Technology Consulting and Engineering

ALTEN

The 20% of software engineering practices that contribute to 80% of the profits

Gualtiero Bazzana

ACM / IEEE International Symposium on
Empirical Software Engineering and Measurement (ESEM)

8th International Symposium on
Empirical Software Engineering and Measurement
September 18-19, 2014, Torino, Italy

Who am I ?

ALTEN

- Technical background (PhD in Computer Science in 1989)
- Working in sw engineering & quality research in the first 6 years of my career
- Entrepreneur and Manager later on:
  - Founder and CEO of a sw engineering service provider, later on successfully sold
  - Currently Managing Director of ALTEN ITALIA, an horizontal IT Consulting company of 900 FTEs, part of an international group of 17000 FTEs listed at the stock exchange
  - Marketing Manager of ISTQB®
  - CEO of ITA-STQB

ISTQB® International Software Testing Qualifications Board

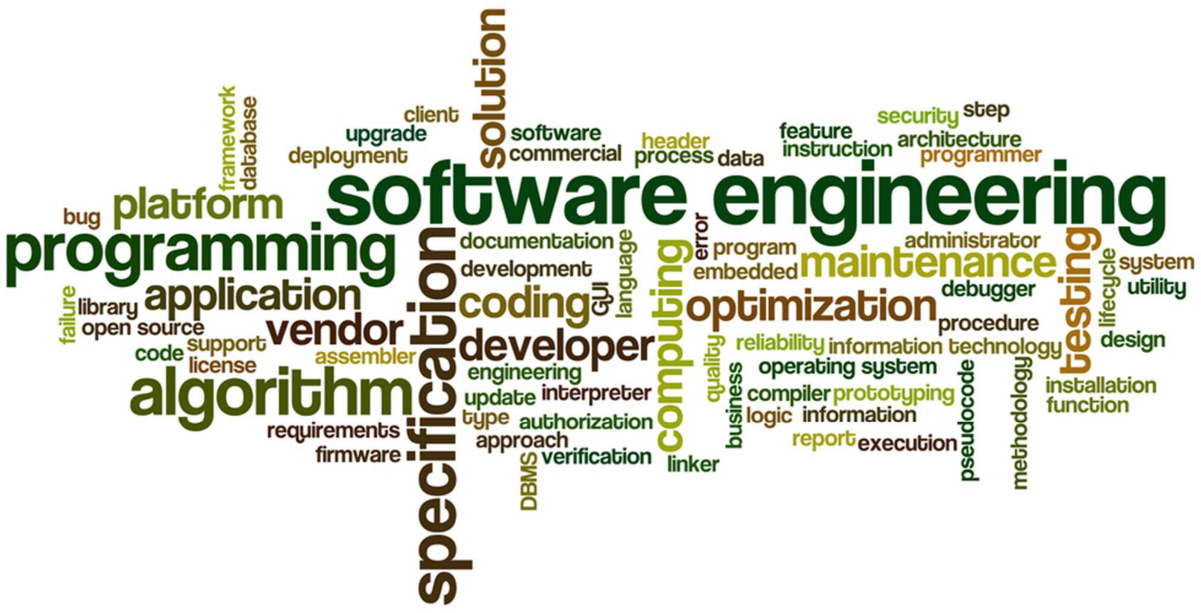
Challenges vs Opportunities

Budget constraints

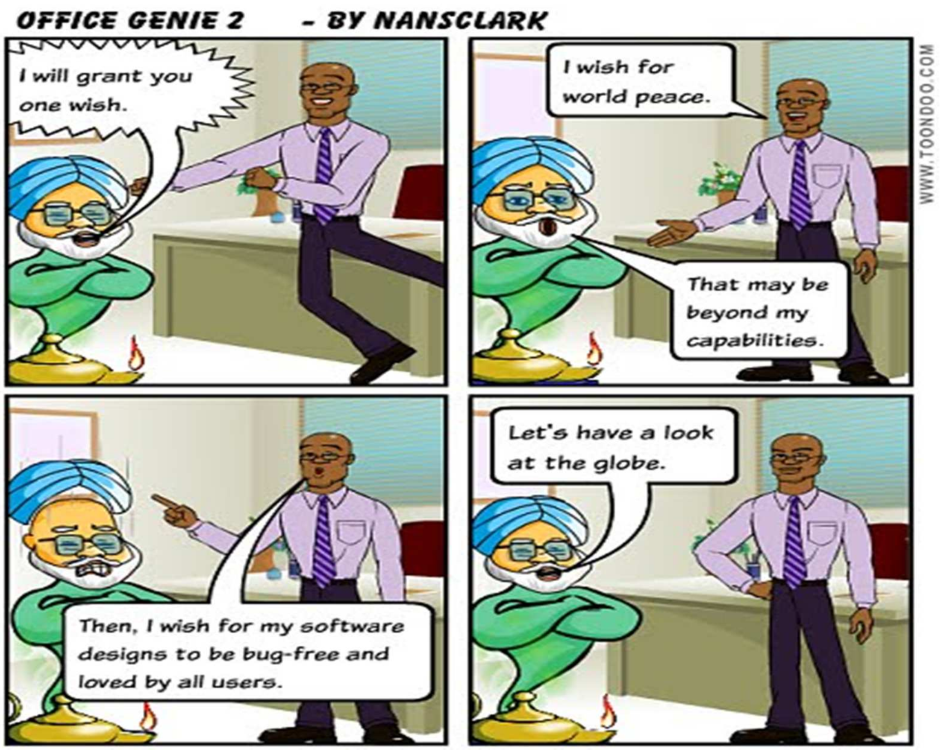
Quality of services

Commitment on results

SW Engineering practices



I have a dream ….


OFFICE GENIE 2 - BY NANSCLARK
I will grant you one wish.
I wish for world peace.
That may be beyond my capabilities.
Then, I wish for my software designs to be bug-free and loved by all users.
Let's have a look at the globe.
WWW.TOONDOO.COM

Applicability of Sw Engineering Practices

- ✓ Rigorous application of sw engineering methods helps to reduce the risk that problems will show during operation, contributing to achieve a better system quality and a cost optimization
- ✓ Software QA and QC are risk management activities, they are more important where risks are higher
- ✓ They applicable to both SW intensive products and IT Systems

A Risk Analysis Perspective

Risk level is normally determined on the basis of the potential impact of a failure

Class A

- Loss of Human Lives

Focus of the speech

Class B

- Huge Economic loss (involving also third parties)

Class C

- Economic Loss

Class D

- Loss of Performed Work

Class A domains

CLASS A – Loss of Human Lives

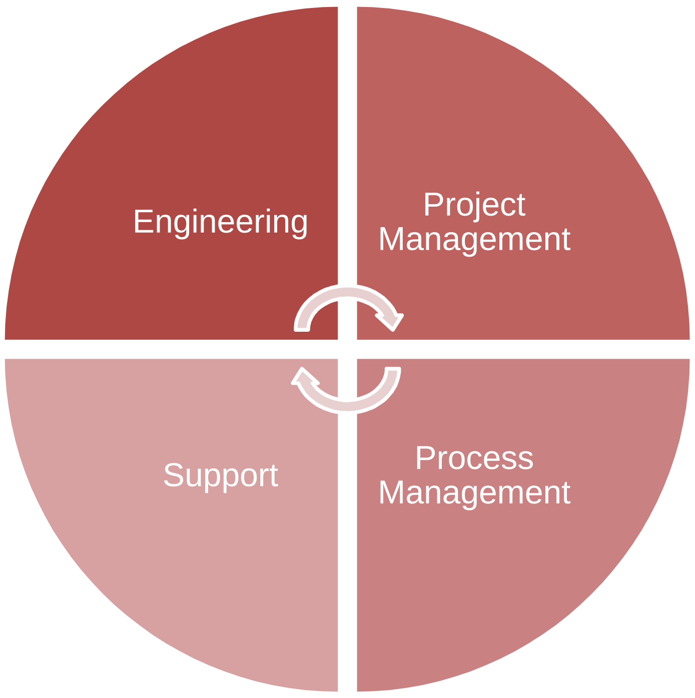
- This is the area of “safety critical applications” that are typically regulated by domain specific standards in terms of levels/ intensity of sw engineering practices
- Application domains
  - Nuclear plants
  - Avionics, defense
  - Automotive (safety related parts)
  - Railway
  - Medical devices with impact on human life

Class B domains

CLASS B – Huge Economic Loss

- This is the area of “SW intensive systems that are distributed in the large” and of “systems requiring a very high reliability/ QoS”
- Application domains
  - TLC infrastructure
  - TLC operators core processes
  - E-commerce/ billing applications that are essential to the company business
  - SW embedded in machine tools
  - Banking “core” applications/ stock exchange
  - Energy/ Gas distribution
  - Automotive Infotainment
  - Control Systems for complex applications
  - Pharmaceutical applications GxP relevant

Deployment of CMMI® process areas in ClassB domains



Engineering practices

Process Group	Level	Process Area	
Engineering	-	Requirements Development	🟡
		Technical Solutions	🟢
		Verification	🟡
		Validation	🟢
		Product Integration	🟡

Project Management

Process Group	Level	Process Area	
Project Management	Basic	Project Planning	🟡
		Project Monitoring and Control	🟢
		Requirements Management	🟡
		Supplier Agreement Management	🟢
	Advanced	Quantitative Project Management	🔴
		Integrated Project Management	🔴
		Risk Management	🟡

Process Management

Process Group	Level	Process Area	
Process Management	Basic	Organizational Process Definition	🟡
		Organizational Process Focus	🔴
		Organizational Training	🟡
	Advanced	Organizational Process Performance	🔴
		Organizational Performance Management	🔴

Support Areas

Process Group	Level	Process Area	
Support	Basic	Measurement and Analysis	🟡
		Configuration Management	🟡
		Process and Product Quality Assurance	🔴
	Advanced	Decision Analysis and Resolution	🟡
		Causal Analysis and Resolution	🔴

Some remarkable stories I was personally confronted with ….

- “We need to simplify our processes: our CMMI-L3 is far too expensive”
- “We cannot be harsh towards the supplier because we declared to management that the project was on spec, on time, on budget to get the bonus”
- “Process tailoring is a good idea … it helps us in finding good reasons not to perform quality control”
- “We do not really need automated regression testing … ask the development team to perform testing … the day after outage of 4 hours, with 10+ million users impacted”
- “With Agile we can get rid of documentation and testing”
- “Since your DAR (Decision Analysis and Resolution) process brings to an outcome that is not in line with our expectations, we will have to decide by our own”
- “Since the testing activities are finding too many bugs and this is putting the go-live date in jeopardy, we have no othetr chance: let’s stop testing on our new mobile apps!”

Some remarkable stories I was personally confronted with (2) ….

All such stories happened in healthy companies, with IT budgets in the range of several hundreds M€ / year ….

New technologies are bringing more wilderness to the sw engineering practices: Web and the Mobile Apps …. We start with toy applications but then we keep on applying the same approach for corporate/ business applications

Do we still teach sw engineering at universities ?!?

Maturity by domain

- TLC infrastructure 🟢
- TLC operators core processes 🟡
- E-commerce core applications/ gambling
- SW embedded in machine tools 🔴
- Banking “core” applications 🔴
- Stock exchange 🟢
- Utilities 🔴
- Automotive Infotainment 🟡
- Control Systems for complex applications 🟡
- Pharmaceutical applications GxP relevant 🟢

A more scientific approach

My personal analysis is clearly biased by the fact that:
- The data sample is not statistically valid
- I am working in a country struggling with economic crisis and in which budgetary constraints are stronger than ever

So, let’s be more analytic and analyze an excerpt from the work of Capers Jones on

“SCORING AND EVALUATING
SOFTWARE METHODS, PRACTICES, AND RESULTS”

It is based on:
- the author’s book Software Engineering Best Practices published by McGraw Hill in 2010.
- Some new data is taken from The Economics of Software Quality published by Addison Wesley in 2012.
- The data is current through mid-2014 for personal courtesy of Mr. Capers Jones.

Quantitative Data Sample

In order to evaluate the effectiveness or harm of various methods and practices a scoring method has been developed by Capers Jones

The scoring method runs
- For 435 sw engineering methods
- Ranking from +10 for maximum benefits to -10 for maximum harm.

The data for the scoring comes from observations among about:
- 150 Fortune 500 companies,
- some 400 smaller companies,
- and 30 government organizations.
- Negative scores also include data from 15 lawsuits.

The rankings are based on about:
- 20,000 projects that span
- 50 industries and
- 24 countries

Some considerations

Selecting a set of “best practices for software engineering” is a fairly complicated undertaking

For software engineering a serious historical problem has been that measurement practices are so poor that quantified results are scarce.
- There are many claims for tools, languages, and methodologies that assert each should be viewed as a “best practice.”
- But empirical data on their actual effectiveness in terms of quality or productivity has been scarce.

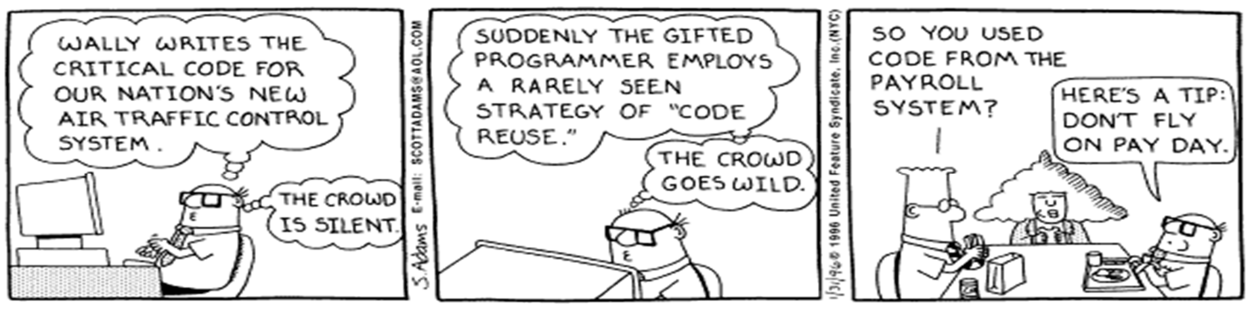
The midpoint or “average” against which improvements are measured are
- traditional application development methods such as “waterfall” development
- performed by organizations that either don’t use the Software Engineering Institute’s capability maturity model or are at level 1.
- Low-level programming languages are also assumed.

This fairly primitive combination remains among the more widely used development method even in 2014 although various forms of Agile have pulled ahead.

24 Methods that improve productivity and costs

	Productivity/Quality Factors	Impact on Work Hours
1	Certified reusable components	-80%
2	Experienced development teams	-65%
3	Mashups < 1000 function points	-50%
4	Patterns - successful applications	-50%
5	Effective methodologies for specific project types	-40%
6	High-level programming languages	-30%
7	Use of inspections for complex systems	-27%
8	Use of SEMAT for complex systems	-25%
9	Experienced managers	-25%
10	Moderate unpaid overtime by teams	-20%
11	Low requirements creep	-20%
12	Logical, planned architecture for large systems	-20%
13	Model-based development	-20%
14	Due diligence on COTS acquisitions	-20%
15	Use of static analysis before testing	-18%
16	High CMMI levels	-15%
17	Low cyclomatic complexity (< 10)	-15%
18	Effective project status tracking	-15%
19	Effective defect prevention (JAD, Kaizen, etc.)	-15%
20	Experienced test teams	-12%
21	TSP or RUP > 5000 function points	-12%
22	Experienced clients	-10%
23	SCRUM < 1000 function points	-10%
24	Agile < 1000 function points	-8%

Dilbert view on Re-Use and Agile


WALLY WRITES THE CRITICAL CODE FOR OUR NATION'S NEW AIR TRAFFIC CONTROL SYSTEM.
THE CROWD IS SILENT.
SUDDENLY THE GIFTED PROGRAMMER EMPLOYS A RARELY SEEN STRATEGY OF "CODE REUSE."
THE CROWD GOES WILD.
SO YOU USED CODE FROM THE PAYROLL SYSTEM?
HERE'S A TIP: DON'T FLY ON PAY DAY.
S. Adams E-mail: SCOTTADAMS@AOL.COM
1/31/96 © 1996 United Feature Syndicate, Inc.(NYC)

WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com scottadams@aol.com
©2007 Scott Adams, Inc./Dist. by UFS, Inc.

26 Methods that hamper productivity

	Productivity/Quality Factors	Impact on Work Hours
1	Informal test case design	8%
2	Scrum > 5000 function points	8%
3	Agile > 5000 function points	9%
4	Low CMMI levels	10%
5	Waterfall > 5000 function points	12%
6	Ineffective methodologies	15%
7	Inexperienced test teams	15%
8	Distributed teams: poor communications	15%
9	Pair programming	16%
10	High cyclomatic complexity (> 25)	18%
11	Poor status tracking	20%
12	Unverified, buggy COTS acquisitions	20%
13	Excessive unpaid overtime by team	23%
14	Manual estimates > 1000 function points	23%
15	Adding personnel to late projects	25%
16	Low-level programming languages	25%
17	Concurrent maintenance and development tasks	30%
18	False claims by outsource vendors	30%
19	Inaccurate manual estimates	33%
20	Inexperienced development teams	35%
21	Chaotic, unplanned architecture for large systems	37%
22	Inexperienced clients	40%
23	Truncating testing to "meet schedule"	45%
24	Concealing problems in status reports	45%
25	Inexperienced managers	50%
26	High requirements creep: poor change control	60%

Dilbert view on Requirements and Testing

TODAY IS THE THREE-YEAR ANNIVERSARY OF OUR FIRST MEETING TO DISCUSS PROJECT REQUIREMENTS.
AND WE'RE STILL DISCUSSING REQUIREMENTS. DOES ANYONE ELSE SEE A PROBLEM HERE?
WHEN YOU'RE DONE, CAN WE TALK ABOUT REQUIREMENTS?
www.dilbert.com scottadams@aol.com
© 2002 United Feature Syndicate, Inc.
4/4/02

YOU SCHEDULED THE END OF THE TEST PHASE AFTER THE START OF THE PRODUCTION PHASE.
WE'RE FEELING CONFIDENT.
IT'S TOO BAD THAT BEING SMART DOESN'T COME WITH SOME SORT OF GOOD FEELING LIKE THAT.
Dilbert.com DilbertCartoonist@gmail.com
5-17-11 ©2011 Scott Adams, Inc./Dist. by Universal Uclick

Software engineering is not a "one size fits all"

In accordance to Capers Jones analysis:

- Software methods depend on the size of the system under development
  - Methods that might be ranked as "best practices" for small programs of 1,000 function points in size may not be equally effective for large systems of 100,000 function points in size.

- Tools, languages, and methods are not equally effective or important for all activities
  - For example a powerful programming language will obviously have beneficial effects on coding speed and code quality. But which programming language is used has no effect on requirements creep, user documentation, or project management

Quality leads and Productivity follows

Quality needs to be improved faster and to a higher level than productivity in order for productivity to improve at all.

- The reason for this is that finding and fixing bugs is overall the most expensive activity in software development.
- Attempts to improve productivity without improving quality first are not effective.

The methods and practices don't occur in isolation

- There are "patterns" of common practices that tend to occur together.
- It is interesting that top-gun projects and companies use best-practice patterns and seldom mix in poor practices.
- Average and lagging groups, on the other hand, tend to use ineffective patterns and few of the more powerful and effective methods.

The impact of best and worst practices

- Best, average and worst for the new development of a large system of 10,000 function points and 550,000 Java code statements
- Telecommunications billing system
- Large systems in this size range are very hazardous and have a high incidence of schedule slips, cost overruns, and outright cancellation.

To reduce the number of variables and focus on methods, in all three cases:

- Averaged 140 work hours per person month
- “Average” personnel
- Java programming language
- Total staff of 50 FTEs in all three cases
- No requirements creep
- Same geographic area and cultural background (USA)
- Co-located teams

The three cases concentrate on:

- Development methods and
- Quality Control methods

CASE 1 – BEST CASE – Master Data

Country code:	001	United States
NAIC industry code:	82	Telecommunications
Regional code:	NJ	New Jersey
Application nature:	010	New development
Application scope:	270	Departmental system
Application class:	170	Unbundled, commercially marketed
Application type:	160	Telecommunications
Application platform:	120	Mainframe

- Team Software Process (TSP)
- Joint application design (JAD)
- Inspections of requirements, design, code
- Static analysis prior to testing
- Mathematical test case design
- Certified test personnel
- Effective project office
- Accurate parametric estimates before starting
- Accurate tracking and status reports
- Large positive ROI

CASE 2 – AVERAGE CASE – Master Data

Country code:	001	United States
NAIC industry code:	582	Telecommunications
Regional code:	AZ	Arizona
Application nature:	010	New development
Application scope:	270	Departmental system
Application class:	170	Unbundled, commercially marketed
Application type:	160	Telecommunications
Application platform:	120	Mainframe

- Agile/Scrum
- Embedded users
- Pair programming
- Test-driven development
- No inspections of requirements, design, code
- No static analysis prior to testing
- No mathematical test case design
- No project office
- Uncertified developer test personnel
- Inaccurate manual estimates before starting
- Partial tracking and informal status reports
- Slightly positive ROI

CASE 3 – WORST CASE – Master Data

Country code:	001	United States
NAIC industry code:	582	Telecommunications
Regional code:	GA	Georgia
Application nature:	010	New development
Application scope:	270	Departmental system
Application class:	170	Unbundled, commercially marketed
Application type:	160	Telecommunications
Application platform:	120	Mainframe

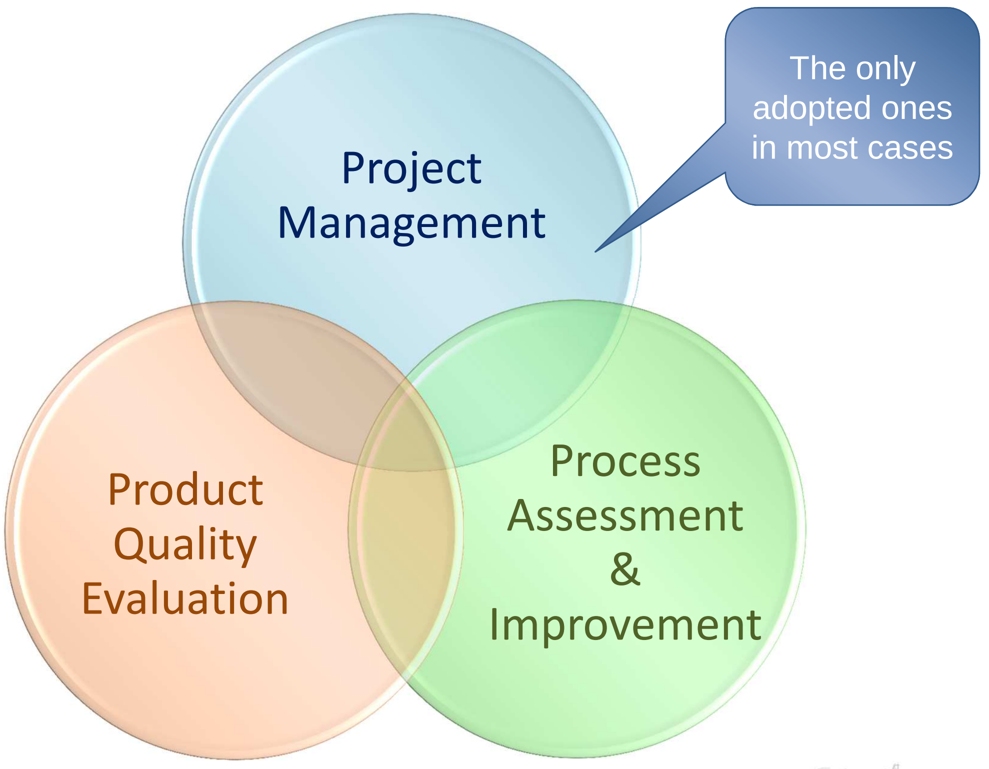
- Waterfall/Cowboy
- Informal partial requirements
- No test-driven development
- No inspections of requirements, design, code
- No static analysis prior to testing
- No mathematical test case design
- No project office
- Uncertified developer test personnel
- Testing truncated to “meet schedule”
- Grossly optimistic manual estimates
- Status tracking conceals major problems
- Negative ROI due to schedule and cost overruns

BENCHMARK – Quantitative Data

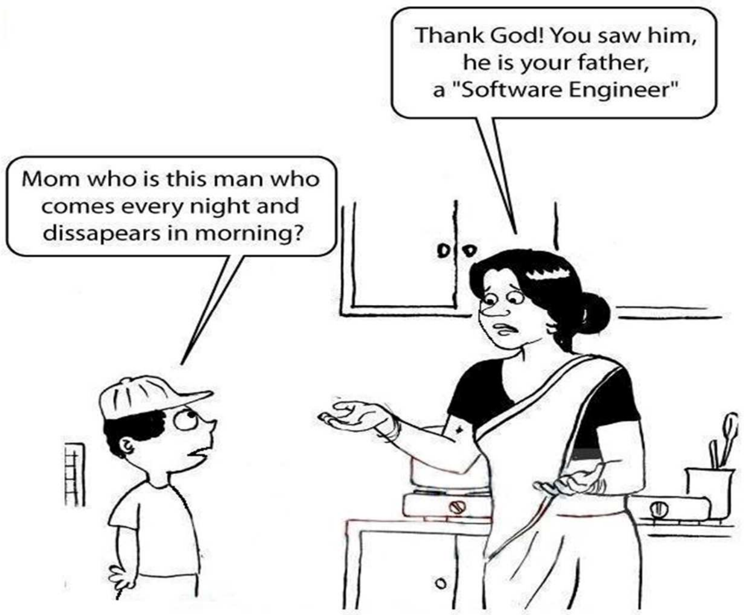
	BEST	AVERAGE	DELTA	WORST	DELTA
Schedule (calendar months)	24	28	16,7%	36	50,0%
Effort (work hours)	160000	196000		252000	
Effort (p-months - 140 hours per month)	1143	1400		1800	
Work hours per function point	16	19,6		25,2	
Function points per month	8,75	7,14		5,56	
LOC per month	481	393		305	
Development Costs	$11,430,000	$14,000,000		$18,000,000	
Costs per function point	$ 1,14	$ 1,40	22,5%	$ 1,80	57,5%
Defect potential	35000	40000		55000	
Defect potential per function point	3,5	4		5,5	
Defect Removal Efficiency	98,50%	95,50%		89,00%	
Delivered defects	525	1800	242,9%	6050	1052,4%
Delivered defects per function point	0,05	0,18		0,6	
High-severity defects	65	270	315,4%	1089	1575,4%
Security flaws	3	25		97	
Technical debt (*)	$385,000	$1,250,000		$8,550,000	
Technical debt per function point	$ 38,50	$ 125,00	224,7%	$ 855,00	2120,8%

Technical debt: economic impact of consequences of poor system design, sw architecture or sw development. The debt can be thought of as cost of the work that needs to be done before a particular job can be considered properly completed

Adoption of metrics


The only adopted ones in most cases
Project Management
Product Quality Evaluation
Process Assessment & Improvement

After all that we have been through ….. We are still in heroic mode


Thank God! You saw him, he is your father, a "Software Engineer"
Mom who is this man who comes every night and dissapears in morning?

Internet Explorer - Logoff Warning

You have been on-line for 1 year.

Do you wish to Log Off and get a Life?

Yes NO

Remind me next year

Europe's preferred Technology Consulting and Engineering firm.

Gualtiero Bazzana
Gualtiero.Bazzana@Alten.it