1°International Conference on
IT Data collection, Analysis and Benchmarking
Rio de Janeiro (Brazil) - October 3, 2013

KPIs used in a 6,000 Function Points Project

Márcio Silveira, PMP
HP-Enterprise Services-ABS
marcio.silveira@HP.com

Presentation Objectives

- ✓ Provide background information about the 6,000 FP project
- ✓ Describe 9 areas of metrics collection
- ✓ Show some KPIs in each area
- ✓ For each area present:
  - ➢ Analysis techniques using real data about the project performance
  - ➢ Make some considerations about the project performance
- ✓ Share lessons learned about the project

Agenda

- Project Background
- About the data and analysis
- Project KPIs Overview
- KPIs
  - Size
  - Duration/Effort
  - Staff
  - Changes
  - Productivity
  - Defects
  - Use Cases
- Metrics Summary
- Lessons Learned
- Last Thought ...
- Q&A

Project Background

- The main propose of the project was HP to develop the detailed design, construction, unit test, functional test and support integration test and assisted operation of 6,500 FP application written in Java on a fixed price/fixed date type of contract charged in R$ xxxx per 1 Function Point delivered.
- 6,500 FP application divided in 28 modules (after a rebaseline reduced to 17). Development strategies : waterfall and agile
- The function point counting was done by the customer/3rd party company based on the requirements that were gathered at that time.
- All requirement definition, analysis and design was done by the customer.
- Initial estimate done using customer sizing information :
  - SLiM Tool from QSM used to estimate each module
  - Historical data used from ISBSG, HP and QSM database
  - Scope creep considered as 10%
  - SLiM interative lifecyle used. Later on some modules adopted an "Agile/Jad" approach
  - Duration : 14 months
  - Person Months : 551
  - Productivity : 10,6 Hours per Function Points
- Project sized only in functional user requirements (FUR-FP). Non-functional requirements (NFRs) were not sized . See a whole discussion on NFRs at www.semq.eu/pdf/fsm-prod.pdf

About the data and analysis

- The data used was baselined on August 31, 2012 where the vast majority of the modules were finished based on the scope of the work.
- In some cases the data was compared with internal benchmarking data and also with external benchmarking data (ISBSG Release 11 – International Software Benchmarking Standards Group, www.isbsg.org) and Caper Jones benchmarking data from : "APPROXIMATE INTERNATIONAL SOFTWARE QUALITY LEVELS EXPRESSED IN IFPUG FUNCTION POINTS Version 2.0 12/06/2011

Project Metrics Overview – Overall Performance

Metrics Summary

•Below we can see some basic metrics and KPIs that can provide a general overview about the project and its performance.

Area	Metrics	Unit	Value	Observations
Size Related (Functional Size)	Project Initial Size - Baseline	Function Points (CPM 4.3.1)	3.510	
	Project Final Size	Function Points (CPM 4.3.1)	3.502	
	Project Change Requests	Function Points (CPM 4.3.1)	2.527	
	Total Project Size	Function Points (CPM 4.3.1)	6.037	
	Project Final Size	Logical Lines of Code	676.222	
Duration Related	Project Duration	Months	20	Until the end of UAT
Effort Related	Total Effort	Hours	277.376	
	Total FTE	Number	1.734	1 FTE = 160 hours
	Average Max hours per Resource	Hours	202	
Staffing Related	Peak Staff	Number	156	
	Average Hours per Resource	Hours	158	
Change Related	Total Change Requests (CR)	Number	259	
	Overall Scope Creep	Percentage	72%	
	Average size of Function Points per CR	Function Points	10	
Productivity Related	PDR (Project Delivery Rate)	Hours/Function Points	47,1	
	Function Points per FTE	Function Points/FTE	3,5	
Defect Related	Defect Density (Defects/FP) - UAT	Ratio	0,03	
	Defect Retention - UAT	Percentage	91,3%	
	Defect Retention - Pos-Implementation	Percentage	98,6%	
Use Cases Related	Number of Use Cases (UC)	Number	172	
	Average of Function Points Per UC	Number	20	
	Max Function Point Size of an UC	Function Points	98	

Sizing Related – Modules in The Scope

Modules	Project Size - Reviewed (Function Points)	Project CRs (Function Points)	Project Size - Reviewed+CRs (Function Points)	Project Size (Lines Of Code)	Installed Application (Function Points)
Module-1	342	638	980	105.295	342
Module-11	423	532	955	107.209	423
Module-9	390	119	509	33.821	418
Module-12	294	208	502	40.354	354
Module-7	339	144	483	40.548	248
Module-3	301	89	390	21.045	237
Module-4	216	164	380	32.052	220
Module-17	197	164	361	32.673	209
Module-5	130	138	268	12.114	146
Module-15	174	52	226	27.347	190
Module-8	163	53	216	11.299	129
Module-14	114	55	169	17.643	131
Module-13	118	48	166	10.342	122
Module-10	107	49	156	9.709	129
Module-2	81	57	138	16.556	81
Module-16	79	18	97	9.000	81
Module-6	42	0	42	2.630	42
Common_Modules	Not Counted	Not Counted	Not Counted	146.585	Not Counted
Total	3.510	2.527	6.037	676.222	3.502

- We had two "big modules" over 900 FPs, but in reality more than 50% of the module´s size was due to changes primarily due to requirements change.

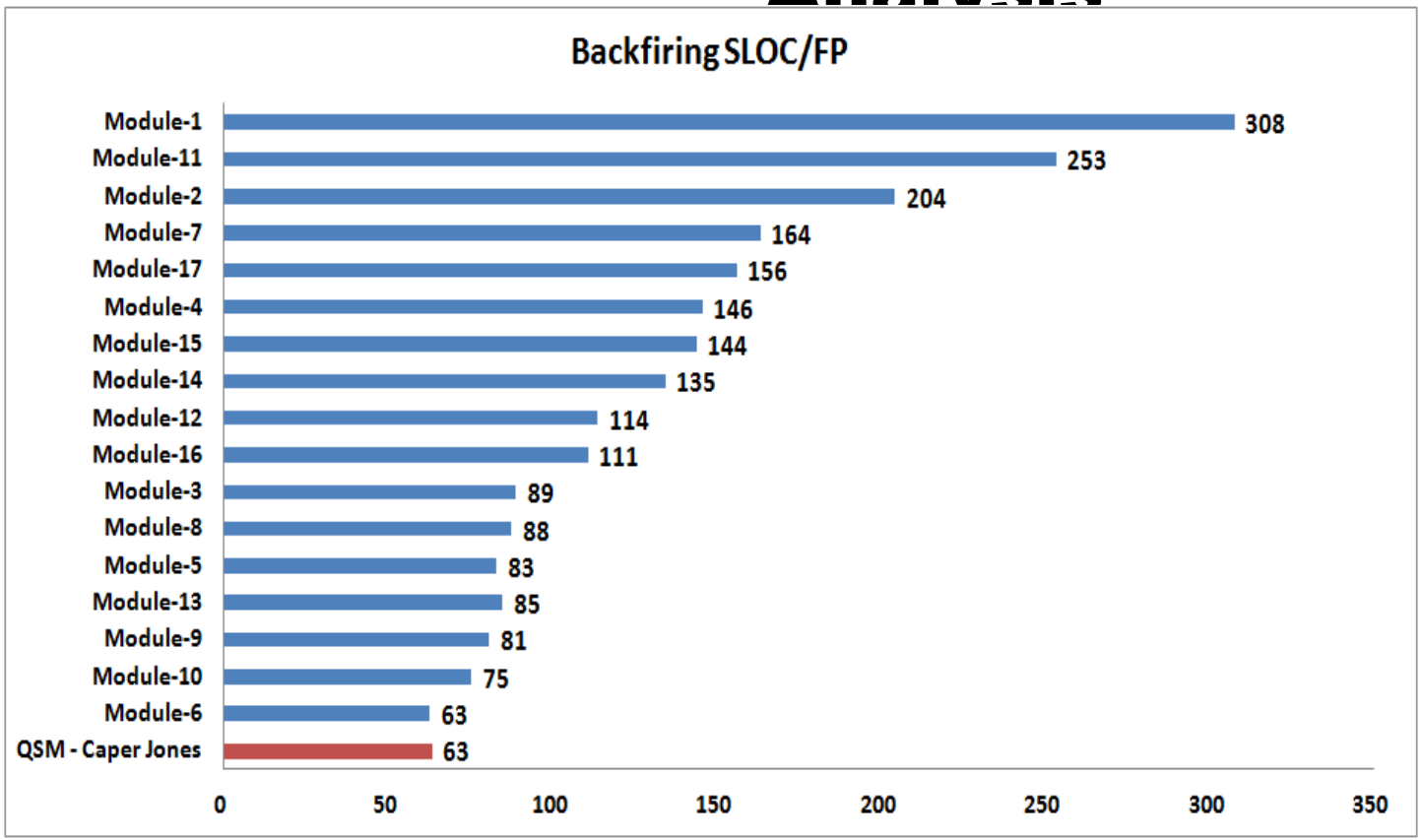
Sizing Related – Controlled Backfiring Analysis

- It is possible to establish a relationship between Function Points (FP) and Logical Lines of Code(SLOC), this is commonly named "backfiring".
- Even though this relationship can vary a lot between different languages and organizations in this project we are at the same organization, using the same languages and processes so this relation may tell us some interesting information since we can considered this a controlled environment.
- For the purpose of benchmarking (of course considering all the issues above) several estimating tools and IT experts use the following relation for Java applications:

  1 FP = 63 Lines of Code for Java

- Base on this we will do some analysis over the size data that we collected (FP and SLOC)
- For more information about controlled backfiring you can read the paper : "'Controlled Backfiring' from Carol Dekkers and Ian Gunter (http://www.qsma.com/pdfs/ITMSVol_VI11.pdf)

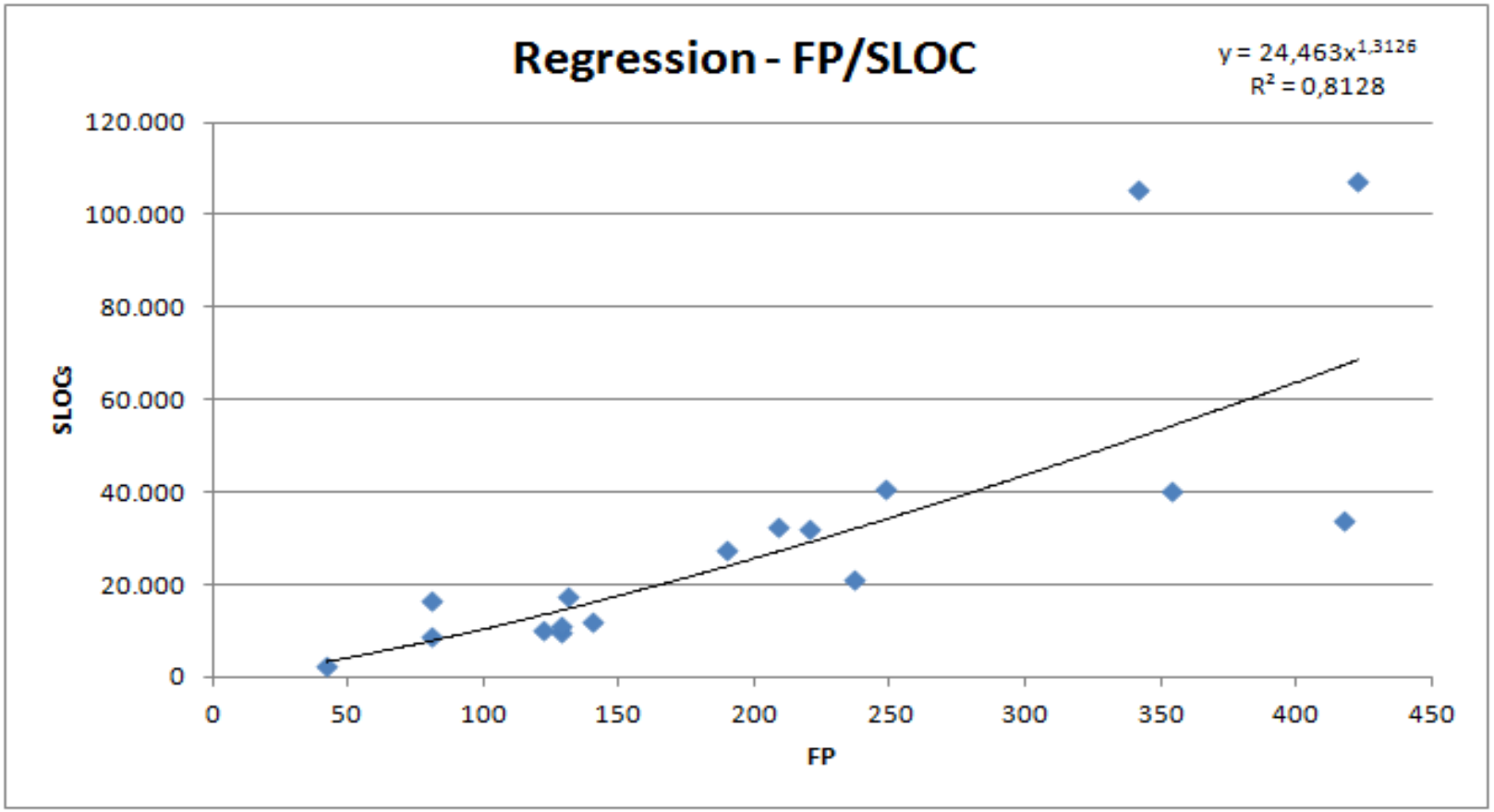
Sizing Related - Controlled Backfiring Analysis



	SLOC/FP
Average	135
Median	114
Stdev	67

- There is a set of modules where there is a good approximation of the market SLOC/FP "standards". These modules are the one where the relation is around 60-90 SLOCs/FPs. Probably in these modules there is less business complexity and non-functional requirements.
- An intermediate set of modules showing a relation 100-200 SLOCs/FP possess a good variation against market standards. Probably the business complexity was moderate and some non-functional requirements were present.
- The last group, 200-300 SLOCs/FPs, were really far from the market standards. They probably have a high business complexity and many non-functional requirements. So when estimating Change Requests or Enhancements in modules of the second and third group extra caution must be taken to factor this complexity in the estimate.

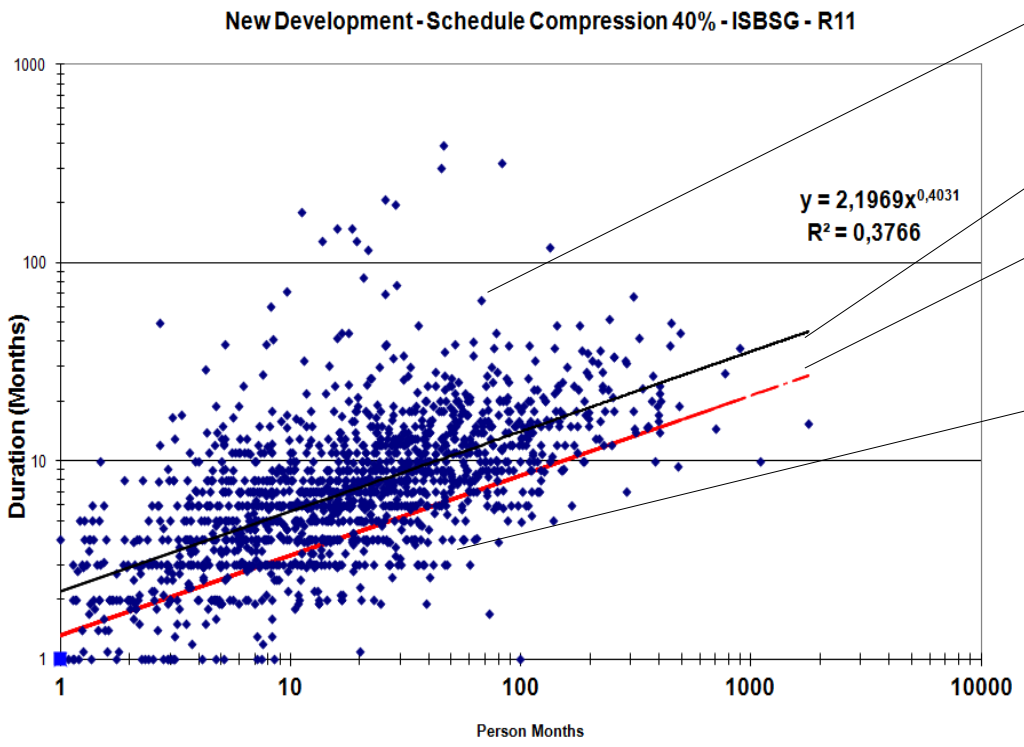
Sizing Related – Function Points Predictor of SLOCs



• The behaviour of size data (FP and SLOC) show us that Function Points can be a good predictor of Lines of Code if you use a power regression. However caution must be taken for modules over 250 FPs or 35KSLOC since the variation maybe considerable in the project.

Effort/Duration Related – "Impossible Zone" Analysis

- The "Impossible Region" is a study of schedule compression. That is, determining the shortest possible time in which a given amount of effort can be completed.
- The edge of the "Impossible Region" marks the point at which schedules can no longer be compressed (40%).
- In order to understand the correlation between effort and duration, linear correlation technique was used. A generally accepted strong correlation is considered when the correlation factor R^2 >=0,60, Adjusted-R^2 = +/- 10% and P-Value < 0,005



Projects

Nominal Duration

Over 40% schedule compression

Even though a project here seems to be contradictory people do crazy things, sometimes pressured by the customer, price to win approach, or business requirements. The projects came from ISBSG R11. Usually running projects with level of schedule compression results in :

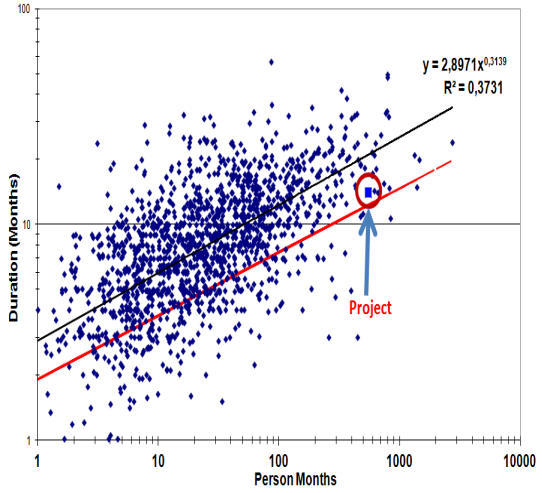
- **Quality issues**
- **Productivity issues**
- **Excessive overtime**
- **Budget overrun**

Observation : Generally accepted correlation factor : R^2 >=0,60, Adjusted R^2 +/- 10% and P-Value < 0,005

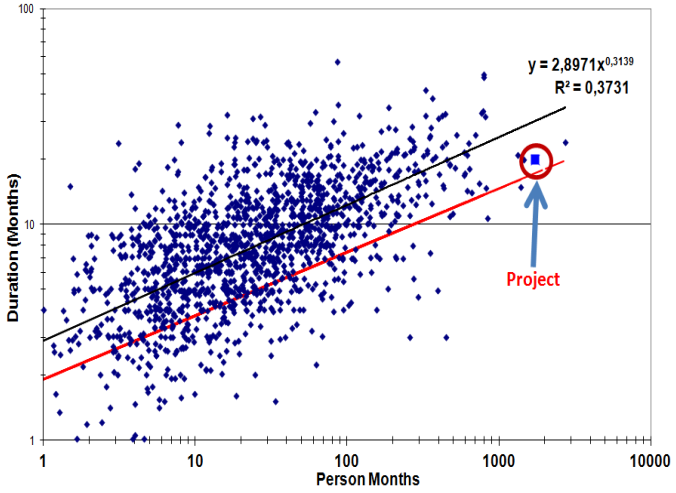
Effort/Duration Related – "Impossible Zone" Analysis

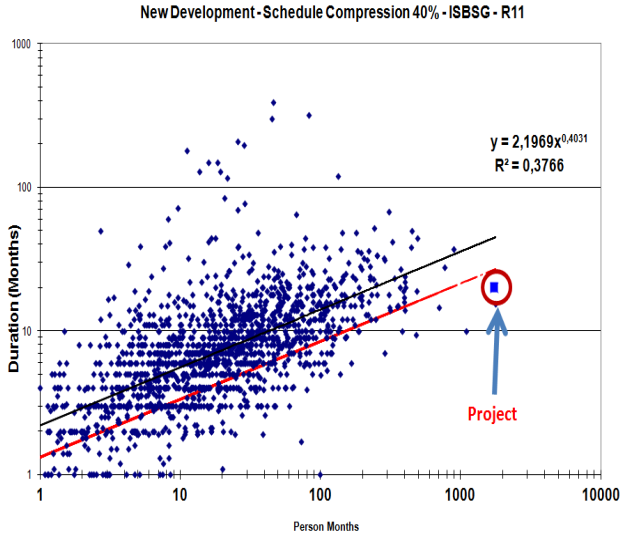
- Analysis was performed using the data from the initial proposal and project actual at the end of August 2012.
- Benchmarking performed against HP data as well as ISBSG data

Proposal August 27, 2010



Actuals August , 2012

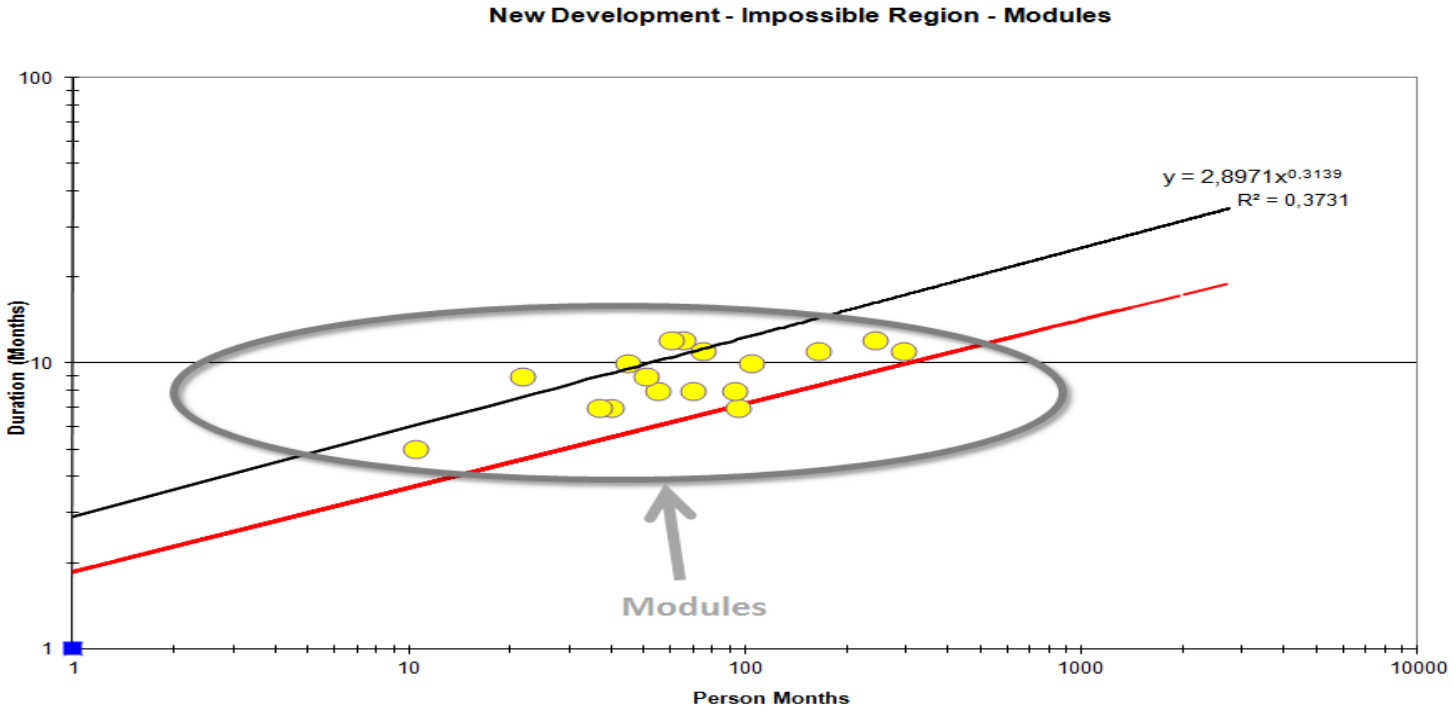




- As can be seen the project was not a typical project since its inception. The project was unique, very big in terms of effort to be spent but it transformed itself in a very challenging project.
- Based on HP and ISBSG historical data both confirmed that we operated at or below the impossible zone.
- The ideal duration projected by both source would be :
  - HP : 29,3 Months
  - ISBSG : 45,6 Months
- So deliver this amount of effort in 20 months was certainly a big challenge.

Effort/Duration Related – Impossible Zone Analysis

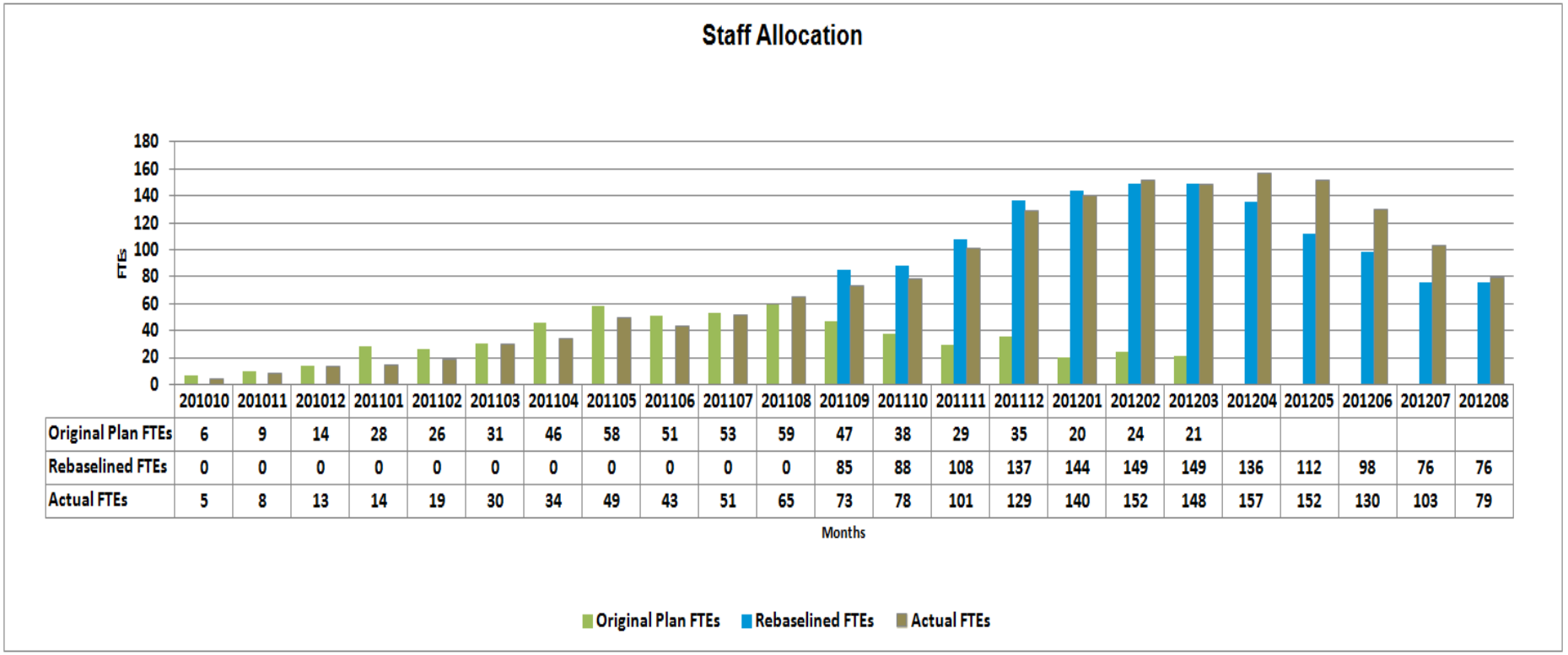
•The analysis below shows how was the schedule compression for the 17 modules



• As can be seen a great deal of modules (yellow bubbles) were near to the "impossible zone" where projects generally start to suffer of issues with productivity, quality, predictability and others.
• Confirming the previous slide the schedule compression seen for the overall project repeats at several modules.

Staffing Related - Staff Allocation

- This Bar Chart depicts the staff allocation of the project. It contains the original staff plan, the re-baseline that was necessary due to project´s delays and the actuals.



	201010	201011	201012	201101	201102	201103	201104	201105	201106	201107	201108	201109	201110	201111	201112	201201	201202	201203	201204	201205	201206	201207	201208
Original Plan FTEs	6	9	14	28	26	31	46	58	51	53	59	47	38	29	35	20	24	21					
Rebaselined FTEs	0	0	0	0	0	0	0	0	0	0	0	85	88	108	137	144	149	149	136	112	98	76	76
Actual FTEs	5	8	13	14	19	30	34	49	43	51	65	73	78	101	129	140	152	148	157	152	130	103	79

- Looking the staff plan is clear that the project that we planned it was not the project that was delivered.
- This staffing plan also show us that we doubled the size of the team in 4 months and in another 4 months we added another 50% meaning that we almost tripled the number of the FTEs from the original plan. This is a notable fact since all the literature about adding people to a project says that if you add people in a troubled project, the project will get worse.

Change Related - Scope Variation

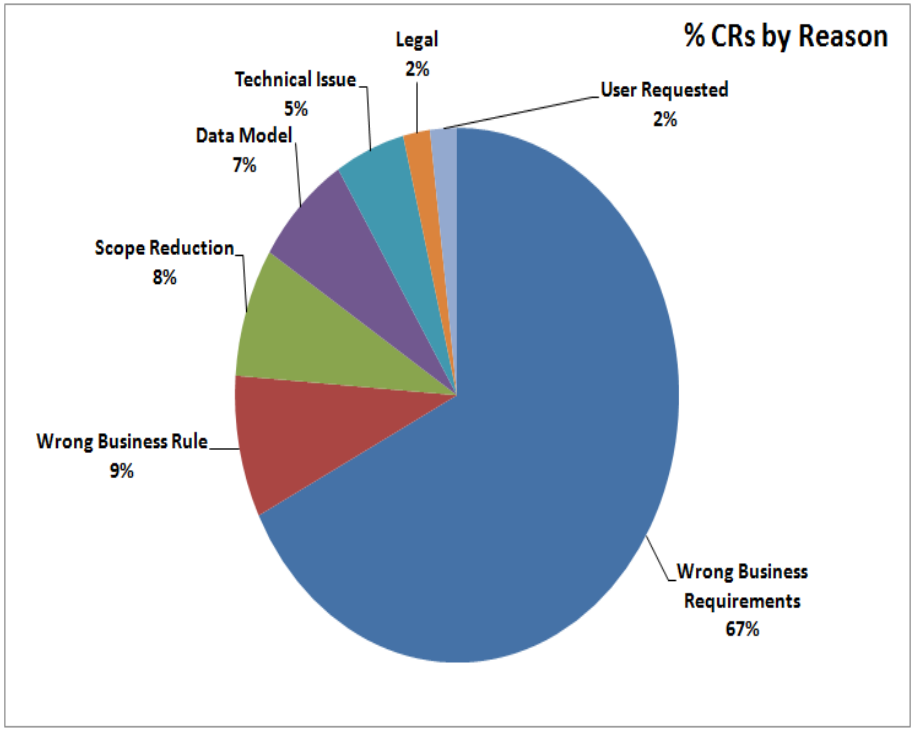
Modules	Initial Project Size	Project Re-baselined Size	# Change Requests	Total Size Change Requests	Module Size	Scope Variation
Module-1	602	342	62	638	980	186%
Module-11	908	423	46	532	955	126%
Module-5	345	130	15	138	268	106%
Module-17	371	197	11	164	361	83%
Module-4	234	216	9	164	380	76%
Module-12	406	294	20	208	502	71%
Module-2	210	81	10	57	138	70%
Module-14	199	114	8	55	169	48%
Module-10	128	107	2	49	156	46%
Module-7	470	339	13	144	483	42%
Module-13	270	118	9	48	166	41%
Module-8	110	163	10	53	216	33%
Module-9	400	390	21	119	509	31%
Module-15	451	174	10	52	226	30%
Module-3	270	301	9	89	390	30%
Module-16	80	79	3	18	97	22%
Module-6	91	42	1	0	42	0%
Totals	5.545	3.510	259	2.527	6.037	72%

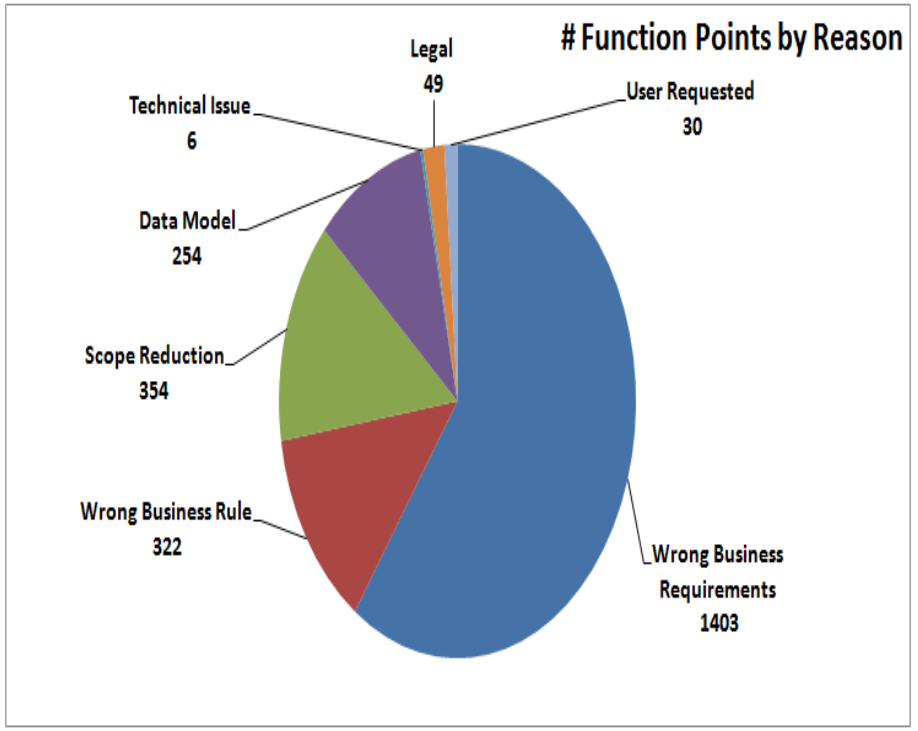
•The action to reduce the scope of the project was not effective because during the project all the modules suffered of scope creep. The most complex modules, Module-1 and Module-11, sufferred with variations over 100% compared with the initial size.

• A total of 259 CRs were generated. In average a module had 15 CRs, each one with an average size of 10 Function Points.

• Considering ISBSG data about the ideal duration of each module and the normal scope creep that projects suffer (1%-2% per month), the project should have around 1214 Function Points of changes against 2,527 Functions Points, more than 2 times than usual.

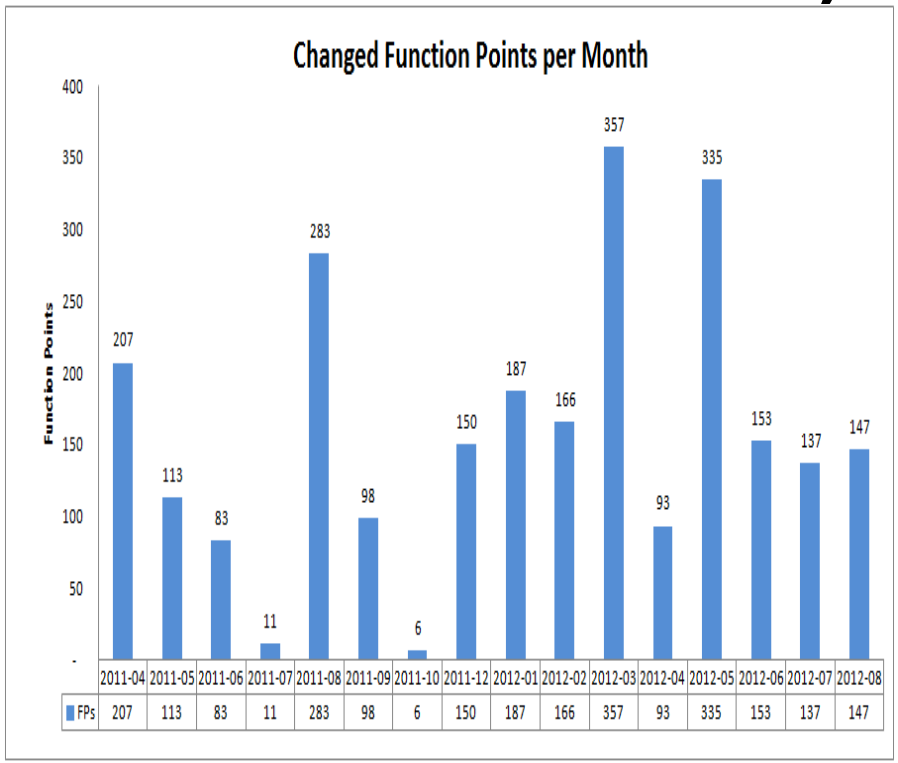
Scope Variation – CRs Reason Stratification

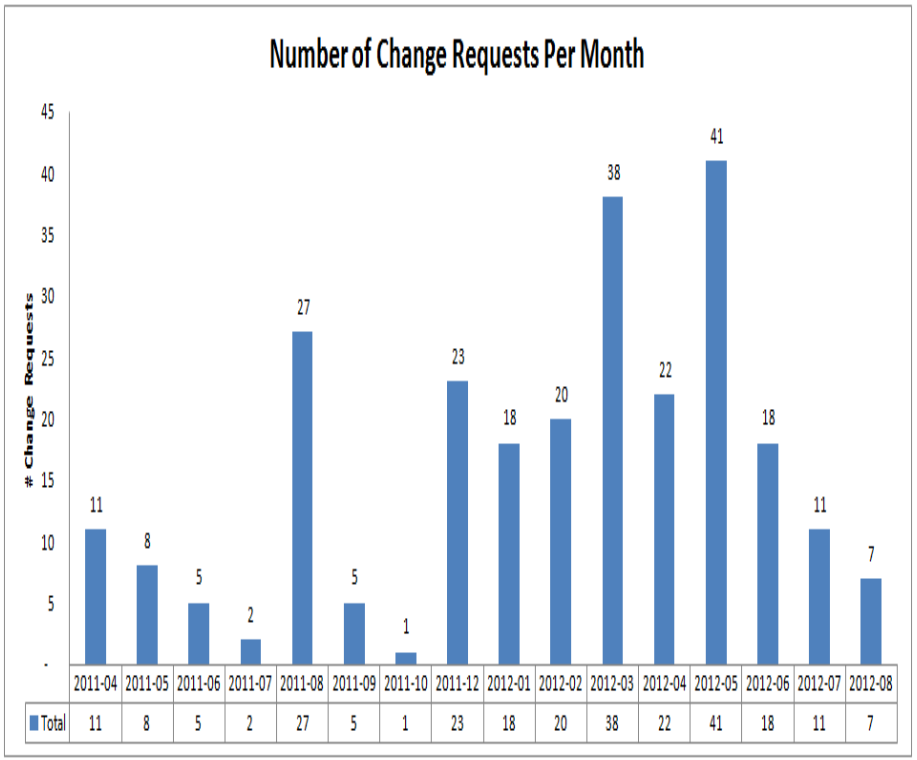




• Approximately 76% of the CRs were due to wrong business requirements or rules, corresponding to 1.725 Function Points in changes to the project (71% of all the changes in Function Points).

cope Variation – CRs Analysis (FP Changed and # CRs)



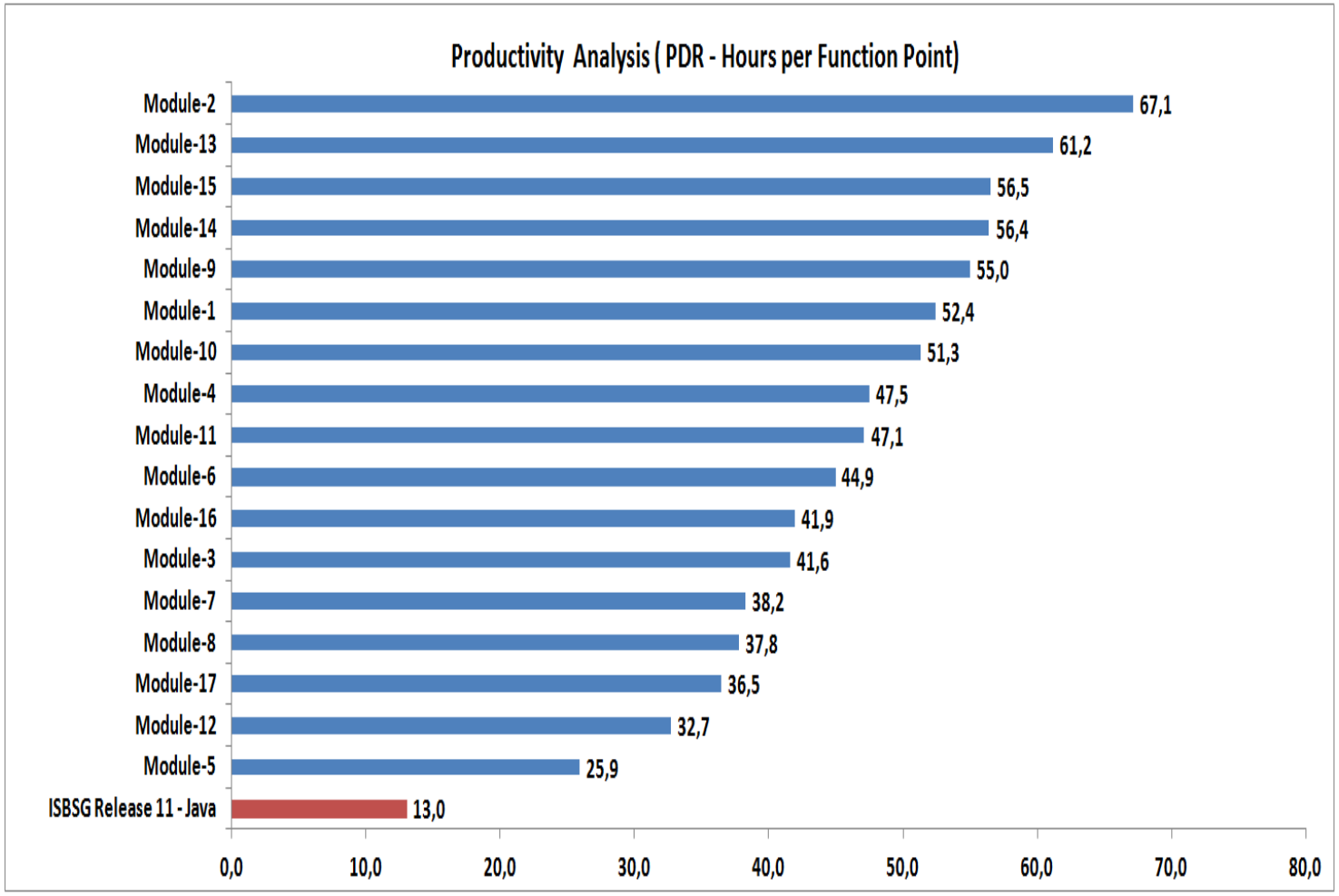


Average	Median	Stdev
158	148	100

Average	Median	Stdev
16	15	12

- These Change Requests charts show us several things at left hand side we have the amount of changed Function Points per month and at the right hand size the number of Change Requests. So grossly, as stated before we had 150 Changed FP per month and we needed to analyze approximately 15 CRs per month, one each two days.
- Another very important aspect that can be seen is the vast majority of the CRs concentrated in the last year of the contract, suggesting that requirements we not mature enough when the project started.

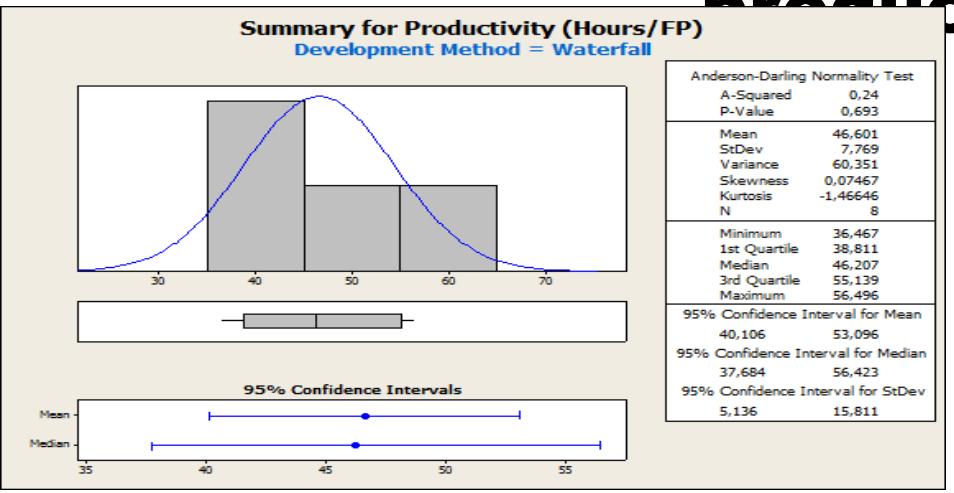
Productivity Related – PDR – Hours Per Function Points

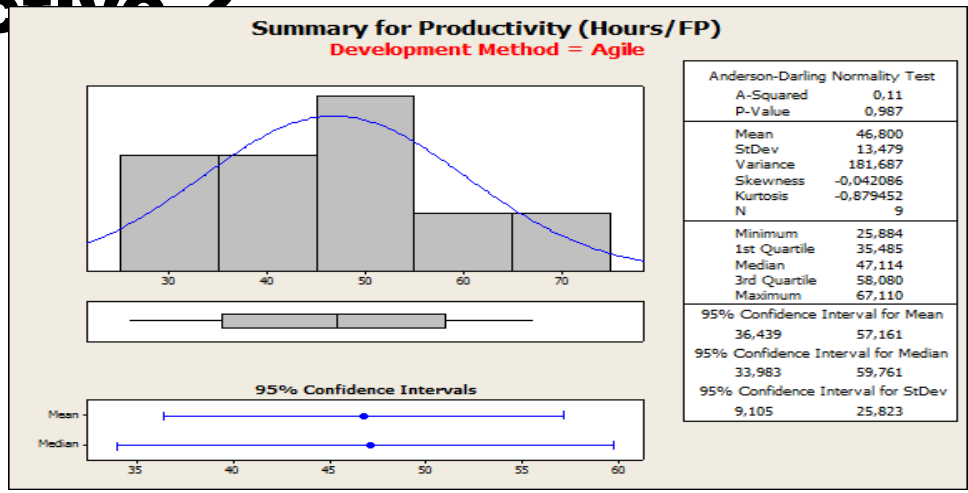


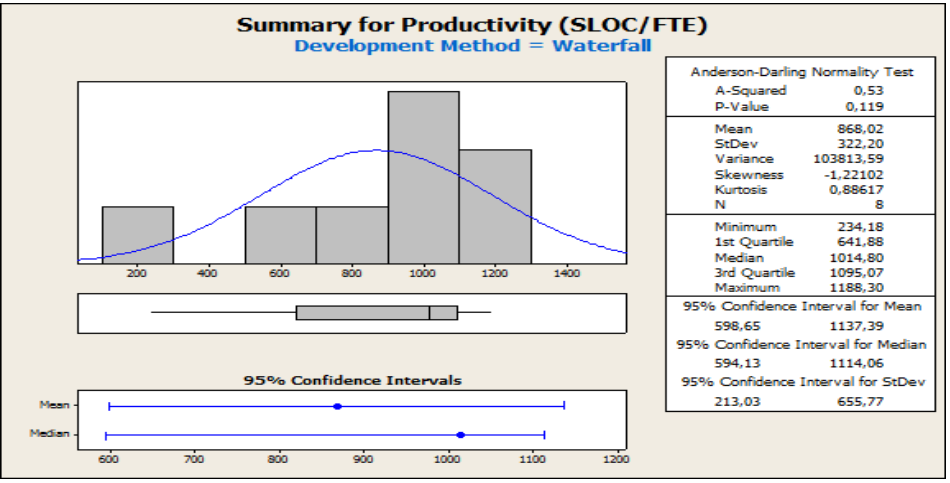
	Hours Per Function Points
Average	46,7
Median	47,1
Stdev	10,8

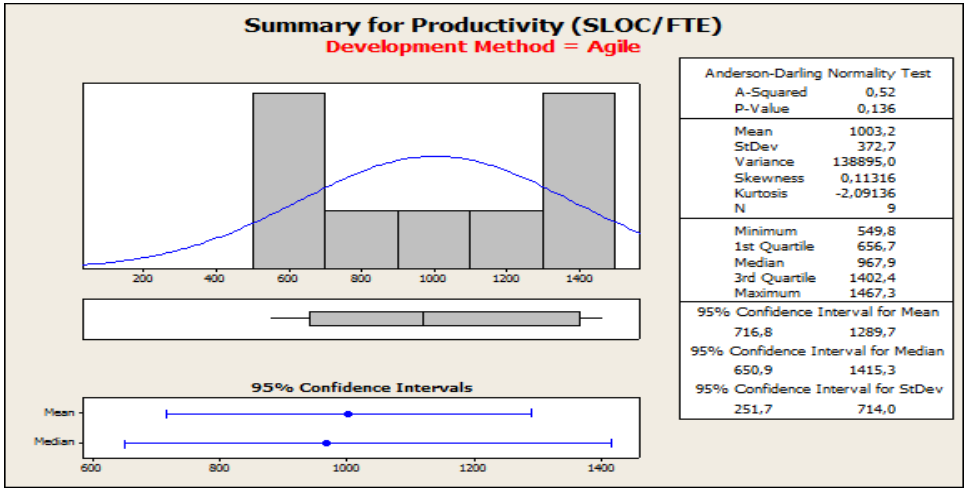
- The productivity (hours/FP) was well below external benchmarking data (ISBSG-R11).
- Another important aspect is that the best productivity (Module-5) was almost 3 times better than the worse productivity (Module-2).
- Please note that PDR = 1/Prod and Prod = 1/PDR

Productivity Analysis – Was Agile method more productive ?





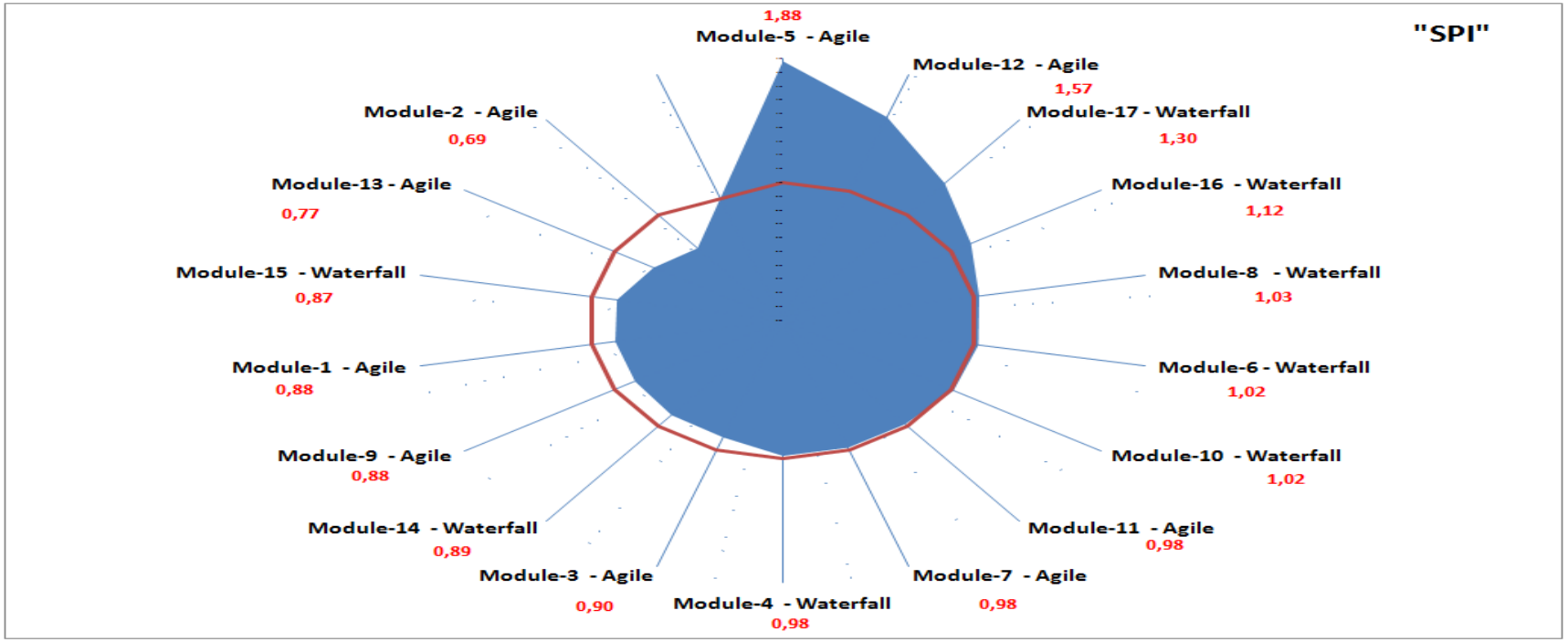




- From the perspective of productivity mean (hours/FP), Agile was slightly worse than Waterfall and presented more variation in the process.
- Concerning SLOC/FTE Agile had a better productivity mean (200 SLOC better) than Waterfall.
- The fact that Agile presented better productivity in SLOC/FTE can be explained by the fact that requirements and code were evolving at the same time, and a possible reason that the phenomena didn´t happened for hours/FP, resides the fact that the "Agile" strategy was used for requirements improvement causing rework.

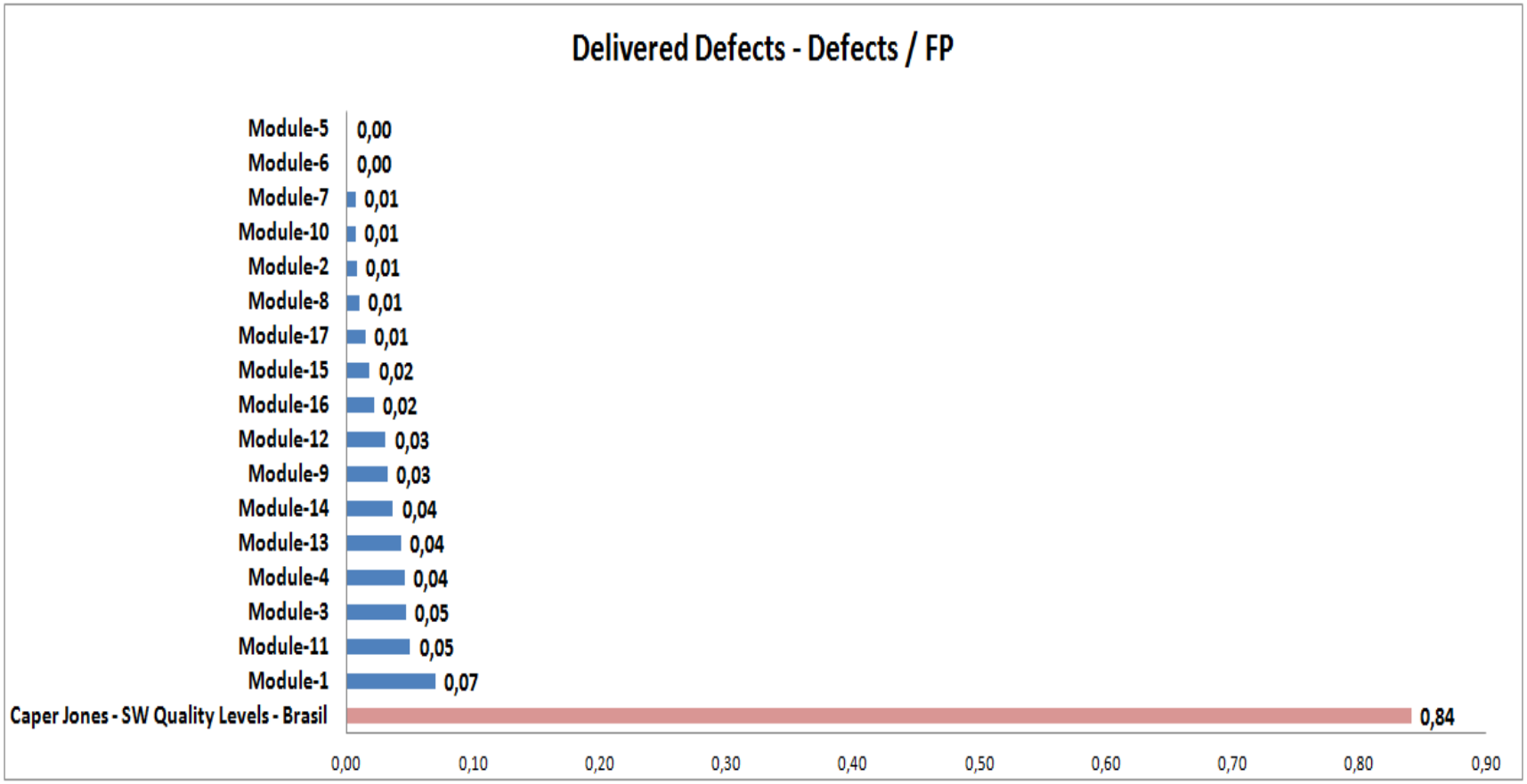
Productivity Related – “SPI” Performance

- The idea on this analysis is to see if the modules contributed more/less time to aggregate function points to the overall project. This is not exactly and SPI (Schedule Performance Index) but has a similar concept.
- Original SPI formula : SPI = EV / PV
- Our SPI = % of module size (FP) contribution to the overall size / % of module effort contribution to the overall effort



• This “SPI” measure is totally related with the productivity (Hours/FP) that we saw before but here we can see in more detail the performance of each module. In the case of “Module-5” even though its productivity (hours/FP) was not good, if compared with external benchmarking data, the project was able to perform 88% better if compared with modules with the “SPI” near to 1,00.

Defect Related – Delivered Defects Analysis

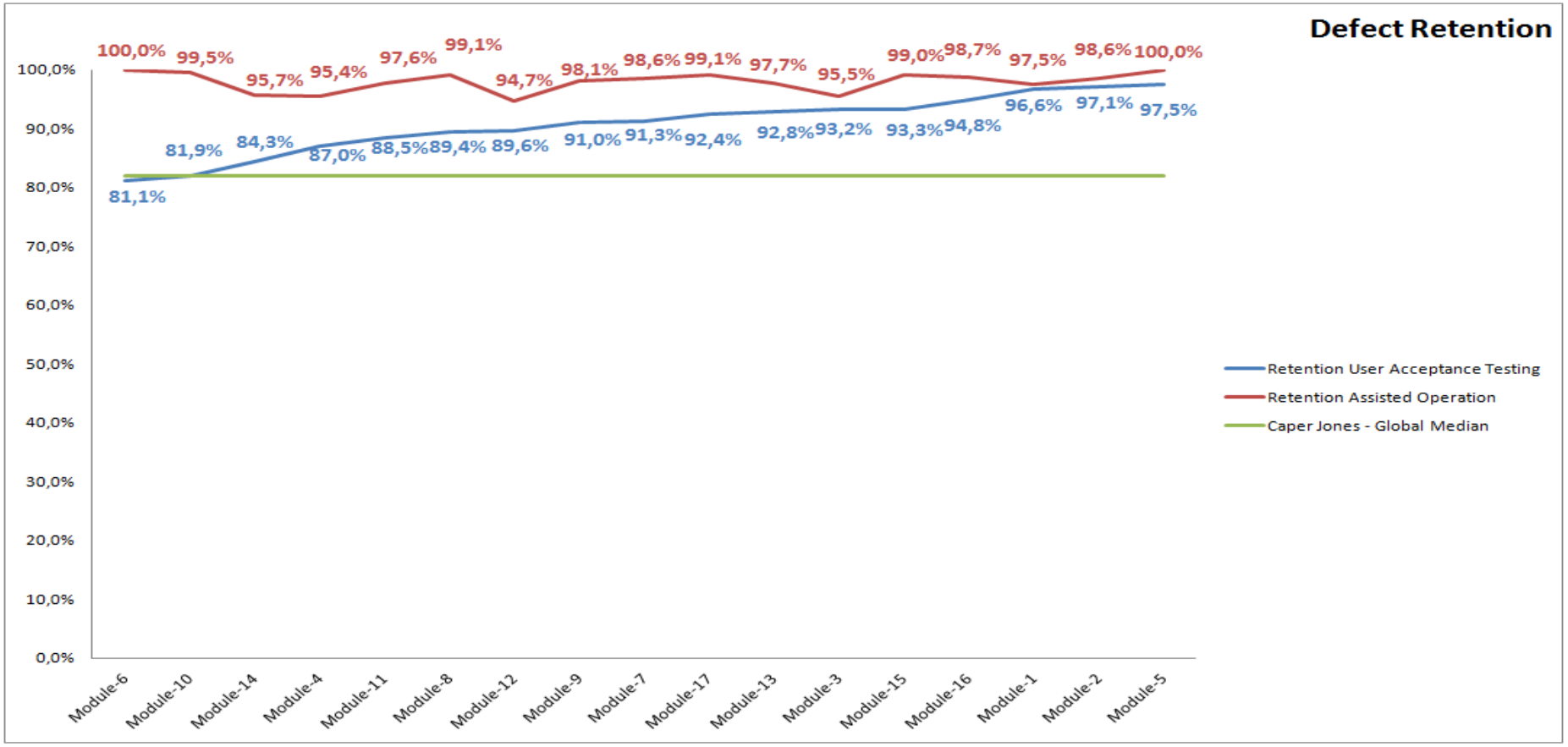


Caper Jones benchmarking data came from : "APPROXIMATE INTERNATIONAL SOFTWARE QUALITY LEVELS EXPRESSED IN IFPUG FUNCTION POINTS Version 2.0 12/06/2011

- As can be seen in the chart above the project had a very good performance related to the amount of defects delivered to the user after the application was installed in a period of 90th days.
- This is very interesting considering all the schedule compression, scope creep, peak staff, etc.
- Maybe next slide can provide some light on this paradox.

Defect Related - Defect Retention Analysis

- The definition of defect retention rate is the percentage of defects detected before the next phase. For the purpose of this analysis we will use the retention rate until UAT and until Assisted Operation



- As can be seen in the chart the major reason for the low delivered defect rate was the high defect retention before the application was installed.
- The retention rate of almost all the modules is greater than the Global Median provided by Caper Jones that is 82%. His definition considers all defects detected before the project is under production and 3 months of operation.

Use Cases Related – Project Overall

Modules-Use Cases	Total of Function Points	% of Module Size
M11-UC001	98	23%
M9-UC002	96	23%
M8-UC001	**82**	**64%**
M13-UC001	**75**	**61%**
M9-UC014	67	16%
M1-UC002	63	18%
M9-UCXXX	58	14%
M16-UC002	**56**	**69%**
M12-UC001	55	16%
M12-UC009	53	15%
M4-UC001	50	23%
M15-UC010	47	25%
M8-UC004	47	36%
M10-UC002	45	35%
M9-UCYYY	43	10%
M6-UC001	42	100%
M1-UC010/UC011/UC012/UC013	42	12%
M12-UC003	42	12%
M11-UC003	41	10%
M12-UC002	41	12%

	UC Average Size	HP Experience in Other Projects
Function Points	20	13-14

- The table shows the 20th biggest Use Cases in the project. As can be seen several of them are almost the same size of a small module.
- Having UCs too big may create unecessary complexity making the development process more difficult and certainly productivity suffers
- Another aspect is that the average size of Use Cases in the project was around 20 FPs while in other HP projects we had UCs averaging 13-14 Function Points.
- While is not 100% conclusive modules with a better performance (SPI, Defect Delivered, productivity) are not showing up on the above list or are showing up only one/two times.

Metrics Summary

Modules	Development Method	Project Initial Size (Function Points)	Project CRs (Function Points)	Project Final Size (Function Points)	Application Size (Function Points)	Application Size (SLOC)	SLOC per Function Points	Total Effort	Productivity (Hours/FP)	Scope Variation	Delivered Defects (Defect/FP)	Defect Retention After Assisted Operation	"SPI"	Average Size UCs (Function Points)	Peak Staff	# Resources worked in the module
Module-5	Agile	130	138	268	146	12.114	83	6.944	25,9	106,4%	0,00	100,0%	1,88	12	8	26
Module-12	Agile	294	208	502	354	40.354	114	16.427	32,7	70,7%	0,03	94,7%	1,57	32	7	19
Module-7	Agile	339	144	483	248	40.548	164	18.469	38,2	42,4%	0,01	98,6%	0,98	19	7	16
Module-11	Agile	423	532	955	423	107.209	253	44.973	47,1	125,7%	0,05	97,6%	0,98	18	16	34
Module-3	Agile	301	89	390	237	21.045	89	16.210	41,6	29,5%	0,05	95,5%	0,90	13	5	12
Module-9	Agile	390	119	509	418	33.821	81	28.001	55,0	30,6%	0,03	98,1%	0,88	38	8	18
Module-1	Agile	342	638	980	342	105.295	308	51.328	52,4	186,5%	0,07	97,5%	0,88	12	21	43
Module-13	Agile	118	48	166	122	10.342	85	10.167	61,2	40,9%	0,04	97,7%	0,77	41	4	9
Module-2	Agile	81	57	138	81	16.556	204	9.258	67,1	70,3%	0,01	98,6%	0,69	16	4	7
Module-17	Waterfall	197	164	361	209	32.673	156	13.154	36,5	83,1%	0,01	99,1%	1,30	17	15	30
Module-16	Waterfall	79	18	97	81	9.000	111	4.056	41,9	22,4%	0,02	98,7%	1,12	41	5	14
Module-8	Waterfall	163	53	216	129	11.299	88	8.164	37,8	32,6%	0,01	99,1%	1,03	65	15	33
Module-6	Waterfall	42	0	42	42	2.630	63	1.887	44,9	0,0%	0,00	100,0%	1,02	42	6	13
Module-10	Waterfall	107	49	156	129	9.709	75	8.021	51,3	46,1%	0,01	99,5%	1,02	26	14	26
Module-4	Waterfall	216	164	380	220	32.052	146	18.024	47,5	75,7%	0,04	95,4%	0,98	28	12	34
Module-14	Waterfall	114	55	169	131	17.643	135	9.507	56,4	47,8%	0,04	95,7%	0,89	19	13	27
Module-15	Waterfall	174	52	226	190	27.347	144	12.786	56,5	30,1%	0,02	99,0%	0,87	19	17	34
Average-AG		269	219	488	263	43.032	153	22.420	46,8	78,1%	0,03	97,6%	1,06	22	9	20
Average-WF		137	69	206	141	17.794	115	9.450	46,6	42,2%	0,02	98,3%	1,03	32	12	26

Lessons Learned

- **Size**
  - **Be careful** with a 6,500 FP project and incomplete requirements and NFRs
  - Use more than one sizing strategy (SLOCs, COSMIC, SNAP, etc.). Function Pioint is not the silver bullet
  - Get a 3rd party involved
- **Duration/Effort**
  - Assess schedule compression
  - Institutionalize time tracking since the begining
  - Re-baseline if needed
  - Factor uncertainty and probability in the estimate
- **Staffing**
  - Additional resources not necessarily lead to "chaos"
- **Changes**
  - They will happen ! Track and manage them ! Otherwise you will be in trouble
  - Size them, categorize them, analyze their behaviour

- **Productivity**
  - Use different productivity metrics, compare them, try to understand the "Whys"
  - Pre-defined productivities will not happen in the majority of the cases
- **Defects**
  - Track them, density, retention rates, delivered
  - Test, test, test, test...
- **Use Cases**
  - Small is beautiful !
- **Development Method**
  - This project is all about immature requirements.
  - "Agile" techniques were critical to improve the requirements.

Last Thought ...

Right now somewhere, a project is failing ...

Tom De Marco – Peopleware

BUT...

In Brazil miracles DO happen ... !

Márcio Silveira

Q&A

Obrigado !!!!!!!

Thank You !!!!!!!

marcio.silveira@hp.com