

Failure is a four-letter word

<u>Andreas Zeller</u> • Thomas Zimmermann • Christian Bird PROMISE 2011, Banff, Canada

Software failures

			Mozilla	Vulnerabil	lities								
security	mailnews			con	itent			extens	ons			nsprpub	
nss	base	imap	base		xslt	xul	canvas	s3d webservie	e python	spellch		pr	
lib	src util	SIC	SITC	p	SIC	temp doc	SIC	soap pr	o xpco	SIC	SFC	test	ts.
libpkix freebl softoken				×	sit xpath	src src		sche w	i walle u	niv sche	md		
modu pki sy	search			世界日		cont	1	java	src	src src		b	
	adddaak compose	Import	hteel	the second se	svg	events xm	ti	xpcom	met pro	e ins typ	misc oth	re internet	
pkix incl ssl util certd smim	addroook compose	Import	rontent	der CO	ontent	src d	xiom	ns	SI ST au	w s p	the the	inclu	ide
top uti r	SIL SIL	COLL SIC	ser	577	STC	xbl xtf		xmiterm	TI SQL		- See the	cp md p	2
		eud oex	CETHIN		間に	src ca		Case line	b xn	i III		L- End	1
ckfw pkr pkr pkr pkr b			X	xom		directory		db	1	ef		xpinst	tall
uiltins ca pkllwr pkcsl2	mime	news exten	io	glue		c-sdk		sqlite3	0	ompiler I	Utilitie	wizar	rd
ns pki1 jar cry	SIC SIC	src paim				Idap	0.00	SITC	Co	de Front	Gener	windows	lib
certhig has his		bs	1-48		111	ibraries o	clie i		n	d Lill	zi ga	setup uni	G
	mapi	dh h	reflect	string two	pelib lib	Idap - 1	B Local		x	Primi	C		3
cmd t	old ma	00 0	xptcal x	pu sr xp	pi x	suncsdk	xp		11版 日			052	n
zlib lib m pk si fips pk ce c	modules		S/C S	Die H H	2D	c-sdk	b	mork	tri Ru	ntim gc	Pack	setup	10
Ha a criu bit S	oji	plugin	ds	base te	sts	Idap		SIC	SF SY	ste sr i	100		
manager jss s	tests sr to	ols sam			iit, lib	raries cli	1	-H-FELLINK	md c	N C Tools	CO D	SIC	_
org	src test	s s def		build con	mpo 📒			11-11-11-11-11-11-11-11-11-11-11-11-11-	i – b	HE TH	Ex D		11
layout	JNI T		obsolete	- High	r nr	intl		editor		toolkit		xpf	e
generic style xul	C Arr A C b	456 D	C	MoreFi	s	uconv		libeditor	txm co	mponents	airbag	compone	boo
base	AC THE PU			idaet	ucv	at src ut	ucv	html base	pla	ace his s	airbag	sear boo	app
SPC	libimg libfont lii	bpr0n zlib		src	1000		Ut	——	com s	do na	NIG IT	hi	
	png jmcge d	ec s src	mac	gtk2	1	UCVCN me		text	txtsv	- TT		ent	200
	S/C					Har Lierale	101	CHIN COM		THE THE	HWT	st —===	- pp
tables mathmi forms	THE REPORT OF	bre libo libb	L HAR		unic	nar iocale	500	catendar	parser	and a second	tools	acce	Issit
base base	rdf softupd s	rc src src	windows	os2 be	eos	chardet	SIL .	noical I	sec	expa trace	e- codes	re S	SIC 1
src	STC STC	hiar xml s li				Fine Street	COD TH	bical libic	sic p		re d f	p atk ba	13 1
svg Hills	HITTER LEINE	pro		xpwi qt	ph	natural.		a Little come	188		le jp		2 1
base re prin in ht bu	at at	Delano Litt	gtk	日前開		netwerk		* test	dom	msgsdi	k c	ck	9
	grx	-	1	xlib co o	0 0	nc http	ftn	base	src	C	expl	t muc I	boe
js	ps vib mar	theb with			μ T	sec.		is2	base	protoco		ib c	H
src tamarin				adding				src		fil. LE	Ednve		HE
xpconnect liveco core		qt phot	bo	nwser	stre	amco test	co	re	119	D plugin	uriloa	der camino	2
src test	otk windo	the wo leb	activex	atk in	bot	HEIST	buil		ther liters	oji .	extha	b src	T
		DI LIHIFI	SIC	Src	Ca	che dns	S		Pest libert	MRJ MR	0	L	
	x11sh	xpr	co co	THE W	veb		- ala		src LLL	plu pl	lib	imston vie	EW.
pcre code MM	cairo	thebe	plu pl	powerp a	1 1	java	_	LITHS	721	rdf	mac	SIC SIC	c
	cairo d	itz src	COLUMN T		well	oclient plu	99ab	browser	atk-	L base ch	ro	Buil Idhon	
	STC 5	rc - Hill	compon	qa tes	sts src.	moz w	f co	omponents	proj	src d	profile	win s i	ste
shell pl e		publi	printin te	este mfc	W		plu plu	ices migrat	THU	Comment of	SF		34.0
	libp	ixma publ	win fi		X	pcom 00	più s	rc boo s	Fitth	docshel	config	caps sto g	gcor
		rc Hille	web		te	a u		THE T	上日指	Dase	6 CH	src sr	chro
		A REAL PROPERTY AND A REAL	111 M		and the second s	And a state of the		and the second se			Tarray and	And Personal Property lies, or other	

Defect distributions













Cost of consequence

Back to basics



Back to basics



Basic actions

public class ImageViewerPlugin extends AbstractUIPlugin {

```
//The shared instance.
private static ImageViewerPlugin plugin;
```

```
/**
 * The constructor.
 */
public ImageViewerPlugin() {
    plugin = this;
}
```



```
/**
 * This method is called upon plug-in activation
 */
public void start(BundleContext context) throws Exception {
    super.start(context);
}
```

Basic actions

public class ImageViewerPlugin extends AbstractUIPlugin {

```
//The shared instance.
private static ImageViewerPlugin plugin;
/**
 * The constructor.
 */
public ImageViewerPlugin() {
   plugin = this;
}
/**
 * This method is called upon plug-in activation
 */
public void start(BundleContext context) throws Exception {
   super.start(context);
}
```

No abstraction

////// *******************************	tttttttttttttt uuuuuuuuuuu V WWWW A B C E IIIII PPPP
nnnnnnnnnnn	TTT
00000000 ppppppppp	VVV
rrrrrrrr	{{{
SSSSSSSSS	}}}

No abstraction

- 1. We can predict defects from programmer actions.
- 2. We can isolate defect-prone programmer actions.

- 1. We can predict defects from programmer actions.
- 2. We can isolate defect-prone programmer actions.
- 3. We can prevent defects by restricting programmer actions.



2. We can isolate defect-prone programmer actions.

3. We can prevent defects by restricting programmer actions.

Release	Total chars	Total files	Files with defects
Eclipse 2.0	44,914,520	6,728	975 (14%)
Eclipse 2.1	56,068,650	7,887	854 (11%)
Eclipse 3.0	76,193,482	10,593	1,568 (15%)

Eclipse bug data [PROMISE 2007]



Eclipse characters

Training Set	Eclipse 2.0	Eclipse 2.1	Eclipse 3.0	Average
Eclipse 2.0	0.74	0.39	0.49	0.54
Eclipse 2.1	0.55	0.64	0.56	0.58
Eclipse 3.0	0.57	0.40	0.64	0.54
Average	0.62	0.47	0.56	0.55

Precision

Training Set	Eclipse 2.0	Eclipse 2.1	Eclipse 3.0	Average
Eclipse 2.0	0.74	0.39	0.49	0.54
Eclipse 2.1	0.55	0.64	0.56	0.58
Eclipse 3.0	0.57	0.40	0.64	0.54
Average	0.62	0.47	0.56	0.55

Precision

Training Set	Eclipse 2.0	Eclipse 2.1	Eclipse 3.0	Average
Eclipse 2.0	0.74	0.39	0.49	0.54
Eclipse 2.1	0.55	0.64	0.56	0.58
Eclipse 3.0	0.57	0.40	0.64	0.54
Average	0.62	0.47	0.56	0.55

Precision

Training Set	Eclipse 2.0	Eclipse 2.1	Eclipse 3.0	Average
Eclipse 2.0	0.32	0.27	0.27	0.28
Eclipse 2.1	0.03	0.18	0.14	0.11
Eclipse 3.0	0.19	0.16	0.20	0.18
Average	0.18	0.20	0.20	0.19

Recall

2. We can isolate defect-prone programmer actions.

3. We can prevent defects by restricting programmer actions.



- 2. We can isolate defect-prone programmer actions.
- 3. We can prevent defects by restricting programmer actions.





2. We can isolate defect-prone programmer actions.

3. We can prevent defects by restricting programmer actions.













2. We can isolate defect-prone programmer actions.

3. We can prevent defects by restricting programmer actions.



2. We can isolate defect-prone programmer actions.

3. We can prevent defects by restricting programmer actions.
1. We can predict defects from programmer actions.

2. We can isolate defect-prone programmer actions.

3. We can prevent defects by restricting programmer actions.

Explicit causes



Explicit causes



IROP keyboard

if (p != null) { int i; while (p[i] < 0) i++; return i; }</pre>

if (p != null) { int i; while (p[i] < 0) i++; return i; }</pre>



when (q != null) { num n; as (q[n] < 0) n++; handback n; }</pre>

when (q != null) { num n; as (q[n] < 0) n++; handback n; }</pre>

when (q != null) { num n; as (q[n] < 0) n++; handback n; }</pre>

100% Semantics preserving

New habits

We can shun these set majuscules,

and the text stays just as swell as

antecedently. Let us just ban them!

New habits

1. We can predict defects from programmer actions.

2. We can isolate defect-prone programmer actions.

3. We can prevent defects by restricting programmer actions.

- 1. We can predict defects from programmer actions.
- 2. We can isolate defect-prone programmer actions.
- 3. We can prevent defects by restricting programmer actions.



1. How about external validity? (findings based on ≥177,000,000 characters + 1,000s of defects; one of largest studies ever)

1. How about external validity? (findings based on ≥177,000,000 characters + 1,000s of defects; one of largest studies ever)

2. Are the correlations significant? (yes - all of them)

1. How about external validity? (findings based on ≥177,000,000 characters + 1,000s of defects; one of largest studies ever)

- 2. Are the correlations significant? (yes all of them)
- **3. Are the measures appropriate?** (all code originally input as characters; no abstraction taken that could interfere)

• Automatic renamings (PROMISE \rightarrow ENGAGEMENT, Eclipse \rightarrow Eclse)

• Automatic renamings (PROMISE \rightarrow ENGAGEMENT, Eclipse \rightarrow Eclse)

Abstraction

(Failure / mistake / error / problem / bug report vs. success / fame)

• Automatic renamings (PROMISE \rightarrow ENGAGEMENT, Eclipse \rightarrow Eclse)

Abstraction

(Failure / mistake / error / problem / bug report vs. success / fame)

• Generalization (ИРОП principle)

Failure is a four-letter word



Defect correlations

IROP

Defect correlations

Failure is a four-letter word



IROP keyboard

- 1. We can predict defects from programmer actions.
- 2. We can isolate defect-prone programmer actions.
- 3. We can prevent defects by restricting programmer actions.



Defect correlations

IROP

Defect correlations

Failure is a four-letter word



IROP keyboard

- 1. We can predict defects from programmer actions.
- 2. We can isolate defect-prone programmer actions.
- 3. We can prevent defects by restricting programmer actions.

Why all this is wrong



Defect correlations

IROP

Defect correlations

Failure is a four-letter word



IROP keyboard

- 1. We can predict defects from programmer actions.
- 2. We can isolate defect-prone programmer actions.
- We can prevent defects by restricting programmer actions.

Correlation vs. Causation

////// *******************************	ttttttttttttttttt uuuuuuuuuuuuuuuuuuuu	
mmmm	PPPP	
000000000 ppppppppp	U VVV	
rrrrrrr	{{{	
SSSSSSSSSSS	}}}	

No abstraction

Machine Learning works

Training Set	Eclipse 2.0	Eclipse 2.1	Eclipse 3.0	Average
Eclipse 2.0	0.74	0.39	0.49	0.54
Eclipse 2.1	0.55	0.64	0.56	0.58
Eclipse 3.0	0.57	0.40	0.64	0.54
Average	0.62	0.47	0.56	0.55

Precision

Cherry Picking



Defect correlations

Fix Causes, not Symptoms



Actionable Findings



Defect correlations

Our Inspiration

http://xkcd.com/882/












Use Book in Class



Use Paper in Class

Failure is a Four-Letter Word

- A Parody in Empirical Research -

Andreas Zeller^{*} Saarland University Saarbrücken, Germany zeller@cs.uni-saarland.de

Thomas Zimmermann Microsoft Research Washington, USA tzimmer@microsoft.com

Christian Bird Microsoft Research Washington, USA cbird@microsoft.com

ABSTRACT

Background: The past years have seen a surge of techniques predicting failure-prone locations based on more or less complex metrics. Few of these metrics are *actionable*, though.

Aims: This paper explores a simple, easy-to-implement method to predict and avoid failures in software systems. The IROP method links *elementary source code features* to known software failures in a lightweight, easy-to-implement fashion.

Method: We sampled the Eclipse data set mapping defects to files in three Eclipse releases. We used logistic regression to associate programmer actions with defects, tested the predictive power of the resulting classifier in terms of precision and recall, and isolated the most defect-prone actions. We also collected initial feedback on possible remedies.

Results: In our sample set, IROP correctly predicted up to 74% of the failure-prone modules, which is on par with the most elaborate predictors available. We isolated a set of four easy-to-remember recommendations, telling programmers precisely what to do to avoid errors. Initial feedback from developers suggests that these recommendations are straightforward to follow in practice.

Conclusions: With the abundance of software development data, even the simplest methods can produce "actionable" results.

number of developers associated with a file. As elaborate as these approaches may be, they all share the same problem which we call *the cost of consequence*: If I know that a module is failure-prone because it frequently changes, should I stop changing it? If I know failures are related to complexity, should I rewrite it from scratch? Any of these measures induces a new risk – a risk which may be greater than the one originally addressed.

In this paper, we take a different approach. We predict failures from the most basic actions programmers undertake, focusing on the actions that introduce defects *as they are being made* – literally at the moment the source code is typed in. Our recommendations are *immediately actionable*: A simple visual representation associates actions with the likelihood of introducing defects – warning programmers before they might hit the wrong key. Our approach is both effective and efficient: In a case study on the Eclipse failure set, it correctly identified up to 74% of the failureprone modules, which is on par with the most elaborate predictors available. Specifically, our contributions include:

- 1) A novel mechanism to associate programmer actions with software defects;
- 2) A predictor that is purely text-oriented, thus lightweight, real-time, easy to implement, and language-agnostic;

Why all this is wrong

Failure is a four-l	etter word
Andreas Zoller • Thomas Zimmern	ann - Christian Bird
PROMISE 2011, Banff,	Canada

Machine Learning works

Eclipse 2.0 0.74 0.39 0.49	0.5	0.49		A 10 A	College 2 C
		0.56	0.55	0.74	Eclipse 2.0
Eclipse 3.0 0.57 0.40 0.64	0.5	0.56	0.40	0.53	Eclipse 2.1
Average 0.62 0.47 0.56	0.5	0.56	0.47	0.62	Average

http://www.st.cs.uni-saarland.de/softevo/irop/

Actionable Findings



Use Paper in Class

Failure is a Four-Letter Word - A Parody in Empirical Research -

Andreas Zeller Saarland University Restricters Germany zeller@cs.uni-saarland.de

Thomas Zimmermann Microsoft Research Washington, USA tzimmer@microsoft.com

ABSTRACT

Background: The past years have acco a surget of techniques producing failure prose locations hased on more or less complex matrice. Prev of these metrics are actionable, though. matrice. Few of these metrics are astronable, though. Aime: This paper registers a simple, corp-in-implement method to produc and avoid fahara in software spheres. The IROP method links elementary merce code Software to known software

fachares to a lightworight, easy to implement flathon. Methank: We sampled the Eclipse data set magning defects to

Christian Bird Microsoft Research Washington, USA obird@microsoft.com

matches of developers associated with a file. As elaborate as they inition at perception particular that it is a total of a distance in the approaches only its, fixey all datas fits same publics which we sail the curst of consequences 11 fitspore limit a weakle in histore-prese because it frequently changes, should 1 meetin it fits income fitspace to the same state of the same state of the scatter of these means induces a term list – a trick which, may be greater than the one originally addressed

In this paper, we take a different approach. We predict failure Merhadt. We seepled the Tollpet data set express defects to flow in these filtiples trianues. We used logitist regression that seesare of the seeding densities in items of previous end that predictive part solutions with defects, second the predictive part solution is interned of previous end that predictive part solutions with defects, second the predictive part solution is interned of previous end that and the interlation end ends total freducts on presenting data in the second previous ends total and the second end of the second end of the internet total freducts on previous ends and the second ends total freducts on previous ends and the second ends total freducts on previous ends and the second ends total freducts on previous ends and the second ends recombined and and the second end of the second ends recondenses. With the short freduction of and the second ends recondenses. To the development and the second ends recondenses.
A result is an analytic the solution of the second ends recondenses. To the development and the second ends recondenses.
A result is an analytic the solution of the second ends recondenses.
A result is an analytic the solution of the second ends recondenses.
A result is an analytic the solution of the second ends recondenses.
A result is an analytic the solution of the second ends recondenses.
A result is an analytic the solution of the second ends recondenses.
A result is an analytic the solution of the solution of the second ends recondenses.
A result is an analytic the solution of the solution of