities



What Developers Don't Know

- Pairing, Tripling, or Mobbing
 - Especially across roles
- Micro-habits & Mind-shifts
- Customer visits
- Listen to the code



Miscommunication on Path from Customer to Developer

- Shorten the path
- Customer telemetry
- Pair with technical support, on both coding and support cases



Hurrying

- Prevent the obvious bugs with common causes
- This is failure demand; it goes away when the failures go away



Accepting Errors

- Count "days since last accident"
- Eliminate triage
- All bugs are more important than remaining features
- Change incentives for PO



Repeating Mistakes

- Immediate stop the line
- Immediate 5-whys
- 15% solutions for each contributing factor, as tasks on the board
- Priority:
 - 1. Mitigate
 - 2. Prevent this class of bugs
 - 3. Fix the bug
 - 4. Anything other than this bug



High Risk of Change

- Disciplined Refactoring
- Build / Measure / Learn
 - For product
 - For team capability
- Roll back non-successful experiments
- Simple design beats extensible design

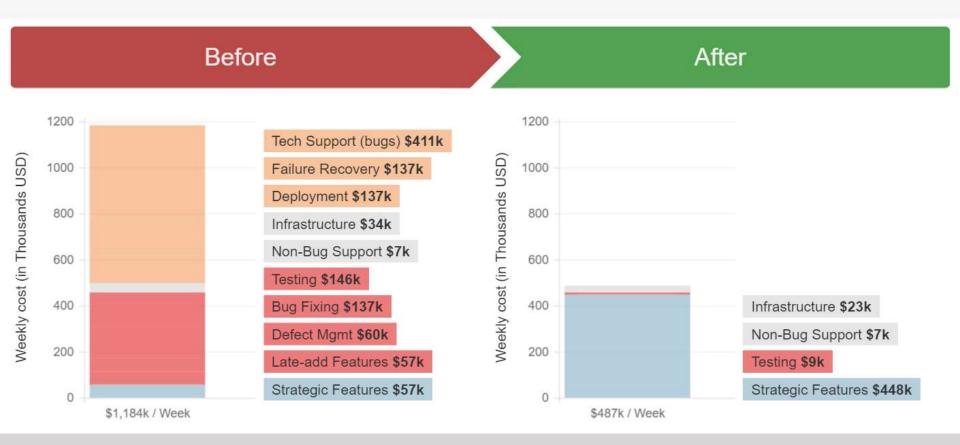




1. Fix the Environment that is causing errors

2. Learn from each bug to prevent more

DoContentHit (and its Application), 2 Subsequent Releases



Thank You

http://bit.ly/PromiscuousPairingPdf http://bit.ly/AgileEngineeringFluency http://bit.ly/NamingIsAProcess http://bit.ly/BugsZeroSlides (this talk)

@arlobelshee, github/arlobelshee, http://arlobelshee.com/