Collecting and Interpreting Bug Metrics

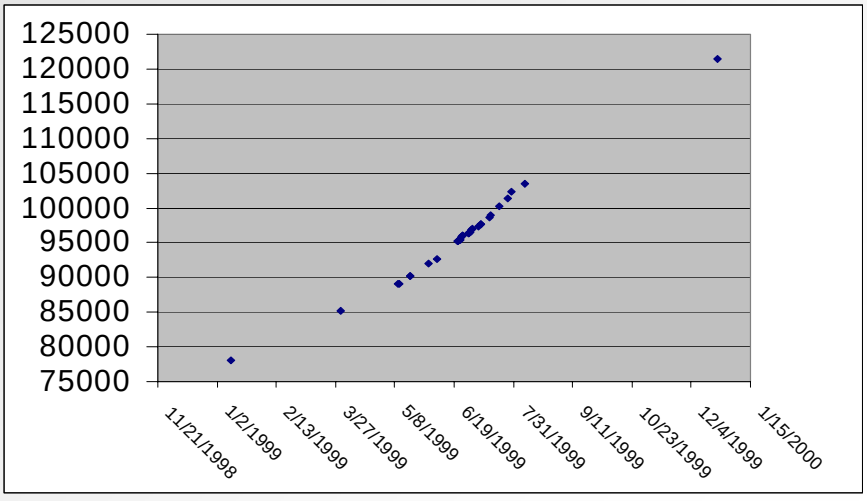
James Bach

Satisfice, Inc.

www.satisfice.com

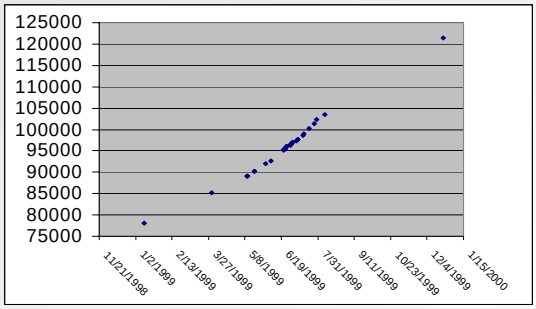
Smoking Gun Graph #1

Defendant sent me only 32 bugs from database…
so I graphed ID# vs. Date



Smoking Gun Graph #1

Defendant sent me only 32 bugs from database…
so I graphed ID# vs. Date

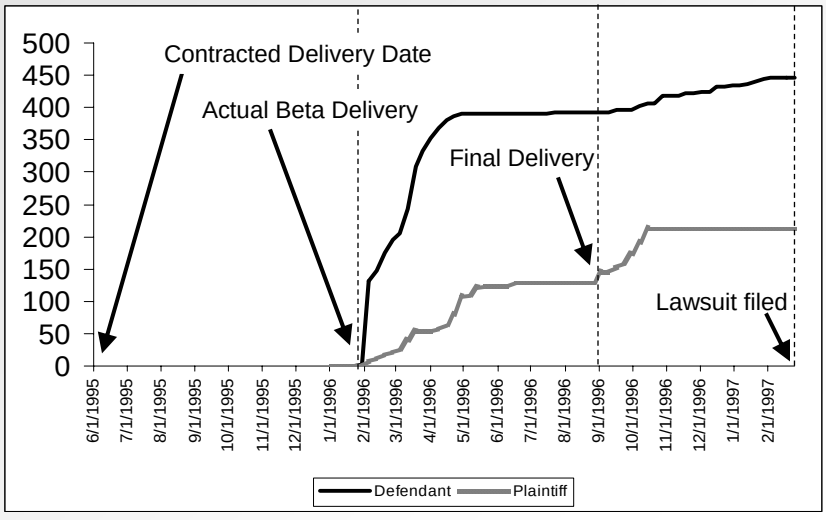


Peak find rate
185 bugs per day!

43,000 bugs reported during the project!

Smoking Gun Graph #2

Reported vs. Reported By Over Time
Portrait of a Sloppy Project



The Story of the Numbers
Supported by depositions and project documents

- The dates on the bugs indicate that bug tracking began in earnest in January of 1996, *seven months after the product was supposed to be completed*.
- Defendant released to Plaintiff at about the same moment they started testing it. *They did not test it before delivery*.
- Defendent performed serious testing for only two months, in February and March of ’96. *Then testing ground to a halt.*
- Throughout the summer, they didn’t work on the project (at least, they didn’t test it and track bugs on it).
- When they restarted the project, they apparently delivered to Plaintiff without testing it first. Plaintiff then began reporting bugs at a high rate until they apparently gave up.

Why Care About Metrics?

- You want to control your projects.
- You want to predict what will happen next.
- **You have questions about your projects.**
- You don’t want to be a victim of bad metrics.

What is a Bug?

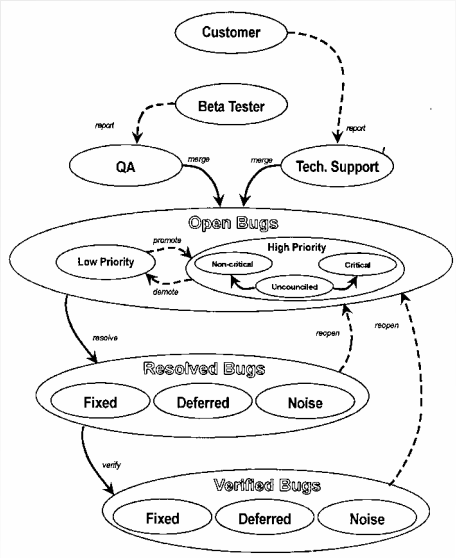
Quality is value to some person.

A ***bug*** is anything that threatens the value of the product.

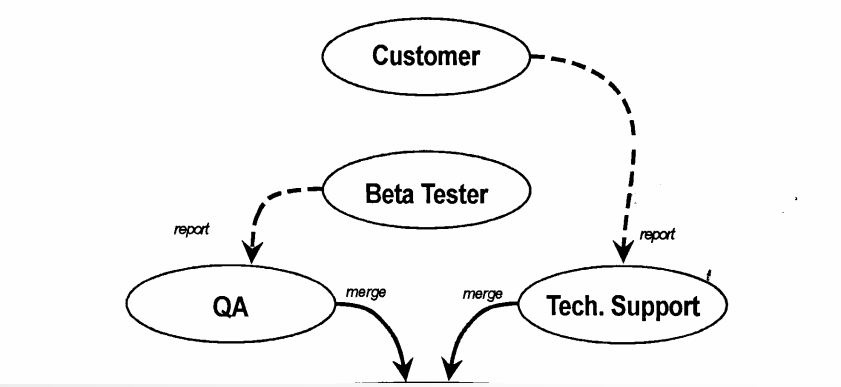
- These definitions are designed to be inclusive.
- Inclusive definitions minimize the chance that you will inadvertently overlook an important problem.
- "Enhancement requests" are bugs, too, but we often filter them out of the metrics.

Bug Cycle

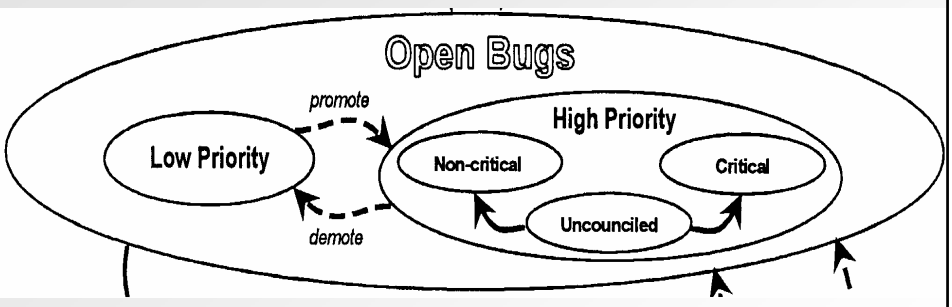
- This is my favorite bug lifecycle concept.
- Testers drive the cycle and assure high quality data in the system.



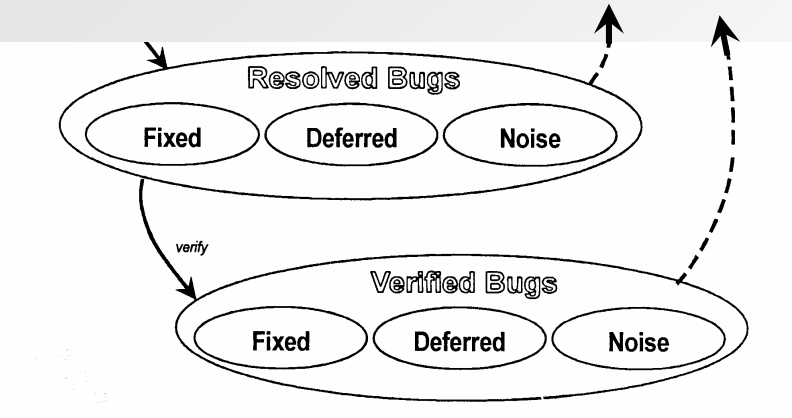
Reporting and Filtering



Triage



Resolution and Verification

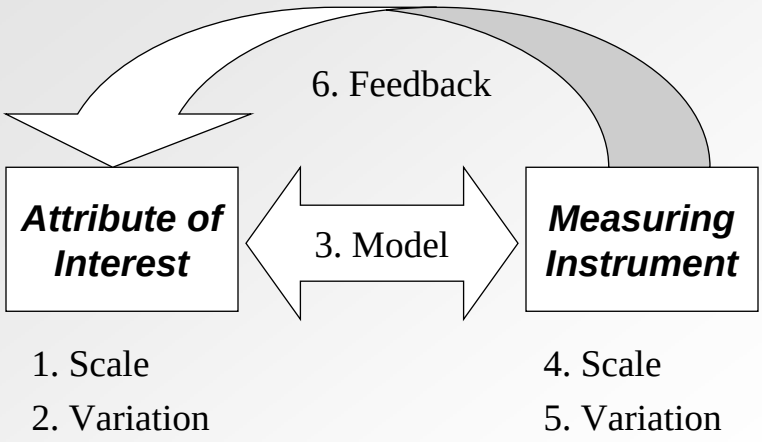


What is a Metric?

- "Measurement is the empirical, objective assignment of numbers, according to a rule derived from a model or theory, to attributes of objects or events with the intent of describing them." – *Cem Kaner*

- A *metric* is a measurement function– the mechanism by which we map a particular attribute to a particular scale.

What is a Metric?



Avoid Control Metrics; Embrace Inquiry Metrics

- A control metric is any metric that drives decisions.
  - *Any metric you use to control a self-aware system will be used by that system to control YOU.*
  - *A metric, by definition, is a simplification of reality, and as such, the same number can represent different realities.*
- An inquiry metric is any metric that helps you ask the right questions at the right time.
  - *An inquiry metric might look like a control metric. The difference is how you use it and what you infer from it.*
  - *Inquiry metrics are also vulnerable to gaming, but the stakes are far lower, so there's less incentive for manipulation.*

Control Metrics Cause Dysfunction in a Self-Aware System

1. **Goal:** You decide what you want.
2. **Question:** You conceive of questions that will reveal if you're getting what you want.
3. **Metric:** You create metrics that tell you if you're getting what you want.
4. **Control:** You make adjustments to the process until you get what you want.

WARNING: If the system is self-aware, then it will control YOU via your own metrics.

Inquiry Metrics Invite Learning

1. **Observe:** You try to see what's happening.
2. **Inquire:** You conjecture about the meaning and significance of the observations. You collect additional observations to corroborate or refute our conjectures.
3. **Model:** You form and improve theories about why the system behaves as it does, how you know that it behaves that way, and what you can do about it.

If you have an inquiry mindset, you can get use out of even very questionable metrics.

Example Control Metrics

- "The developer who creates the fewest bugs will receive a bonus."
- "Testers must average at least three bugs found per day."
- "The product may not be released unless we've gone at least one week without finding a bug."
- "The product may not be released with more than 10 high severity bugs."

Example Inquiry Metrics

- "Why do some developers have more bugs reported on their code than others? How might it be *good* to have more bugs reported? How might it be *bad*?"
- "How do find rates differ among testers? What makes them differ?"
- "What does the bug find rate tell us about the readiness of the product? Could the find rate be falling because testing is slack, and not because the product is good?"
- "What are the high severity bugs? Should we fix them?"

Data Collection Principles

- Any data not routinely used will be *routinely corrupted*.
- Any data collection to be remembered will be *routinely forgotten*.
- Any data expensive to collect will *routinely ignored*.
- Any data that carries a penalty will be *routinely suppressed*.

therefore…

- Establish a corruption test for data you collect and don't use.
- Automate collection so that it doesn't depend on human memory; set default values to blank.
- Minimize the impact of data collection on people; don't try to measure every little thing.
- Use multiple sources for data; establish corruption tests; stop using metrics for control purposes.

Data Inquiry Ideas

- **Look For…**
  - Continuities
  - Discontinuities
  - Extremes
  - Profiles
  - Contradictions
- **Read…**
  - *How to Lie With Statistics*
  - *How to Lie With Charts*
  - *Why Does Software Cost So Much?*
  - *Measuring and Managing Performance in Organizations*
- **Beware of…**
  - Extrapolating from meager data
  - Extrapolating non-linear phenomena
  - Suspiciously expected data
  - Baselines that keep changing
  - Measuring individual people

Questions Bug Metrics Might Help With ***Test Process***

- Productivity
  - Are the testers working at full speed?
  - Are they speeding up or slowing down?
  - Is anything blocking their work?
  - Are the testers burning out or staying sharp?
  - Are testers and developers getting along together?
- Status
  - How close is the test effort to reaching a point of diminishing returns?
  - What kinds of problems are being found?
  - Is the bug tracking system being used properly?
- Focus
  - Are the testers focused on the areas of greatest risk or need?
  - Could the testers use some outside help?

Questions Bug Metrics Might Help With ***Quality Improvement***

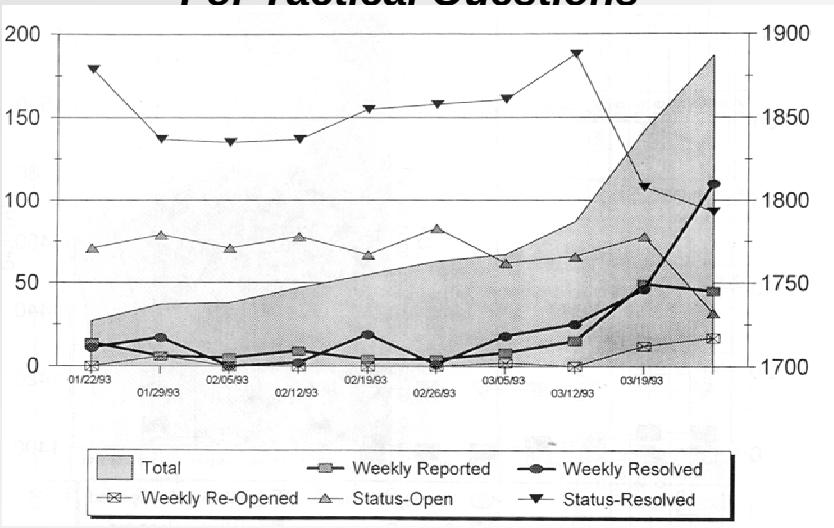
- Productivity
  - Are the developers working at full speed?
  - Are they speeding up or slowing down?
  - Is anything blocking their work?
- Status
  - Is the product improving?
  - How close is the product to being good enough?
  - What must happen in order to meet the schedule?
- Focus
  - Is the work distributed effectively among the developers?
  - Do any areas of the product need more attention to improve more quickly?
  - Is the triage process working well?

Bugs Are a Surrogate Metric

**"If you can't be with the one you love,
love the one you're with."**

- A surrogate metric is used when you can't measure an attribute you care about. Instead, you measure something *you believe or hope is correlated* to that attribute.
- We can't measure how our products will actually behave in the field, nor what our customers will think of them.
- Laboratory testing *(not in the field)* by testers *(not the actual users)* using test data *(not actual user data)* with an unfinished product *(not the actual finished product)* is a **surrogate** measure of quality.

Weekly Bug Metrics are Nearly Useless
For Tactical Questions



Daily Metrics!

- Consider producing a bug metric summary:
  - Daily
  - Automatic
  - Archived
  - Plain text
- This becomes a resource for retrospective analysis.
- Additionally, daily graphs can help you notice fast breaking dynamics.
- Include weekends in the graphs.

My Favorite Bug Fields For Metrics

- Status
- Date Reported
- Who Found
- Assigned To
- Program Area
- Severity
- Priority
- **Triage Status**
- Resolution

Make sure you have an audit trail!

Many other fields might be cool to add, but you have to weigh the cost and consider likely inaccuracies of the data.

Severity:
Garden Path Model

- **A:** Crash or Data Loss
- **B:** Failure, No Workaround
- **C:** Failure, Obvious Workaround
- **D:** Minor Problem
- **E:** Enhancement Request

This scale is based mostly on the symptoms of the problem Not the actual severity.

Priority:
When to Fix

High Priority
- **1:** Fix quickly
- **2:** Fix later

Low Priority
- **3:** Fix before release
- **4:** Deferral candidate

I recommend that the testers set this field first, then developers and managers can modify

Resolution:
Fixed, Deferred, or Just Noise

- **Fixed:** The product was changed in some way.
- **Deferred:** It's fixable, but we won't fix it now.

noise

- **Cannot Reproduce:** After reasonable investigation, the developer has been unable to see the problem.
- **Duplicate:** It's already been reported.
- **As Designed:** The "problem" is intentional.
- **Tester Error:** The problem didn't happen.

***Cannot Reproduce* is only temporary noise.**
It represents a mystery.
Ignore mystery bugs at your *peril!*

Open Priority/Severity Crosstab
Indicators of quality

*** Open Summary (by Priority and Category):

	1	2	3	4	TOTAL	TOTAL%
A	49	15	20	0	84	8.76%
B	59	172	78	0	309	32.22%
C	29	204	73	0	306	31.91%
D	16	126	36	0	178	18.56%
E	8	14	57	1	80	8.34%
TOTAL	161	531	264	1	957	99.79%
TOTAL%	16.79%	55.37%	27.53%	.10%	99.79%	

- Percentages are calculated base on total bugs in database (to help catch misfiled bugs).
- In a healthy project, I'd expect some low severity bugs to be high priority.
- Compare the profile over time and across projects.

15-Day Resolution/Severity Crosstab
Indicators of quality

*** 15-day Resolution Profile (by Category):

	A	B	C	D	E	TOTAL	TOTAL%
Fixed	50	78	49	37	5	219	58.24%
Deferred	3	16	10	2	23	54	14.36%
Cannot Reproduce	3	3	4	4	1	15	3.99%
As Designed	1	6	16	6	7	36	9.57%
Test Case Error	0	15	2	3	1	21	5.59%
Duplicate	1	7	12	6	1	27	7.18%
TOTAL	58	125	93	58	38	372	98.94%
TOTAL%	15.43%	33.24%	24.73%	15.43%	10.11%	98.94%	

- Look for the deferral/fix ratio (among non-E bugs).
- Look for A's and B's that are not reproduced.
- Compare the profile over time and across projects.
- I use 15-day snapshots to smooth out weekend effects.

Verified Resolution/Severity Crosstab
Indicators of quality

*** All-Verified Resolution Profile (by Category):

	A	B	C	D	E	TOTAL	TOTAL%
Fixed	1279	1995	1316	813	262	5665	61.88%
Deferred	66	214	151	126	422	979	10.69%
Cannot Reproduce	257	244	142	83	9	735	8.03%
As Designed	35	224	179	76	158	672	7.34%
Test Case Error	107	249	92	35	25	508	5.55%
Duplicate	108	161	131	52	33	485	5.30%
TOTAL	1852	3087	2011	1185	909	9044	98.79%
TOTAL%	20.23%	33.72%	21.97%	12.94%	9.93%	98.79%	

- Compare with the 15-day snapshot.

Reopen Crosstab
Communication, care, or controversy?

*** Reopened Summary (by Priority and Category):

	1	2	3	4	5	6	TOTAL	TOTAL%
A	10	18	4	0	0	0	32	17.98%
B	3	45	36	0	0	2	86	48.31%
C	0	11	13	1	0	0	25	14.04%
D	0	2	5	2	1	0	10	5.62%
E	0	3	4	0	0	13	20	11.24%
TOTAL	13	79	62	3	1	15	173	97.19%

- Compare crosstab of open bugs.
- We'd expect the reopen profile to match the opens, if mistakes are distributed equally, and testing attention is evenly applied.
- (This graph is from a project that used 6 priority levels.)

15-day Throughput Stats
Indicators of Effort

- On large projects this is pretty interesting.

```
*** Overall 15-day Find Rate:

  28.07 average per day
   87   on the best day
    7   found today
  421   15-day total

*** Overall 15-day Resolve Rate:

  25.07 average per day
   56   on the best day
   26   resolved today
  376   15-day total

*** Overall 15-day Verification Rate:

  23.20 average per day
   57   on the best day
   21   verified today
  348   15-day total
```

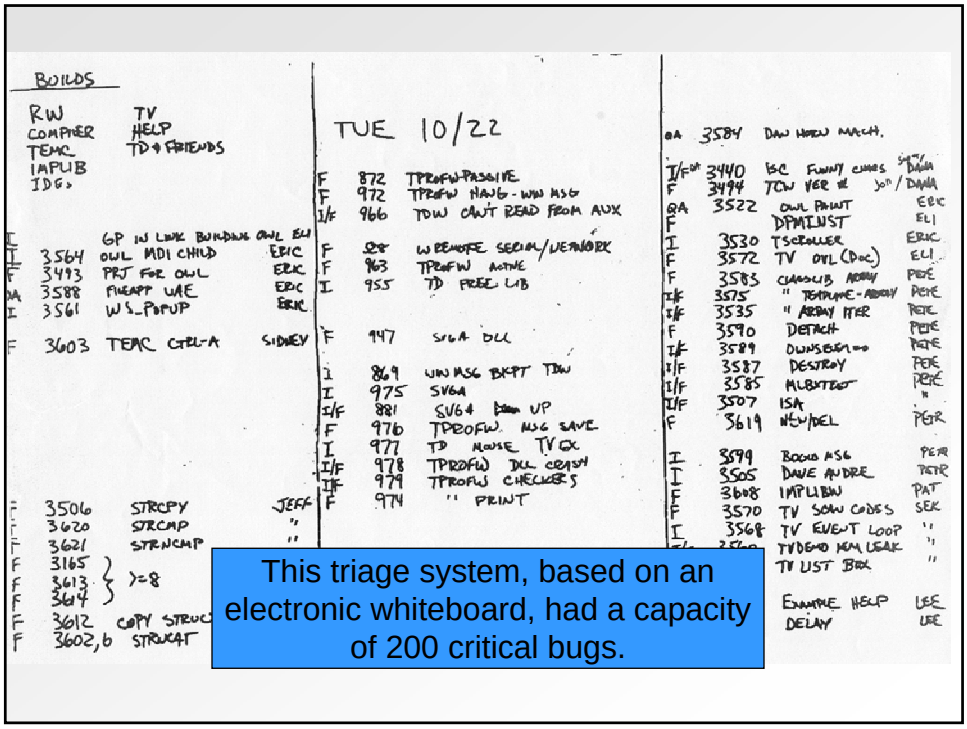
Other Interesting Bug Metrics

- **Priority 1 Find Rate**
  - This is the find rate of bugs that at any point in time become priority 1. It is a retrospective analysis.
- **Open Bug Aging Report**
  - A histogram of how long bugs have been open. This is most useful early in long projects.
- **Deferred Bug Aging Report**
  - A histogram of how long deferred bugs have been waiting on the list. It helps spot chronically deferred bugs.
- **Promotions/Demotions/Deferrals Chart**
  - Three lines that help us see the triage process at work.

Triage:
Change Control Late in the Project

- **<blank>:** Not reviewed
- **F:** Approved for fix
- **I:** Developer must investigate and report
- **Q:** Tester must investigate and report
- **R:** Document in “readme”
- **D:** Document in help/manual

My definition of a *critical bug*:
A triaged bug with a priority of 1 or 2



Critical Bugs/Product Area Crosstab

to do list for release

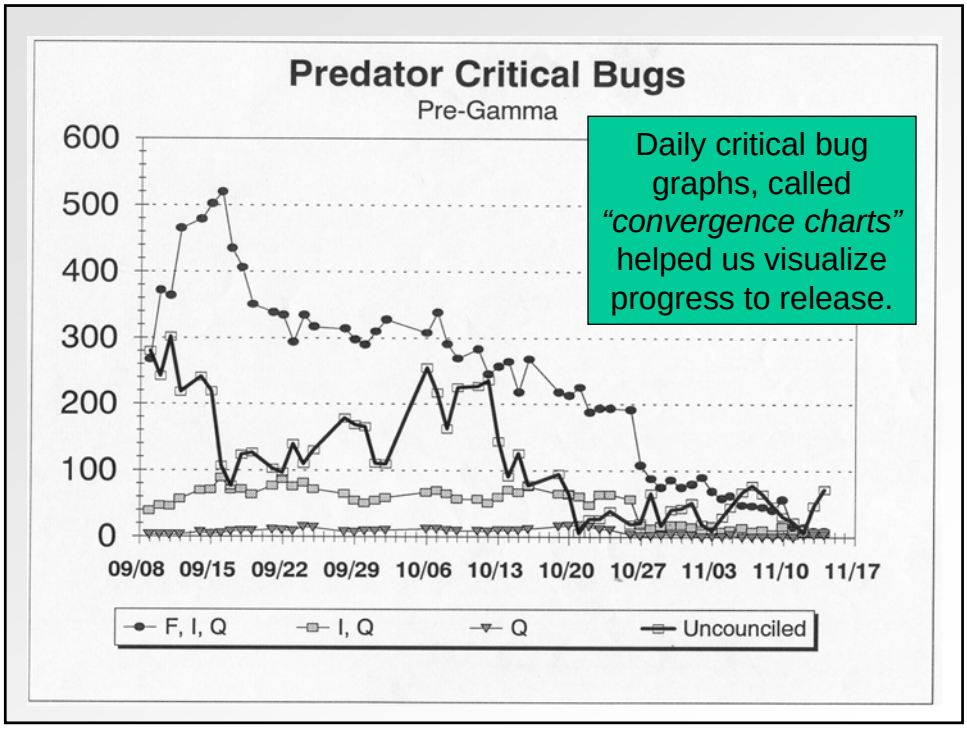
*** Critical (by Product):

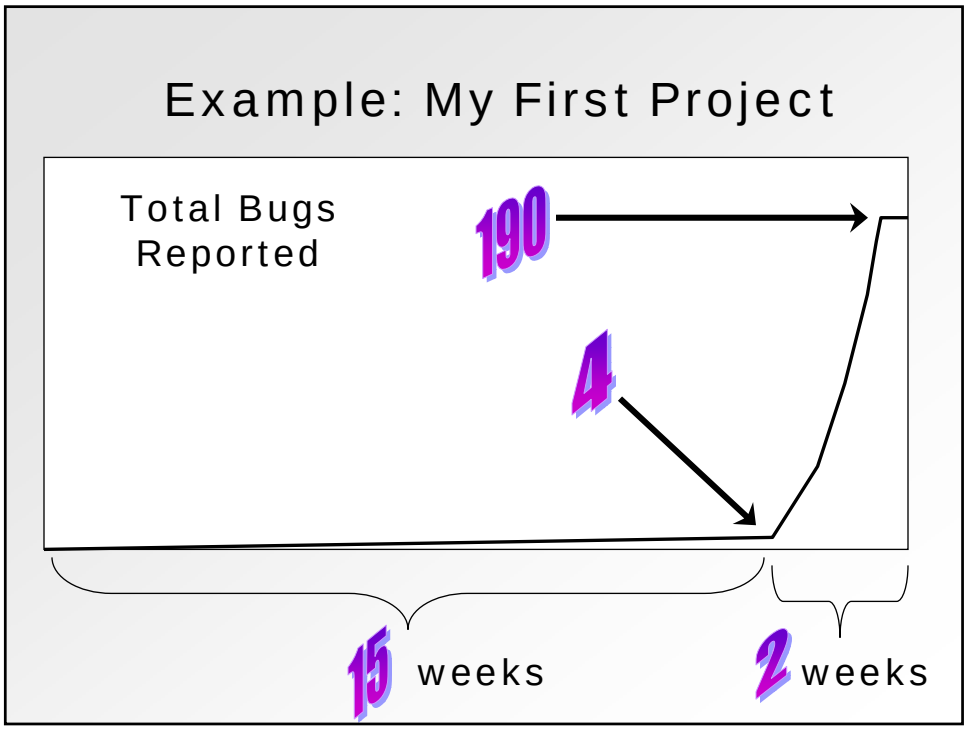
	F	I	Q	TOTAL	TOTAL%
BCW	296	43	2	341	93.68%
B40	22	0	0	22	6.04%
BRC	0	0	0	0	0.00%
RW	0	0	1	1	.27%
32TD	0	0	0	0	0.00%
TDL	0	0	0	0	0.00%
TOTAL	318	43	3	364	100.00%
TOTAL%	87.36%	11.81%	.82%	100.00%	

*** Uncounciled (by Product & Priority):

	1	2	TOTAL	TOTAL%
BCW	19	76	95	31.46%
B40	57	131	188	62.25%
BRC	0	6	6	1.99%
RW	0	9	9	2.98%
32TD	0	1	1	.33%
TDL	1	2	3	.99%
TOTAL	77	225	302	100.00%
TOTAL%	25.50%	74.50%	100.00%	

Triage from a database changed our perceptions of quality.


Predator Critical Bugs
Pre-Gamma
600
500
400
300
200
100
0
09/08 09/15 09/22 09/29 10/06 10/13 10/20 10/27 11/03 11/10 11/17
F, I, Q
I, Q
Q
Uncounciled
Daily critical bug graphs, called *"convergence charts"* helped us visualize progress to release.


Example: My First Project
Total Bugs Reported
190
4
15 weeks
2 weeks

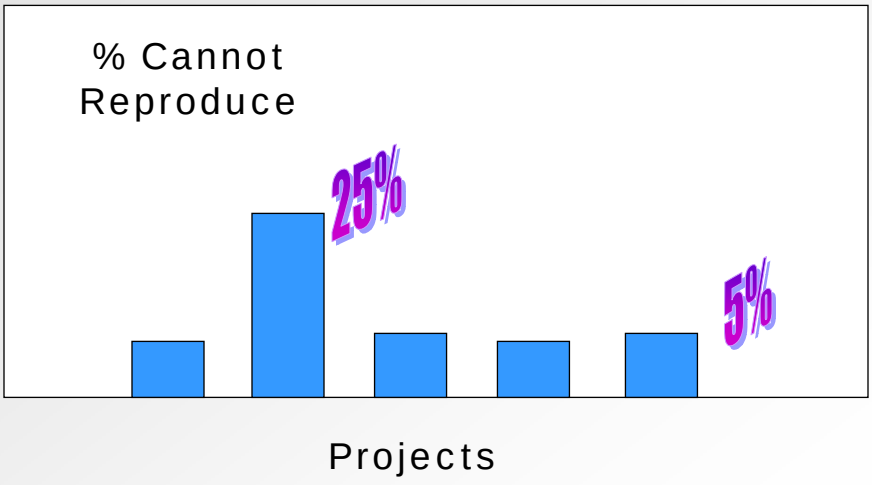
Example: My First Project

- The original tester was faking it.
- We used a tiger team in the last 2 weeks.
- The reporting system broke down in the last several days before we shipped the beta. That's why the line goes flat– but reports were still made by hand.

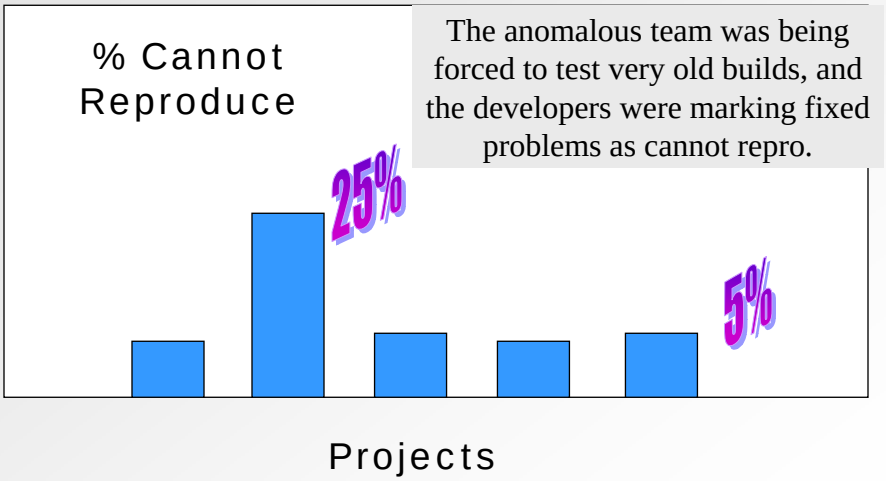
Lesson:

- **Pay attention.**
- **Don't assume that all known problems are being reported.**

Example: Anomalies #1

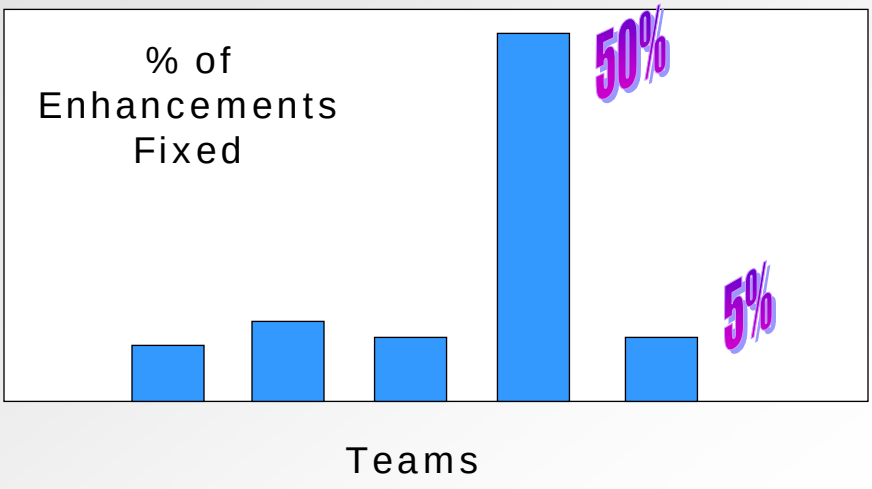


Example: Anomalies #1



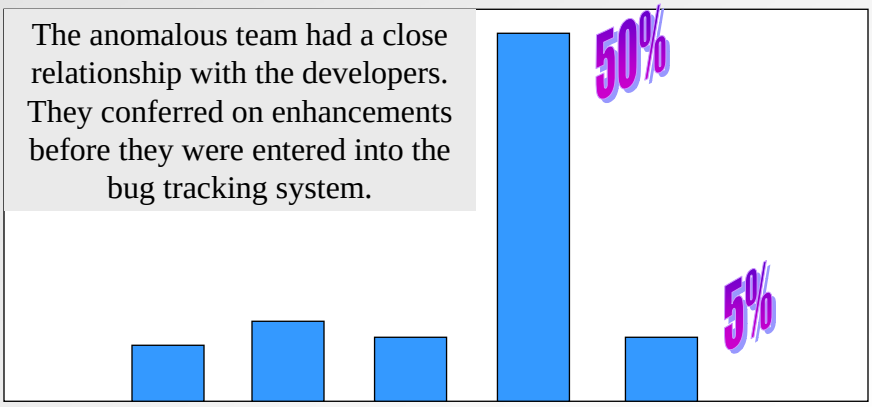
The anomalous team was being forced to test very old builds, and the developers were marking fixed problems as cannot repro.

Example: Anomalies #2



Example: Anomalies #2

The anomalous team had a close relationship with the developers. They conferred on enhancements before they were entered into the bug tracking system.



Teams

Example: Anomalies

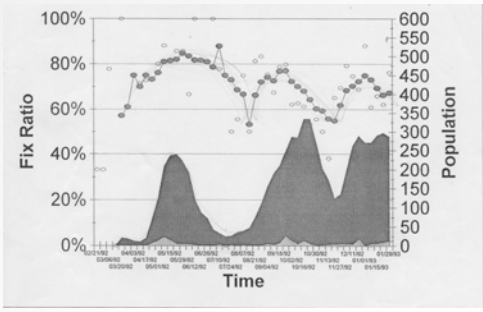
Lesson:

Metrics will vary from team to team due to all kinds of process and relationship differences.

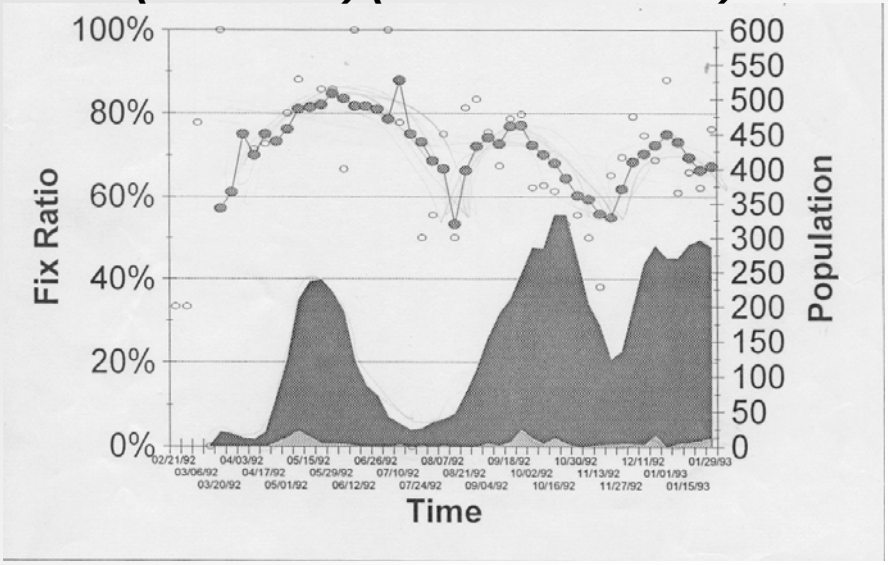
An inconsistency just means there may be something interesting to investigate.

EXAMPLE: Communication Quality
(Fix+Defer)/(Fix+Defer+Noise)

- Hypothesis: **Communication problems lead to noise bugs.**
- I plotted the proportion of noise bugs, not the absolute number, so that the amount of bug resolution activity would not confuse the issue.
- Since proportions vary widely with population, I also graphed the total population of resolved bugs.
- **I concluded that this metric is misleading and unusable. Why?**



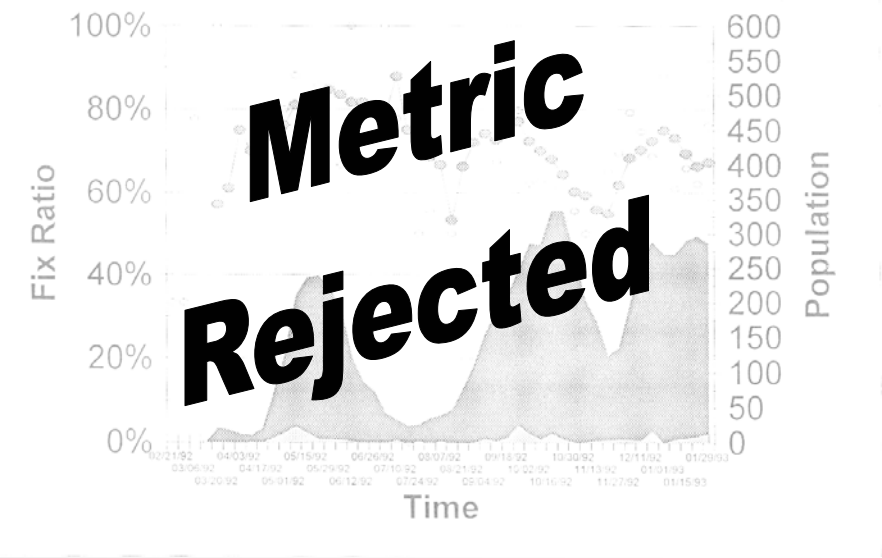
EXAMPLE: Communication Quality
(Fix+Defer)/(Fix+Defer+Noise)



EXAMPLE: Communication Quality
(Fix+Defer)/(Fix+Defer+Noise)

- Bugs are resolved over a period of time, sometimes long after they are reported.
- But I wanted to know the communication quality *at the time bugs are reported*.
- By looking at individual resolutions and talking to developers, I discovered the reason fix ratios fall during periods when few bugs are resolved: *low hanging fruit*.
- When developers can't fix bugs quickly, they try at least cast out the obvious noise bugs. That causes the fix ratio to artificially rise, later on.
- To screen out that effect, I need to measure, for the bugs reported on a given day, what the *eventual* resolution will be for those specific bugs. That takes too long to unfold.

EXAMPLE: Communication Quality
(Fix+Defer)/(Fix+Defer+Noise)



EXAMPLE: Communication Quality

(Fix+Defer)/(Fix+Defer+Noise)

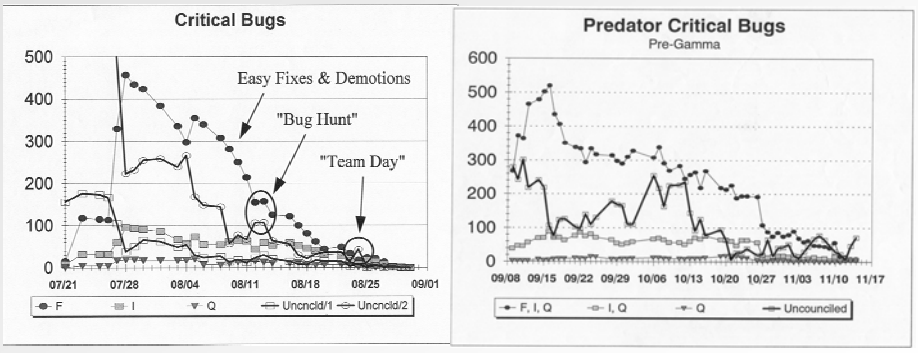
Lesson:

Be skeptical about your metrics. Many good sounding ideas lead to nonsense numbers.

When your attitude is inquiry, rather than control, metrics "failures" like this are no problem.

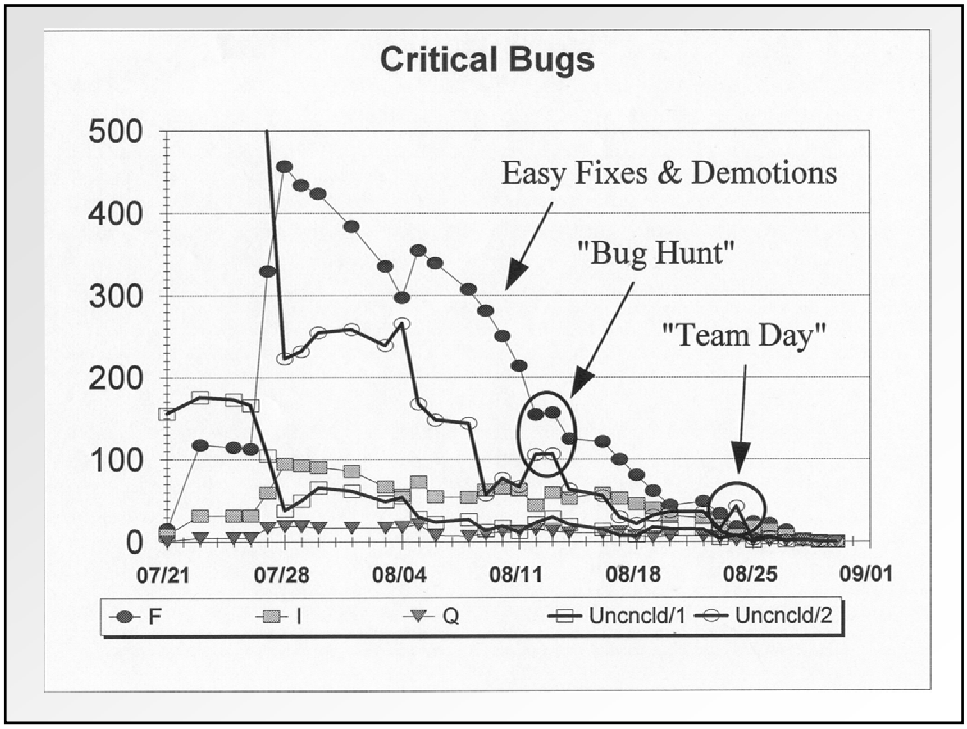
EXAMPLE:

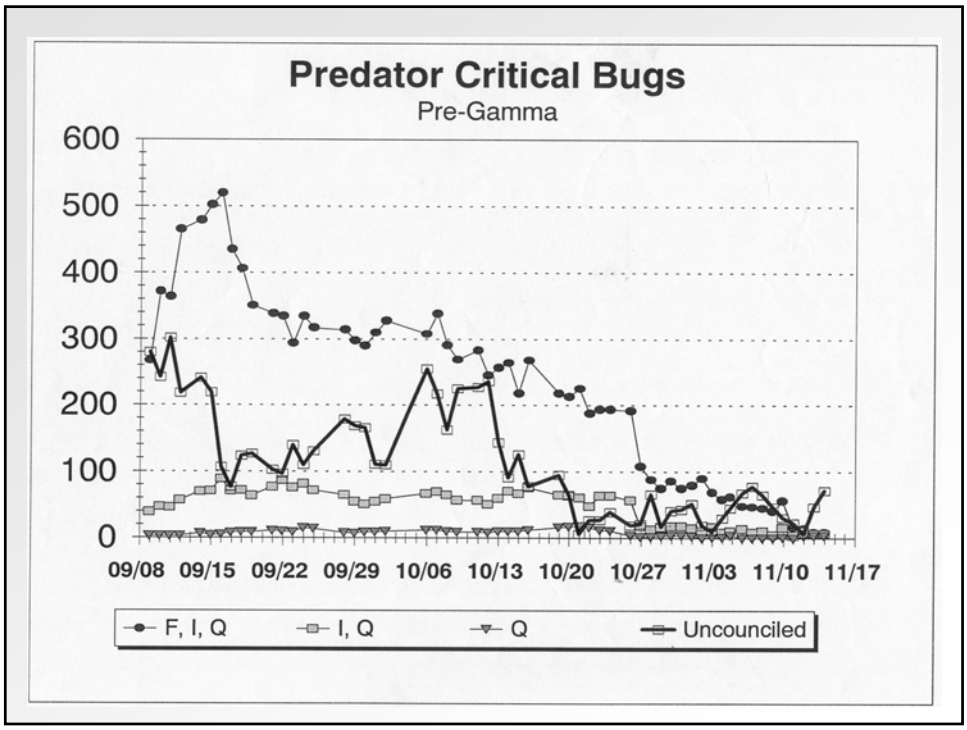
Why was convergence harder during the second cycle?



Beta Cycle

Final Cycle

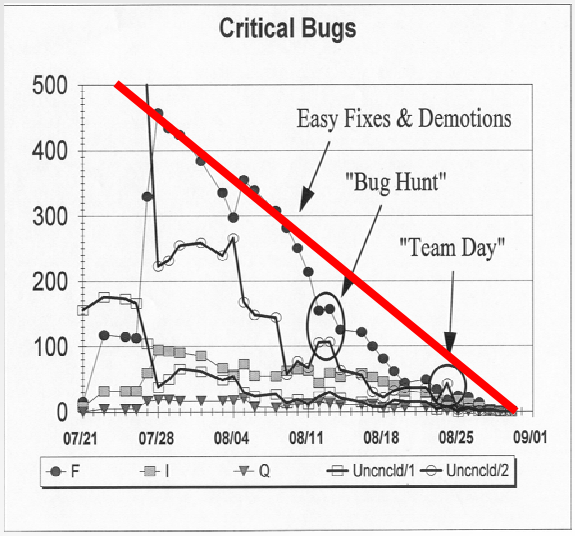

Critical Bugs
500
400
300
200
100
0
Easy Fixes & Demotions
"Bug Hunt"
"Team Day"
07/21
07/28
08/04
08/11
08/18
08/25
09/01
F
I
Q
Uncncld/1
Uncncld/2

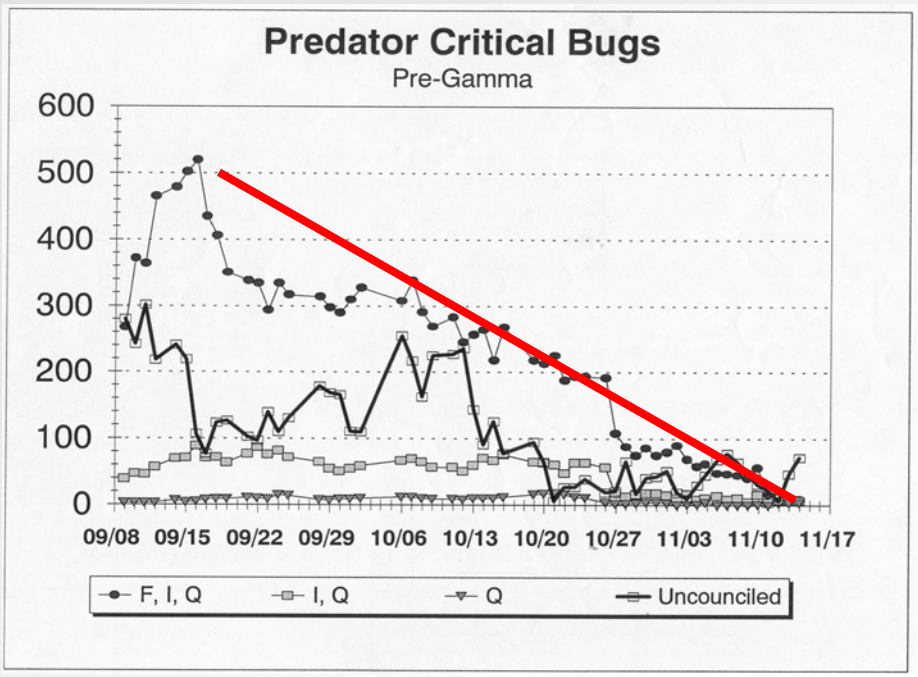

Predator Critical Bugs
Pre-Gamma
600
500
400
300
200
100
0
09/08
09/15
09/22
09/29
10/06
10/13
10/20
10/27
11/03
11/10
11/17
F, I, Q
I, Q
Q
Uncounciled

Process Notes Help Interpretation

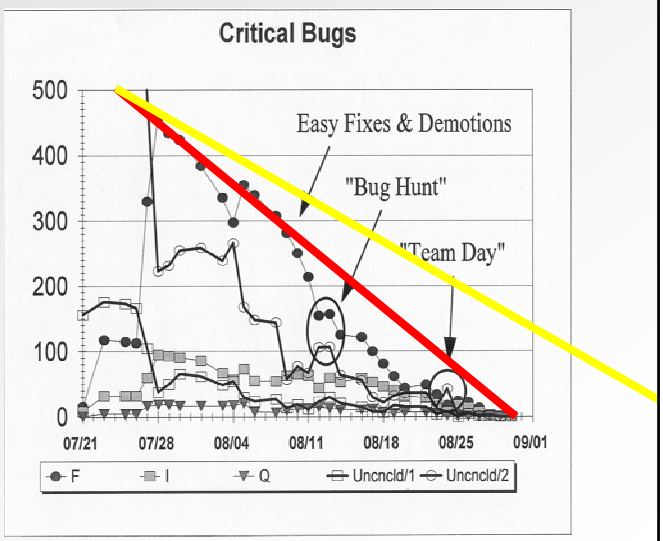
On 7/15, the official date was moved from 7/30 to 8/13.
On 7/20, I predicted that Beta 2 would ship on 8/24. The official date was 8/13.
On 7/27 a determination was made that we are not converging quickly enough. We will attempt to bring in the date b
On 7/27, the target was set vaguely at the end of the month
On 8/13, the bug hunt spike hit.
On 8/13, the ship date planned in July, there were still 150 critical bugs.
On 8/16, the first full council was held for the basic tools and OWL.
On 8/20, a half day was lost due to the beach party
On 8/23 the first team day spike hit.
On 8/24, team day bugs were still coming in.
On 8/24, we met and delayed ship from 8/27 to 8/30 or 8/31.
On 8/24, final signoff procedures were examined
On 8/27, Spencer thought we might be able to signoff on Saturday, 8/28
On 8/28, we worked all day, but decided at 5:30 that we wouldn't sign off that night.
On 8/29, we had to rebuild and recut for Frodo problems.
On 8/29, many verifications were done without updating the BTS.
On 8/30, we discovered that Classlibs had been built wrong... another recut.

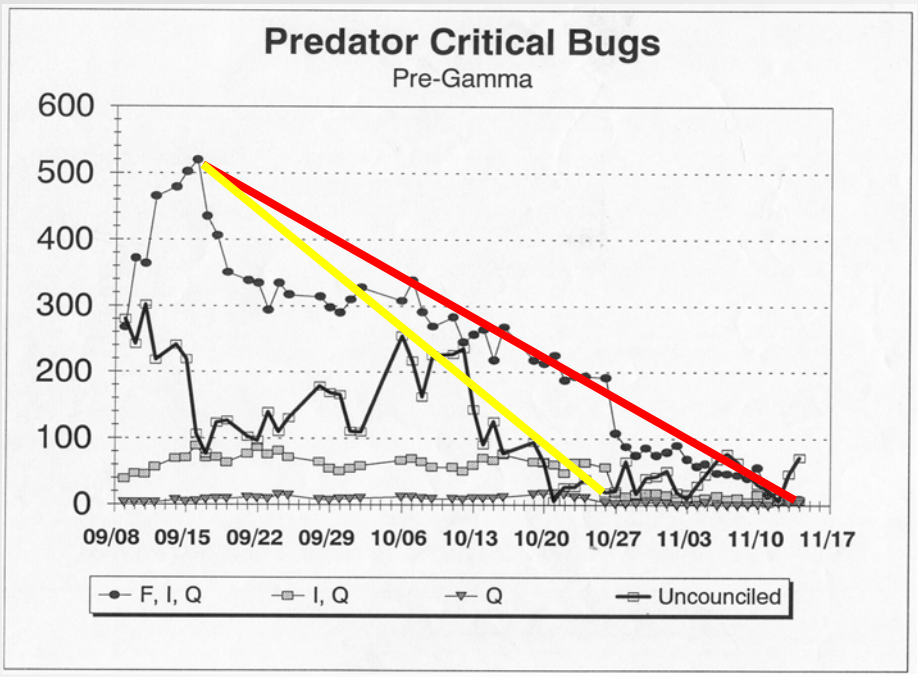
To Compare Slopes, Normalize Scales





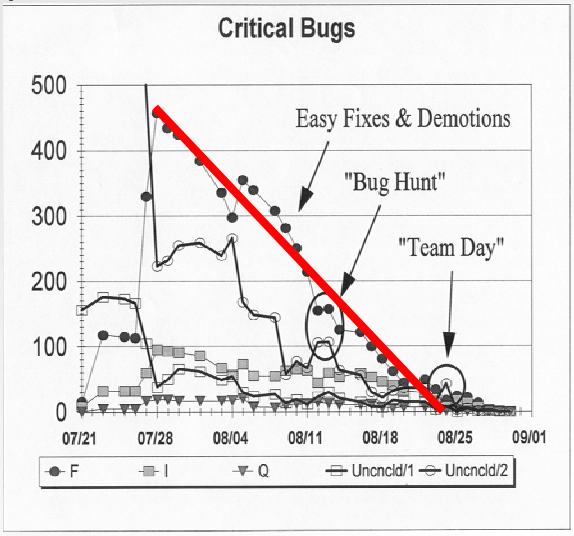
To Compare Slopes, Normalize Scales

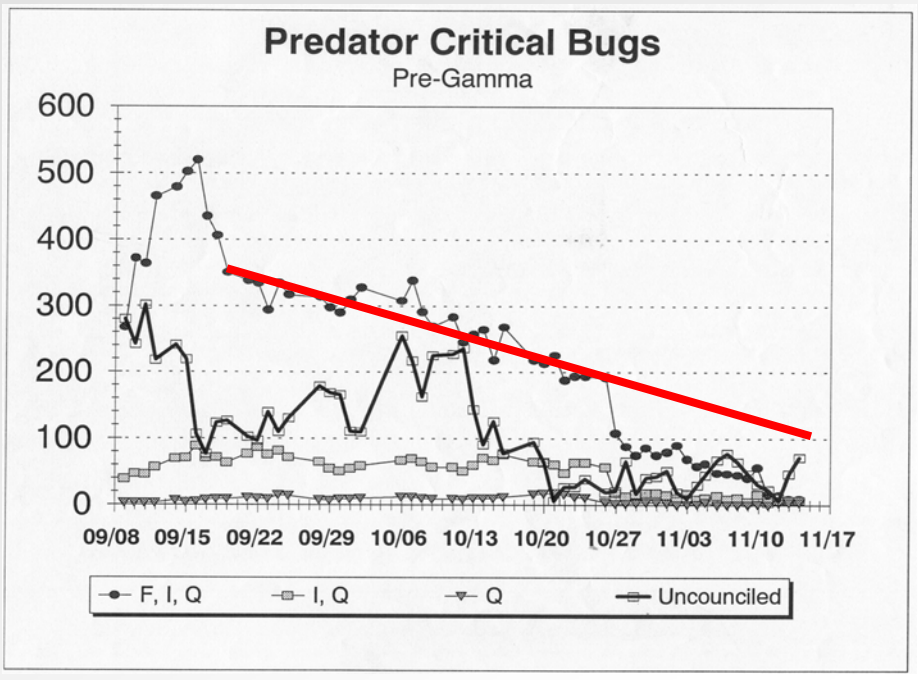


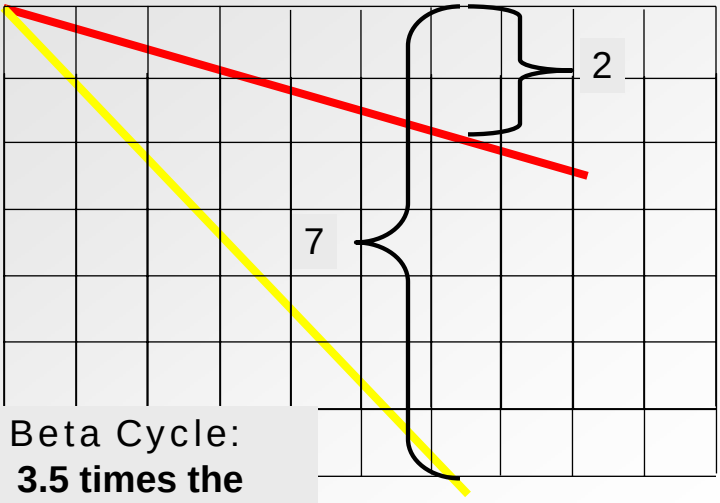


Focus the question on what matters.

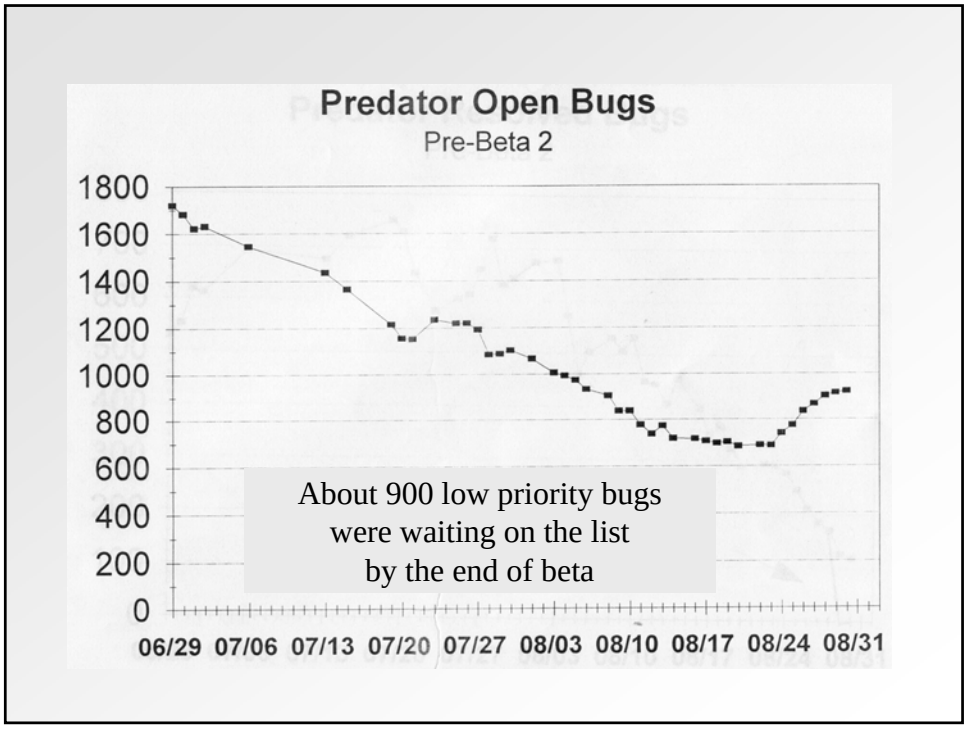
What I really care about is why the majority of the first cycle was nicely steep, whereas the majority of the second cycle was pretty flat.

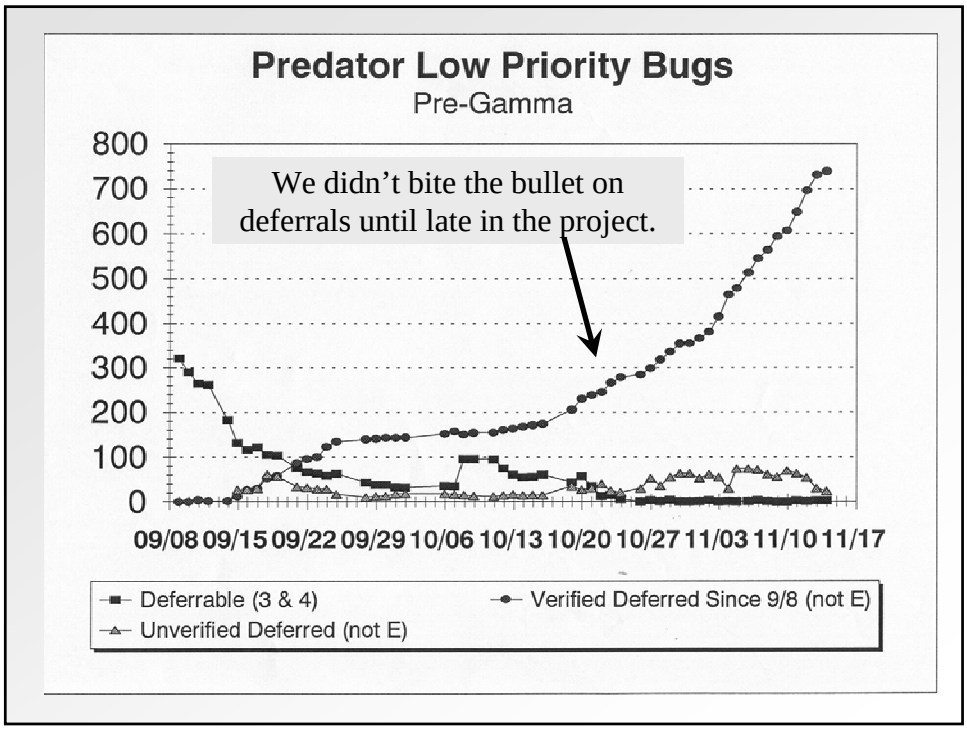






Beta Cycle:
3.5 times the rate of convergence


Predator Open Bugs
Pre-Beta 2
1800
1600
1400
1200
1000
800
600
400
200
0
06/29 07/06 07/13 07/20 07/27 08/03 08/10 08/17 08/24 08/31
About 900 low priority bugs
were waiting on the list
by the end of beta


Predator Low Priority Bugs
Pre-Gamma
800
700
600
500
400
300
200
100
0
09/08 09/15 09/22 09/29 10/06 10/13 10/20 10/27 11/03 11/10 11/17
We didn't bite the bullet on
deferrals until late in the project.
Deferrable (3 & 4)
Verified Deferred Since 9/8 (not E)
Unverified Deferred (not E)

EXAMPLE:

Why was convergence harder during the second cycle?

Lesson:

Beware of bugs hidden in plain sight. Monitor a set of charts that account for *every* bug.

(And beware of bugs that people don't officially report, because they want your charts to be look better.)

SkyWalker Bug Analysis

```
*****    SkyWalker Bug Summary    *****
4/26/94, 08:18:44

*** Critical (by Product):

              F       I       Q   TOTAL   TOTAL%
       ----------------------------------------
BR           419      23      33     475   98.55%
ODAP           5       1       1       7    1.45%
       ----------------------------------------
TOTAL        424      24      34     482  100.00%

*** Uncounciled (by Product and Priority):

              1       2   TOTAL   TOTAL%
       --------------------------------
BR            11      72      83   98.81%
ODAP           0       1       1    1.19%
       --------------------------------
TOTAL         11      73      84  100.00%

*** Open Summary (by Priority and Category):

              1       2       3       4       5       6   TOTAL   TOTAL%
       ------------------------------------------------------------------
A             31      97      65       1       0       0     194    9.76%
B             41     430     531      10       1      11    1024   51.53%
C              9      63     282      62       0      11     427   21.49%
D              2      28      83      59       3       0     175    8.81%
E              1      22      14       2       0      90     129    6.49%
       ------------------------------------------------------------------
TOTAL         84     640     975     134       4     112    1949   98.09%

*** Resolved Summary (by Priority and Category):

              1       2       3       4       5       6   TOTAL   TOTAL%
       ------------------------------------------------------------------
A             34      84      23       0       0       0     141   16.65%
B             36     296      82       2       7       4     427   50.41%
C              5      59      60      19       5       2     150   17.71%
D              2      25      35      15       0       1      78    9.21%
E              0       2       9       0       0      20      31    3.66%
       ------------------------------------------------------------------
TOTAL         77     466     209      36      12      27     827   97.64%

*** Reopened Summary (by Priority and Category):

              1       2       3       4       5       6   TOTAL   TOTAL%
       ------------------------------------------------------------------
A             10      18       4       0       0       0      32   17.98%
B              3      45      36       0       0       2      86   48.31%
C              0      11      13       1       0       0      25   14.04%
D              0       2       5       2       1       0      10    5.62%
E              0       3       4       0       0      13      20   11.24%
       ------------------------------------------------------------------
TOTAL         13      79      62       3       1      15     173   97.19%
```

This is a very high number of critical bugs. Our largest project on record had no more than this at it's peak.

This is also a very high number. About 600 category A & B bugs that are *not* considered high priority.

The quality standard is probably low if six hundred crashes and hard failures are priority 3.

There are more unverified bugs here than any other project at our company. Also, about 15% of these will be reopened by QA.

Again, this is a high number of reopened bugs. Each reopened bug represents disagreement or miscommunication between R&D and QA.

*** Verified Resolution Profile (by Category):

	A	B	C	D	E	TOTAL	TOTAL%
Fixed	1640	2892	808	449	154	5943	68.82%
Deferred	12	92	32	47	28	211	2.44%
Cannot Reproduce	498	519	165	154	4	1340	15.52%
As Designed	6	300	94	19	61	480	5.56%
Test Case Error	5	32	4	3	2	46	.53%
Duplicate	91	162	60	28	3	344	3.98%
TOTAL	2252	3997	1163	700	252	8364	96.85%

*** Resolved Resolution Profile (by Category):

	A	B	C	D	E	TOTAL	TOTAL%
Fixed	94	237	74	36	31	472	55.73%
Deferred	0	22	5	4	2	33	3.90%
Cannot Reproduce	24	58	14	14	0	110	12.99%
As Designed	2	46	31	6	6	91	10.74%
Test Case Error	0	1	4	2	0	7	.83%
Duplicate	11	34	9	6	0	60	7.08%
TOTAL	131	398	137	68	39	773	91.26%

*** 15-day Resolution Profile (by Category):

	A	B	C	D	E	TOTAL	TOTAL%
Fixed	154	472	119	88	25	858	48.61%
Deferred	1	90	30	48	17	186	10.54%
Cannot Reproduce	45	87	26	31	2	191	10.82%
As Designed	1	192	54	17	42	306	17.34%
Test Case Error	1	8	5	2	1	17	.96%
Duplicate	10	70	24	22	1	127	7.20%
TOTAL	212	919	258	208	88	1685	95.47%

*** 15-day Find Rate (by Date and Category):

	A	B	C	D	E	TOTAL	TOTAL%
4/12/94	13	25	11	7	1	57	7.72%
4/13/94	47	54	14	12	5	132	17.89%
4/14/94	11	38	8	9	2	68	9.21%
4/15/94	4	12	6	3	3	28	3.79%
4/16/94	4	11	0	2	2	19	2.57%
4/17/94	8	11	5	7	0	31	4.20%
4/18/94	12	39	10	10	0	71	9.62%
4/19/94	19	44	11	9	1	84	11.38%
4/20/94	10	43	9	1	0	63	8.54%
4/21/94	11	35	7	8	0	61	8.27%
4/22/94	11	30	6	1	0	48	6.50%
4/23/94	1	19	4	1	0	25	3.39%
4/24/94	1	0	2	0	0	3	.41%
4/25/94	7	31	6	4	0	48	6.50%
4/26/94	0	0	0	0	0	0	0.00%
TOTAL	159	392	99	74	14	738	100.00%

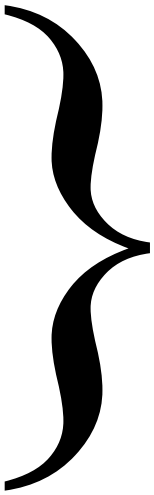
Notice how the fix percentage is steadily dropping and the deferral percentage is rising in each of these profiles.

This indicates that the pressure to ship has begun forcing bugs to be deferred.

This is a high and steady find rate, indicating that a lot of testing is happening and that a lot of problems are being found.

The fact that 75% of all bugs are severe (A&B) is troubling. In software that's nearing shippability, you expect to see that be no more than 50%. **This is evidence that "spit and polish" bugs are being ignored.**

```
*** Overall 15-day Find Rate:

  49.20 average per day
  132   on the best day
    0   found today
  738   15-day total
:::

*** Overall 15-day Resolve Rate:

 117.67 average per day
  189   on the best day
   21   resolved today
 1765   15-day total
:::

*** Overall 15-day Verification

  94.07 average per day
  193   on the best day
    2   verified today
 1411   15-day total
:::


*** Overall 15-day Reopen Rate:

  16.27 average per day
   38   on the best day
    0   reopened today
  244   15-day total
:::
```



This activity can only be described as “furious” The team is definitely going all out.

The reopen rate is 15% of the resolve rate, though. That’s evidence of poor communication or cooperation between R&D and QA. 5% is typical for other projects at our company.