PNSQC

OCTOBER 10-12 2022

Vittalkumar Mirajkar
VitalkumarrMirajkar@gmail.com

Defining data pointer for software testing efficiency measurement

Defining data pointer for software testing efficiency measurement

Problem statement:

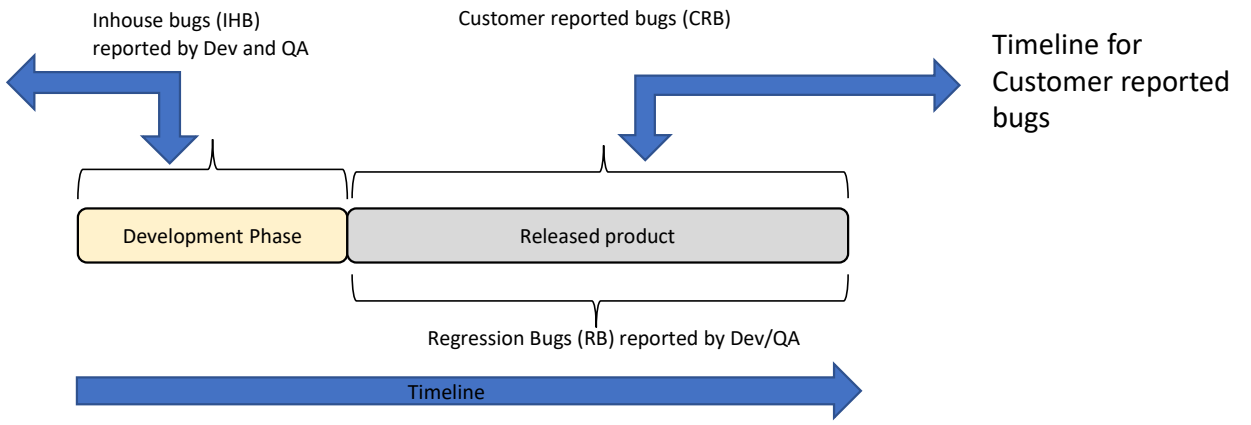
- **Measure how good current releases are in simple, quick and repeatable way**
- **Define actionable approach to reduce customer bugs and improve quality**

Why do we test software before release ?

- Release better product ? Better quality ? Achieve better user experience ?
  - and the list goes on..
- All the motivation can be summarized as "**To release better quality software to production** "

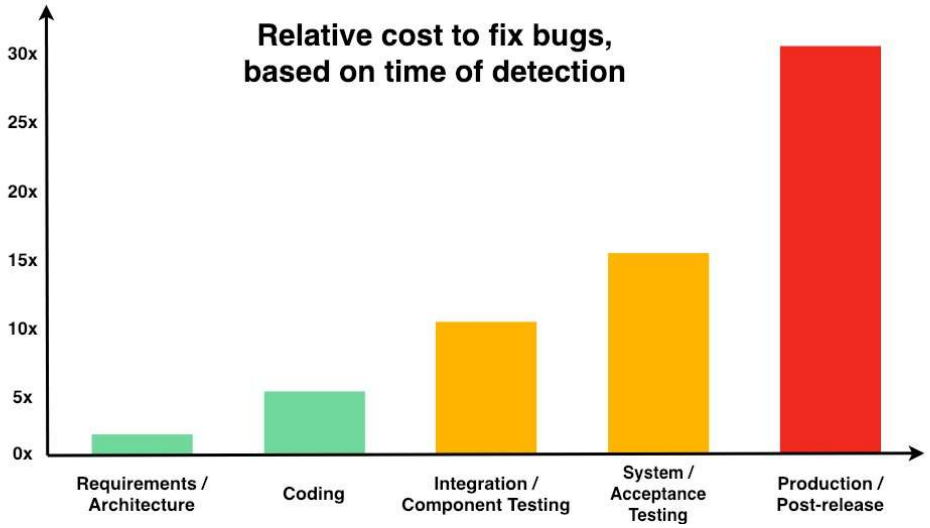
How do we measure software quality

- Unit test coverage
- Path coverage
- Requirements coverage
- Number of defects
- Percentage of automated test coverage



Cost of fixing a bug

Despite religiously following the quality metrics, production bugs are a living reality.



- Current measurements does not give real time data as to what will be the impact of uncovering potential productions bugs.
- To measure production bugs real impact, there is a need for post release impact measurement, which is a lagging indicator and not real time.

Quality Volatility

- In Derivative trading PUT and CALL open interests Ratio (aka PCR) PCR is widely used to forecast market direction with Put/Call ratios.
- PCR ratio between 0.10 to 10 determines the market Index range for the week.
- Below it the Open Interest Data for Nifty 50 for the current weekly expiry (10th Oct)

OI	CHNG IN OI	VOLUME	IV	LTP	CHNG	BID QTY	BID PRICE	ASK PRICE	ASK QTY	STRIKE PRICE	BID QTY	BID PRICE	ASK PRICE	ASK QTY	CHNG	LTP	IV	VOLUME	CHNG IN OI	OI	PCR
105	4	30	-	772.5	-67.5	1,500	737.2	795.3	150	16,550.00	550	5.35	5.65	200	-4.15	5.65	19.9	1,14,579	2,622	12,163	115.8380952
334	43	639	-	721.8	-10.65	5,950	711.8	745.1	450	16,600.00	250	6.65	6.95	1,850	-4.95	6.95	19.57	3,33,038	11,580	42,868	128.3473054
85	-1	83	-	678	-6.85	50	670.25	702.1	350	16,650.00	300	8.25	8.85	300	-5.55	8.8	19.35	1,13,401	6,670	10,553	124.1529412
798	71	1,509	-	627.9	-8.3	500	622.95	631.25	100	16,700.00	350	10.85	11.15	250	-6.65	10.85	19.03	4,00,984	20,703	48,964	61.35839599
200	3	332	-	574.8	-15.55	600	575.7	587.8	150	16,750.00	250	13.2	14.3	950	-6.5	14.35	19.03	1,64,053	11,467	16,422	82.11
1,564	386	5,858	-	534.9	-9.8	50	533	535	250	16,800.00	150	16.75	17.6	1,250	-9	16.75	18.47	5,73,600	33,041	64,555	41.27557545
187	48	463	-	484	-16.9	50	458.95	509.95	4,350	16,850.00	100	21.1	21.2	1,700	-9.95	21.1	18.29	1,98,915	5,030	9,843	52.63636364
1,588	14	7,399	-	441.65	-15.45	100	440.3	449	150	16,900.00	550	25.9	26.05	50	-11.25	25.9	17.99	5,66,470	12,904	41,881	26.37342569
858	533	3,725	12.68	402.45	-12.25	100	395.9	402.85	50	16,950.00	50	32.25	32.95	100	-10.7	34	18.13	2,54,873	7,678	12,150	14.16083916
12,141	-1,721	1,16,385	12.4	356.15	-16.15	300	355.7	359.75	200	17,000.00	100	39.55	40	350	-13.6	40	17.62	13,61,714	36,341	86,992	7.165142904
697	247	13,126	12.98	315.4	-17.8	11,650	310.1	344.3	12,150	17,050.00	150	49.05	50.9	250	-12.5	50.9	17.69	4,22,065	22,465	30,969	44.43185079
5,937	995	1,24,157	14.01	280.8	-14.4	150	276.75	280	250	17,100.00	750	60.95	61.5	100	-14.85	60.95	17.35	11,58,131	14,092	53,319	8.980798383
3,792	2,021	93,314	14.17	244.2	-15.8	250	241.55	245	100	17,150.00	100	74.2	76	5,750	-14.25	75	17.3	5,08,553	6,924	12,712	3.352320675
23,875	9,911	7,96,539	13.94	207.4	-18.75	100	207.4	209.2	50	17,200.00	800	91	91.4	100	-15	91	17.22	18,65,018	33,238	68,889	2.885403141
17,275	14,318	7,77,815	14.51	179.95	-14.25	950	177.4	179	1,100	17,250.00	250	108.6	110.45	300	-15.9	108.55	17.04	10,99,433	14,646	21,975	1.272069465
65,104	26,229	23,80,340	14.07	147.15	-18.45	50	147.15	150	50	17,300.00	3,850	131	132.8	100	-12.25	132.5	17.31	21,15,903	22,498	64,693	0.993687024
27,481	13,238	7,59,811	14.04	121	-17.8	450	121	124.4	50	17,350.00	250	153.75	155.05	50	-9.75	157	17.33	3,71,121	2,723	14,944	0.543793894
82,843	2,148	16,62,275	14.13	99	-15.3	50	98.45	100	6,800	17,400.00	300	180.9	182.45	200	-9.4	182.45	17.18	5,71,399	-16,049	23,935	0.288920005
19,794	8,053	4,86,924	14.17	79.7	-13.1	200	79.1	79.7	900	17,450.00	150	211	214.05	750	-8.05	213.15	17.33	55,226	838	2,388	0.120642619
264658		7230724																12248476		640215	2.419027575

Measuring Quality Volatility (QV)

Can we use something similar to PCR, simple yet effective way to measure Quality direction?

Let's build Quality Volatility:

In House Bugs (IHB):

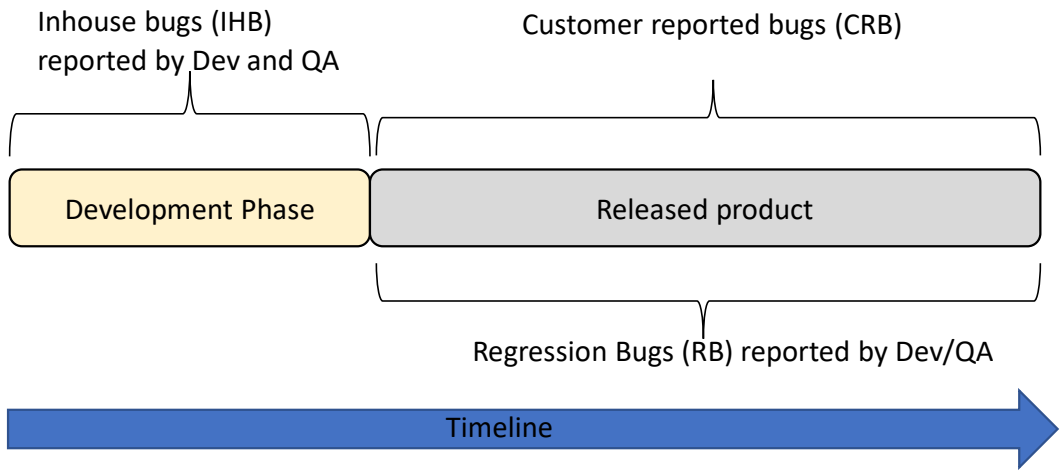
- Bugs uncovered during the development cycle by both Dev and QA

Customer Reported Bugs (CRB):

- Any bugs reported by the customer from field

Production / Regression bugs:

- These are uncovered because the In-House Testing phase did miss them.



Defining Quality Volatility (QV)

Simple Quality Volatility can be defined as

$$Quality\ Volatility\ = \frac{Total\ number\ Customer\ Reported\ Bugs}{Total\ number\ of\ In\ House\ Bugs}$$

*The weight associated with each bug is the same. A priority 5 (P5) bug is treated as same weight as priority 1 (P1) bug.

To make the Quality volatility more realistic, Weight needs to be associated with the respective priority of the bug reported. Weights are $\{P1 = 5, P2 = 4, P3 = 3, P4 = 2, P5 = 1\}$, where P1 is Priority 1 bug and so on.

$$Weighted\ Quality\ Volatility\ = \frac{CRB\ \{(5 *\ P1\ count) + (4 * P2\ count)\ + (3 * P3\ count)\ + (2 * P2\ count)\ + (1 * P5\ count)\}}{IHB\ \{(5 *\ P1\ count) + (4 * P2\ count)\ + (3 * P3\ count)\ + (2 * P2\ count)\ + (1 * P5\ count)\}}$$

What Quality Volatility indicates

- Quality volatility ratio represents stability of the product
- A series of these measurements give us trend
- Quality volatility is lagging indicator
- This ratio can be found when both data points i.e CRB and IHB numbers are available
- When we have a series of Quality Volatility ratios from multiple releases, it is possible to extrapolate the next value in the series,

Ratio	Status	What does it mean	Suggested recommendation
0 - 0.2	Green	Every 10 bugs found in house, after production we can expect 2 bugs	• Add more test cases • Increase test coverage
0.2 - 0.5	Yellow	Every 2 bugs logged in house, 1 bug was logged by customer.	• Review existing test cases • Add new test cases
0.5 - 1	Red	Every bug found in house; customer is matching the count	• Identify testing gaps • Increase coverage • Make customer logged bugs as part of regression testing • Increase manual as well as Automation coverage
> 1	Black	Customer is logging more bugs than in house.	• Current testing has shortcomings • Do architecture review of the features built and customer expectation • Identify customer user cases and build test scenarios for effective testing

Real time Quality Volatility Data:

Simple quality volatility

Volatility Index : In House reported Bugs (IHB) / Customer Reported Bugs (CRB) 0 – 0.2 , 0.2 – 0.5 , > 0.5

Release	Product 1			Product 2			Product 3			Product 4			Product 5		
	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility
N+4	20	7	0.35	25	9	0.36	10	4	0.40	4	1	0.25	10	4	0.40
N+3	12	6	0.5	15	8	0.53	9	4	0.44	12	1	0.08	13	2	0.15
N+2	15	16	1.07	17	22	1.29	4	8	2.00	3	8	2.67	10	7	0.70
N+1	15	13	0.87	26	24	0.92	7	9	1.29	18	10	0.56	14	8	0.57
N	25	16	0.64	20	23	1.15	9	15	1.67	7	19	2.71	18	10	0.56

Weighted quality volatility

*The weights are Blocker = 5, Critical = 4, Major = 3

Release	Product 1									Product 2									Product 3									Product 4									Product 5								
	IHB				CRB				QA	IHB				CRB				QA	IHB				CRB				QA	IHB				CRB				QA Volatility	IHB				CRB				QA
	Bl	Cr	Ma	Total	Bl	Cr	Ma	Total	Volatility	Bl	Cr	Ma	Total	Bl	Cr	Ma	Total	Volatility	Bl	Cr	Ma	Total	Bl	Cr	Ma	Total	Volatility	Bl	Cr	Ma	Total	Bl	Cr	Ma	Total		Bl	Cr	Ma	Total	Bl	Cr	Ma	Total	Volatility
N+4	5	7	8	20	2	2	3	7	0.35	1	9	15	25	3	2	4	9	0.41	2	1	7	10	1	1	2	4	0.43	1	2	1	4	0	1	0	1	0.25	2	4	4	10	1	2	1	4	0.42
N+3	5	3	4	12	1	3	2	6	0.47	2	4	9	15	1	4	3	8	0.57	2	2	5	9	1	2	1	4	0.48	0	0	12	12	0	1	0	1	0.11	5	6	2	13	0	2	0	2	0.15
N+2	2	7	6	15	3	8	5	16	1.11	0	4	13	17	2	7	13	22	1.40	2	1	1	4	2	6	0	8	2.00	0	0	3	3	2	5	1	8	3.67	4	5	1	10	2	3	2	7	0.65
N+1	4	5	6	15	2	8	3	13	0.88	3	4	19	26	5	16	3	24	1.11	1	2	4	7	1	7	1	9	1.44	0	2	16	18	1	9	0	10	0.73	2	4	8	14	1	6	1	8	0.64
N	5	11	9	25	5	6	5	16	0.67	2	3	15	20	4	17	2	23	1.40	6	1	2	9	1	13	1	15	1.50	0	3	4	7	6	12	1	19	3.38	1	1	16	18	0	4	6	10	0.60

** Our observation, between Simple QV and Weighted QV, there is a very minor change in Volatility

How do we use Quality Volatility ratio information to increase the quality ? Let's park that question for now

Test case effectiveness

Test case effectiveness is the test case's potential to uncover a bug.

- Once the code is fixed, the code is now immune to the test case conditions
- Post fix, the test case is unlikely to break same code flow
- Subsequent runs of the test case are reconfirmation test case of regression cycles

Release over release if we do not uncover a bug, the test Bug detection potential value factor decrease. How do we measure this ?

Half Life Equation

Test case effective ness can be measured using Half Life equation.

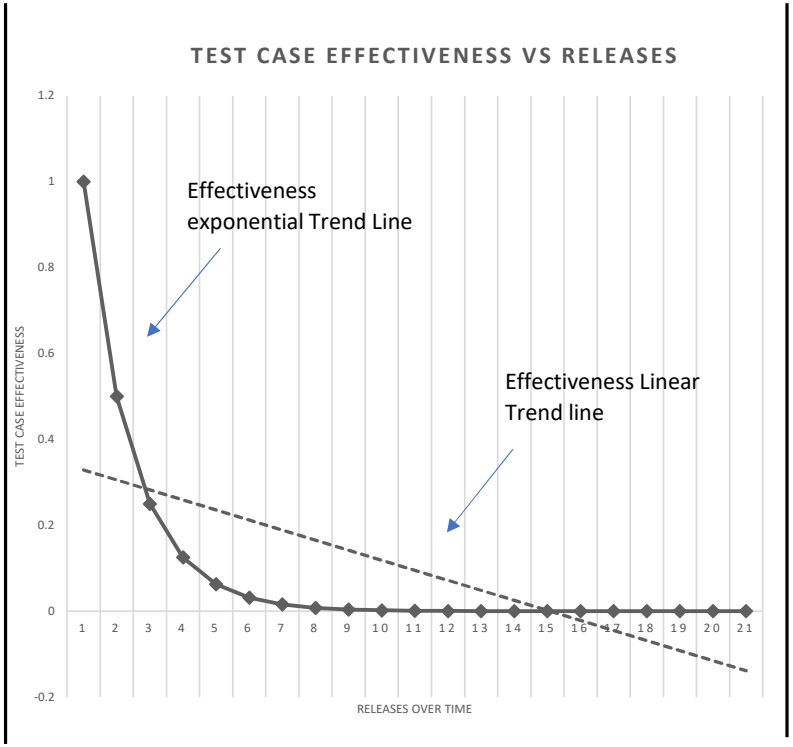
$$N_{(t)} = N_{(0)} * \left(\frac{1}{2}\right)^{\frac{t}{t_{\frac{1}{2}}}}$$

- $N_{(0)}$ = Potential value of test case to find a bug during test design phase.
  - ❑ The Value = "1" as each test case can find 1 bug for the code flow it is/was designed to test.
- $N_{(t)}$ = Effectiveness retained of a test case at the end of "t" release execution.
- t = Number of releases / executions that have been completed.
- $t_{(1/2)}$ = Time taken for test case to reach half its effectiveness.

Test case effectiveness half life when t = 1

Using the values of: $N_{(0)} = 1$, t = 1 to n (where n is current release number), $t_{(1/2)} = 1$

N(t) = Every subsequent release	Test case Effective ness	Probability of detection %
1	1.000000	100.000000
2	0.500000	50.000000
3	0.250000	25.000000
4	0.125000	12.500000
5	0.062500	6.250000
6	0.031250	3.125000
7	0.015625	1.562500
8	0.007813	0.781250
9	0.003906	0.390625
10	0.001953	0.195313
11	0.000977	0.097656
12	0.000488	0.048828
13	0.000244	0.024414
14	0.000122	0.012207
15	0.000061	0.006104
16	0.000031	0.003052
17	0.000015	0.001526
18	0.000008	0.000763
19	0.000004	0.000381
20	0.000002	0.000191



Observations:

- After 10 releases, probability of test case uncovering a bug is < 0.01%
- Post 15 release, potential effectiveness = 0
- Best time to review a test case is after 10 release cycles

*If a test case detects a bug in runs, reset the value of test case equal to the number of releases it was run.

What next do we do with Quality Volatility

QA Volatilely : Identify which components are concern for customer

Test Case effectiveness: Identify which test cases / test groups are up for review

Steps we adopted to bring Quality Volatility <0.2 (a.k.a better quality)

- Incorporate Exploratory testing, Exploratory testing is one of the quickest ways to uncover bugs
- Using Test case effectiveness, identify test cases which need review
- Design review by QA : “Best place to find a bug is even before it is coded” This comes much before code review and far more effective.
- For sustenance projects, last 3-6 release customer issues are must to be included in regression cycles.

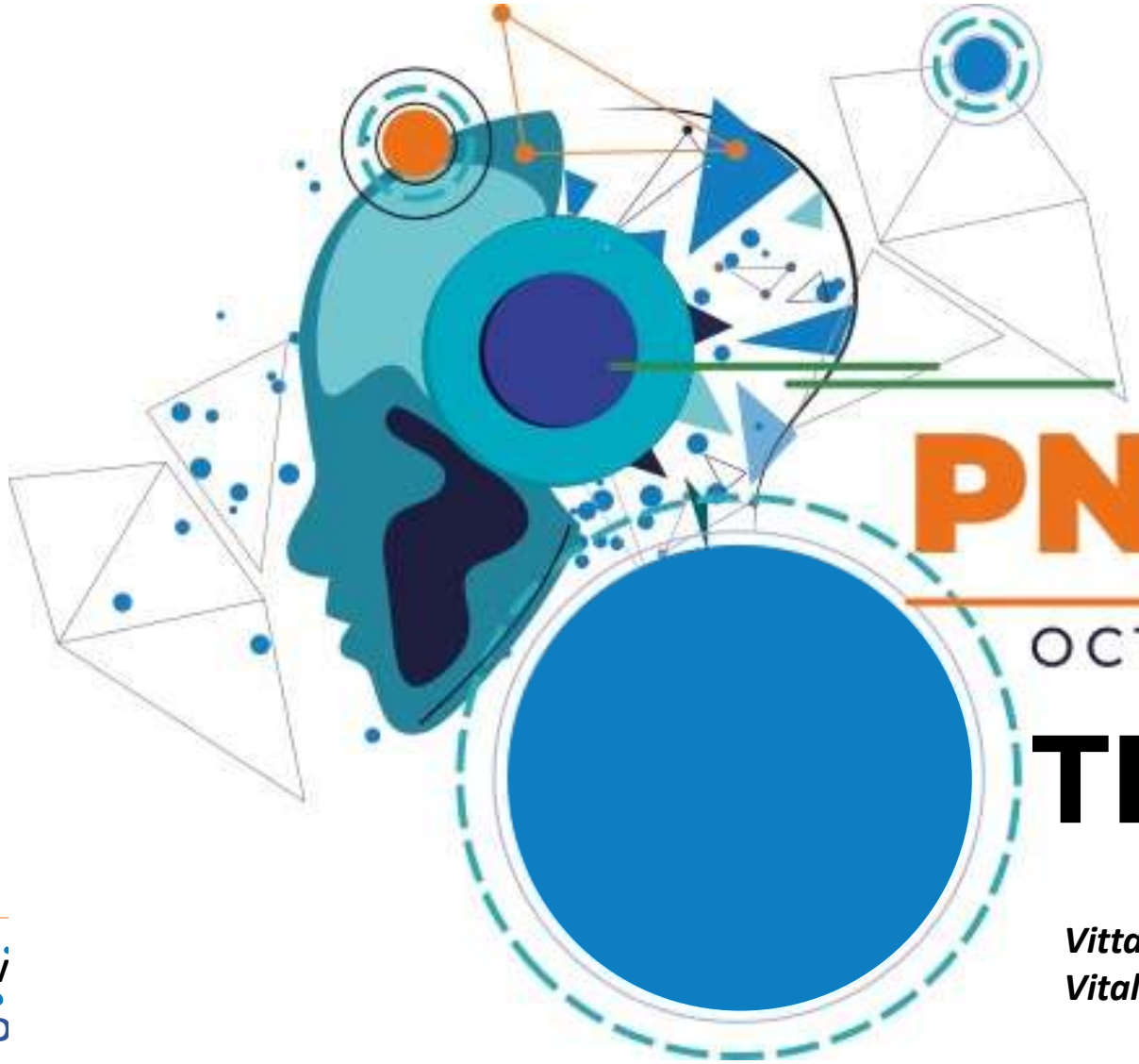
Result of we achieved

Release	Product 1			Product 2			Product 3			Product 4			Product 5		
	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility
N+4	20	7		25	9		10	4		4	1		10	4	
N+3	12	6		15	8		9	4		12	1		13	2	
N+2	15	16	1.07	17	22	1.29	4	8	2.00	3	8	2.67	10	7	0.70
N+1	15	13	0.87	26	24	0.92	7	9	1.29	18	10	0.56	14	8	0.57
N	25	16	0.64	20	23	1.15	9	15	1.67	7	19	2.71	18	10	0.56

Release	Product 1			Product 2			Product 3			Product 4			Product 5		
	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility	IHB	CRB	QA Volatility
N+4	20	7	0.35	25	9	0.36	10	4	0.40	4	1	0.25	10	4	0.40
N+3	12	6	0.5	15	8	0.53	9	4	0.44	12	1	0.08	13	2	0.15
N+2	15	16	1.07	17	22	1.29	4	8	2.00	3	8	2.67	10	7	0.70
N+1	15	13	0.87	26	24	0.92	7	9	1.29	18	10	0.56	14	8	0.57
N	25	16	0.64	20	23	1.15	9	15	1.67	7	19	2.71	18	10	0.56

Key Takeaways

- We do not need complex measurements to track software quality
- Customer reported bugs is what matters
- Exploratory testing and QA teams' involvement in feature design review are most cost-effective technique for early bug detection
- Code does become immune to testing over multiple releases
- Moving from "test to pass" to "test to break" testing mindset is critical for bug uncovering
- There is negligible difference between Simple QV and Weight QV. For a start, we can begin with Simple QV to help identify concerning products



PNSQC

OCTOBER 10-12 2022

THANK YOU

Vittalkumar Mirajkar
VitalkumarrMirajkar@gmail.com