stest
SOFTWARE & SYSTEMS TESTING

Knowing you test really efficiently

Roman Jurkech

Questions you may have to answer one day…

- How good your testing is?
- Is testing of the current release better than the previous one?
- Is testing in our project better than in other projects?
- How much testing is enough?
- Are we testing too little?
- Are we testing too much?
- What is the actual benefit of testing for business?
- What is the return on investment for our testing?
- Why shouldn’t I just fire all of you???

What are the benefits of testing for business?

- Quantitative
  - Defects which (after fix and retest) did not get to the next stage
  - Cost savings
- Qualitative
  - Improved reputation
  - Increase in trust
  - Prevention of legal disputes
  - Decreased risk of project or product failure

Which project has the most efficient testing?

Fictional examples, not real projects!

Project	A		B		C	
System testing						
Testing period (number of days)	30		60		40	
Testers	4		5		7	
Effort (man-days)	93		250		266	
Number of planned test cases	480		1283		918	
Number of executed test cases	417	87%	1067	83%	894	97%
Requirement coverage (by test cases)		96%		88%		98%
Number of detected defects	323		592		720	
Number of resolved defects	302	93%	519	88%	695	97%
Number of closed defects	297	92%	491	83%	683	95%
Number of duplicates	18	6%	53	9%	33	5%
Average number of test cases per tester	80,75	25%	118,4	20%	102,9	14%

All of the above is nice, however:

- **How successful were we in terms of discovered defects?**
- **How much money we saved because of testing?**

Indicators displayed here will not give you answers to any of those questions!

Defect Detection Percentage (DDP)

- How to calculate DDP (basic variant):

1. Count DDDT – defects discovered during testing
2. Count DDAT – defects discovered after testing (usually after live release)

$$\text{DDP} = \frac{\text{DDDT}}{\text{DDDT} + \text{DDAT}}$$

Calculating Defect Detection Percentage

Fictional examples, not real projects!

Project	A		B		C	
System testing						
Testing period (number of days)	30		60		40	
Testers	4		5		7	
Effort (man-days)	93		250		266	
Number of planned test cases	480		1283		918	
Number of executed test cases	417	87%	1067	83%	894	97%
Requirement coverage (by test cases)		96%		88%		98%
Number of detected defects	323		592		720	
Number of resolved defects	302	93%	519	88%	695	97%
Number of closed defects	297	92%	491	83%	683	95%
Number of duplicates	18	6%	53	9%	33	5%
Average number of test cases per tester	80,75	25%	118,4	20%	102,9	14%
Acceptance testing						
Number of detected defects	81		168		233	
First 6 months of production						
Number of detected defects	26		84		106	

Total number of discovered defects	430		844		1059	
Defect detection rate during system testing		75%		70%		68%

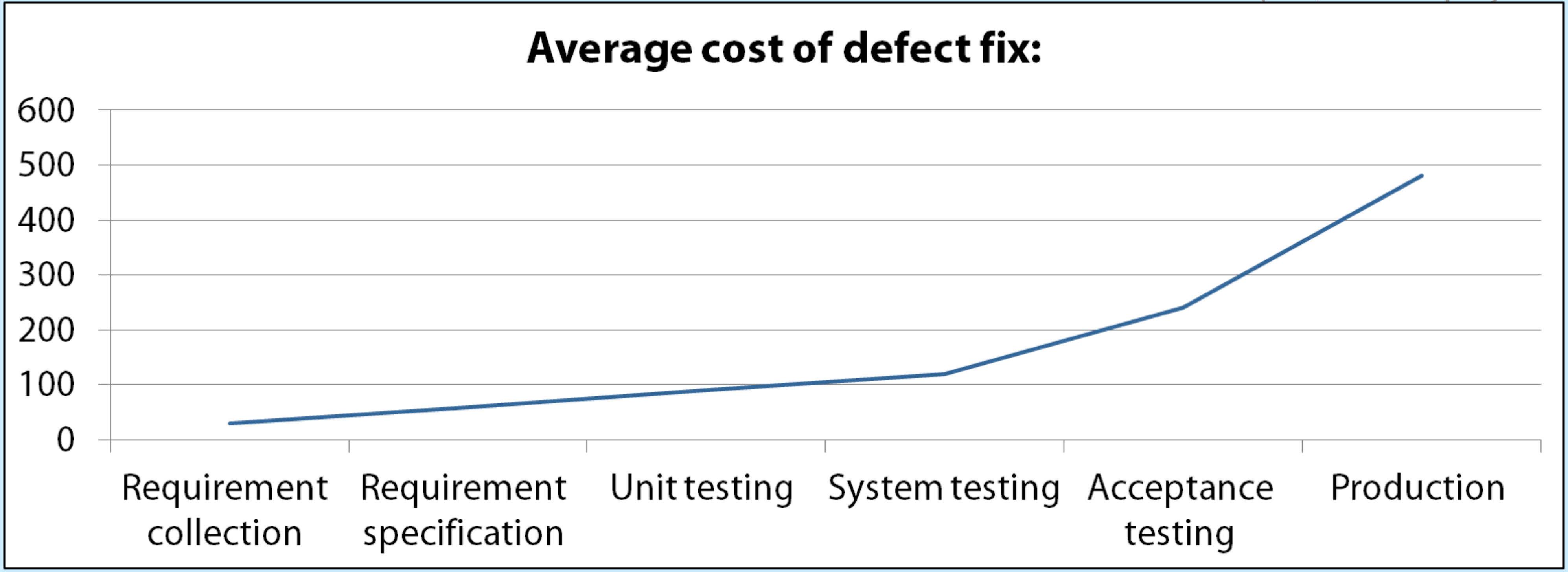
Calculating cost saving

- Cost Savings:

1. Calculate cost of testing
2. Calculate cost of fixing defects discovered during testing
3. Calculate cost of fixing defects discovered during later stages of software development life cycle
4. Calculate cost savings achieved by testing

Average cost of a defect fix

Fictional examples, not real projects!



Average hourly rate:

30 €

Defect detection phase	Average effort required to fix one defect	Cost of defect fix
Requirement engineering	1 mh	30 €
Technical design	2 mh	60 €
Unit testing	3 mh	90 €
System testing	4 mh	120 €
Acceptance testing	8 mh	240 €
Production	16 mh	480 €

Calculating savings

Fictional examples, not real projects!

A

Defect detection during	Average effort required to fix one defect
Requirement engineering	1 mh
Technical design	2 mh
Unit testing	3 mh
System testing	4 mh
Acceptance testing	8 mh
Production	16mh

Average hourly rate	30 €

Cost of system testing	93 MDs	*	8 h	*	30 €	=	22320 €
Cost of defect fixing during system testing	323 defects	*	4 h	*	30 €	=	38760 €
Total cost of system testing and defect fixing							61080 €

Cost of fixes for defects discovered during acceptance testing	323 defects	*	8 h	*	30 €	=	77520 €
Cost of fixes for defects discovered during production	323 defects	*	16 h	*	30 €	=	155040 €

Minimal calculated savings achieved by system testing	16440 €	21%
Maximal calculated savings achieved by system testing	93960 €	61%

Calculating savings

Fictional examples, not real projects!

B

Defect detection during	Average effort required to fix one defect
Requirement engineering	1 mh
Technical design	2 mh
Unit testing	3 mh
System testing	4 mh
Acceptance testing	8 mh
Production	16 mh

Average hourly rate	30 €

Cost of system testing	250 MDs	*	8 h	*	30 €	=	60000 €
Cost of defect fixing during system testing	592 defects	*	4 h	*	30 €	=	71040 €
Total cost of system testing and defect fixing							131040 €

Cost of fixes for defects discovered during acceptance testing	592 defects	*	8 h	*	30 €	=	142080 €
Cost of fixes for defects discovered during production	592 defects	*	16 h	*	30 €	=	284160 €

Minimal calculated savings achieved by system testing	11040 €	8%
Maximal calculated savings achieved by system testing	153120 €	54%

Calculating savings

Fictional examples, not real projects!

C

Defect detection during	Average effort required to fix one defect
Requirement engineering	1 mh
Technical design	2 mh
Unit testing	3 mh
System testing	4 mh
Acceptance testing	8 mh
Production	16 mh

Average hourly rate	30 €

Cost of system testing	266 MDs	*	8 h	*	30 €	=	63840 €
Cost of defect fixing during system testing	720 defects	*	4 h	*	30 €	=	86400 €
Total cost of system testing and defect fixing							150240 €

Cost of fixes for defects discovered during acceptance testing	720 defects	*	8 h	*	30 €	=	172800 €
Cost of fixes for defects discovered during production	720 defects	*	16 h	*	30 €	=	345600 €

Minimal calculated savings achieved by system testing	22560 €	13%
Maximal calculated savings achieved by system testing	195360 €	57%

Which project has the most cost effective system testing?

Fictional examples, not real projects!

Project	**A**	**B**	**C**
Cost of system testing	22320 €	60000 €	63840 €
Cost of defect fixing during system testing	38760 €	71040 €	86400 €
Total cost of system testing and defect fixing	61080 €	131040 €	150240 €
Cost of fixes for defects discovered during acceptance testing	77520 €	142080 €	172800 €
Cost of fixes for defects discovered during production	155040 €	284160 €	345600 €
Minimal calculated savings achieved by system testing	16440 €	11040 €	22560 €
Maximal calculated savings achieved by system testing	93960 €	153120 €	195360 €
Minimal calculated % savings achieved by system testing	[✓ **21%**]	**8%**	**13%**
Maximal calculated % savings achieved by system testing	[✓ **61%**]	**54%**	**57%**

Advanced application of DDP

- **Expanding the DDP to multiple stages of testing – we measure it for:**
  - Unit testing
  - System testing
- **After system testing the product goes live**
- **DDP is the number of defects found by a test level, divided by the number found by that test level and any other means afterwards (ISTQB Glossary)**
- **Application seems straightforward**
- **Analysis and understanding of results might be tricky**

Project GreatestThingSinceSlicedBread™ release 1

- **There are 200 defects hidden in the first release of our project and testing delivers the following:**
  - Unit testing – 100 defects – DDP=100/200=50%
  - System testing – 80 defects – DDP=80/100=80%
- **20 defects (10%) are discovered only in production**

Project GreatestThingSinceSlicedBread™ release 2

- **Amazing thing happens and there are 200 defects hidden in this release AGAIN!**
- **However, a greatly skilled programmer Lucy joined the company**
- **Lucy extended unit tests and discovered 10 additional defects which would have gotten to production so we end up with:**
  - Unit testing – 110 defects – DDP=110/200=55% (up by 5%)
  - System testing – 80 defects – DDP=80/90=89% (up by 9%)
- **And we have only 10 defects (=5%) getting into production now**
- **Nice effort from Lucy but the system testers really killed it! Or did they?**

Project GreatestThingSinceSlicedBread™ release 3

- **Another release, but 200 defects YET again!**
- **Lucy was a bit angry after the last release, she put so much effort into her unit tests but the system testing guys received all the fame and glory**
- **She worked twice as hard for this release and implemented even more unit tests but they catch 20 defects which would be previously discovered during system test stage:**
  - Unit testing – 130 defects – DDP=130/200=65% (up by 10% this time)
  - System testing – 60 defects – DDP=60/70=86% (down by 3%)
- **Looks like Lucy really did it this time! Or did she?**

Interpreting DDP numbers

- **Extremely high DDP can mean several things:**
  - Testing is fantastic, everyone should get a bonus!
  - Is anyone even using the system? At all???
  - Are we counting the numbers right?
- **Very low DDP could implicate:**
  - Testing is horrible, everyone is fired!
  - We don’t have sufficient resources for testing (time, manpower…)
  - Testing has no clue about the product due to missing or vague requirements.

Other considerations for using DDP?

- **It is not as easy to implement, use and interpret as it might look.**
- **Start "small", just count all the testing before release, get to more structured measurement later.**
- **Decide amount of time after "live" to include into DDP – e.g. 1, 3, 6 months**
- **Defect severity can play important part – you may decide to measure only certain severity level(s) or measure them separately.**
- **Once you establish satisfactory DDP measuring procedures, use them consistently.**
- **Monitor trends over time – it can give you many insights. DDP will not only tell you how good your testing is but also how external factors affect your testing.**
- **Using historical DDP rates to make prediction – possible, but it must not be just taken and used "as is"!**

Other considerations for using DDP?

- **DDP can be skewed, both intentionally and unintentionally:**
  - Purposeful over-reporting of defects by the test team – duplicates, defect reports without any value, etc.
  - Manipulating reports of defects from time after live release.
  - Enhancements and feature requests can be reported as defects.
  - Obscure defects unrelated to the most recent release are reported.

Other considerations for using DDP?

- **Maturity of the project – both extreme ends of the spectrum might be unsuitable, e.g.:**
  - Project which has just started, has no live release in sight.
  - Project in maintenance mode with very little development.
- **Size of the project – it can be just too small to make sense.**
- **Are your defect tracking procedures ready for DDP?**
  - Ideally your tracking system should be able to support automatic calculation of DDP if configured properly.
- **Don’t rely on DDP only – there are other lovely test metrics ☺**

Conclusion

- Whenever it makes sense, collect data which enables you to measure test efficiency.
- Strive to describe benefits of testing from the business point of view.
- You can do no wrong with a few hard numbers – saved money speaks very loud and some colorful charts can do wonders at management meetings ☺
- Try to do measurement at all stages of testing.
- DDP measured should be interpreted with cool head, you need to consider circumstances and avoid "traps".

Questions?