A Practitioner View of CMMI Process Performance Models

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Robert Stoddard and Rusty Young
March 20, 2008

Permission to use SAS JMP Screen Shots

Screen shots and other statistical tool information have been used with permission from SAS Institute. Information about JMP® statistical discovery software can be found at www.jmp.com.

JMP® is interactive, comprehensive, visual software from SAS. It dynamically links statistics with graphics right on your Windows, Macintosh, or Linux desktop, empowering you to explore data interactively and bring understanding to your organization.

Permission to use Crystal Ball and Minitab Screen Shots

Portions of the input and output contained in this module manual are printed with permission of Oracle (formerly Decisioneering). Crystal Ball 7.2.2 (Build 7.2.1333.0) is used to capture screenshots in this module.

The Web page for Crystal Ball is available at http://www.crystalball.com

Portions of the input and output contained in this presentation are printed with permission of Minitab Inc. using version 15

Minitab company web page is http://www.minitab.com

Topics

Purpose of this Tutorial

The Proposal Phase

Project Management Planning

- The Use of S curves
- Escaped Defect Analysis Modeling

Performance Models in

- Requirements
- Design
- Build
- System Test

Summary

Purpose of this Tutorial

Purpose

This tutorial is meant to inform practitioners of the:

- Essential Ingredients of CMMI Process Performance Models
- Examples of CMMI Process Performance Models across the lifecycle
- Examples of methods to implement various quantitative models for CMMI Process Performance Models

Essential Ingredients of CMMI Process Performance Models

Statistical, probabilistic or simulation in nature

Predict interim and/or final project outcomes

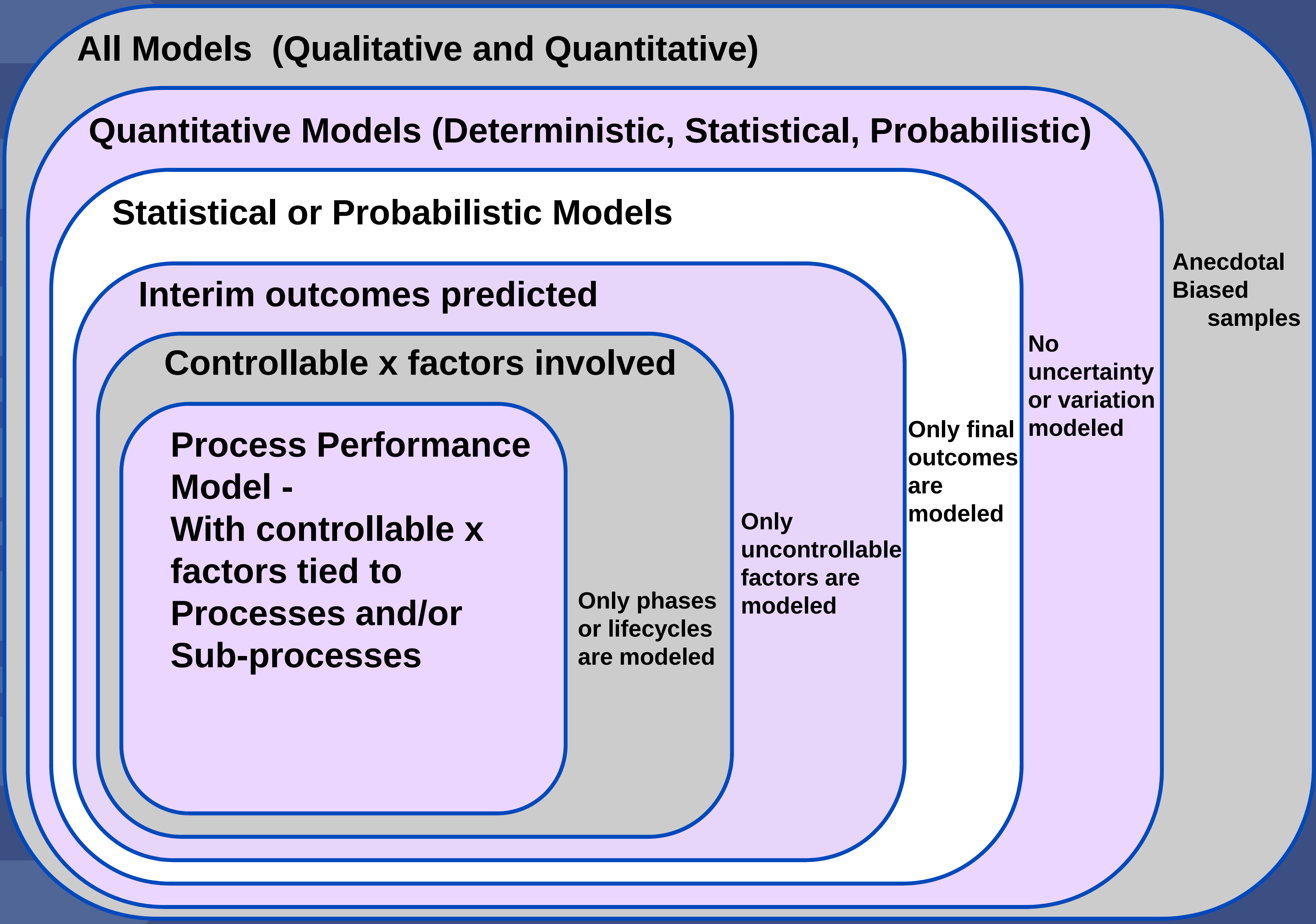
Use controllable factors tied to sub-processes to conduct the prediction

Model the variation of factors and understand the predicted range or variation of the outcomes

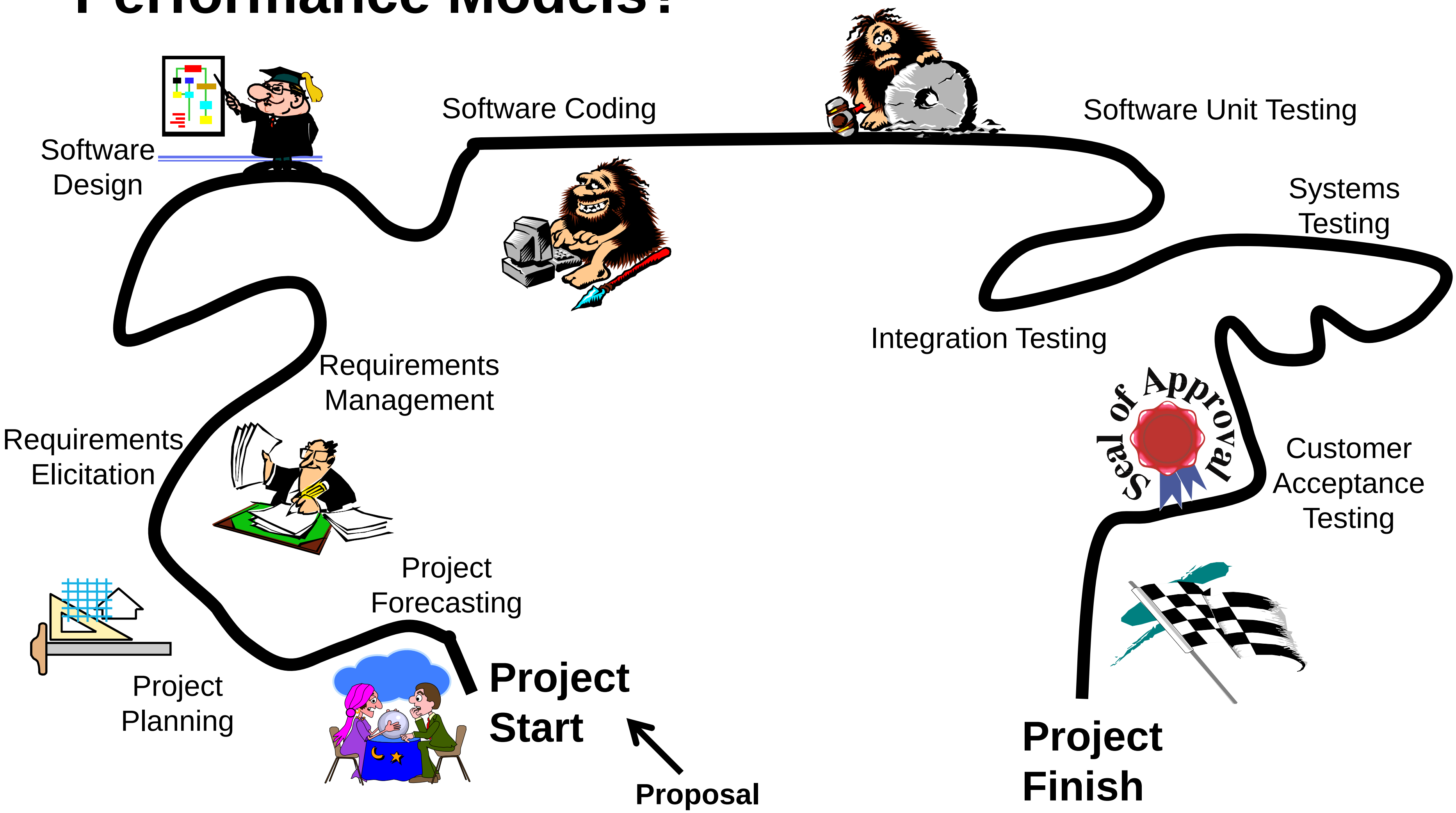
Enable “what-if” analysis for project planning, dynamic re-planning and problem resolution during project execution

Connect “upstream” activity with “downstream” activity

Enable projects to achieve mid-course corrections to ensure project success


All Models (Qualitative and Quantitative)
Quantitative Models (Deterministic, Statistical, Probabilistic)
Statistical or Probabilistic Models
Interim outcomes predicted
Controllable x factors involved
Process Performance Model - With controllable x factors tied to Processes and/or Sub-processes
Only phases or lifecycles are modeled
Only uncontrollable factors are modeled
Only final outcomes are modeled
No uncertainty or variation modeled
Anecdotal Biased samples

When and Why Do We Need Process Performance Models?



The Proposal

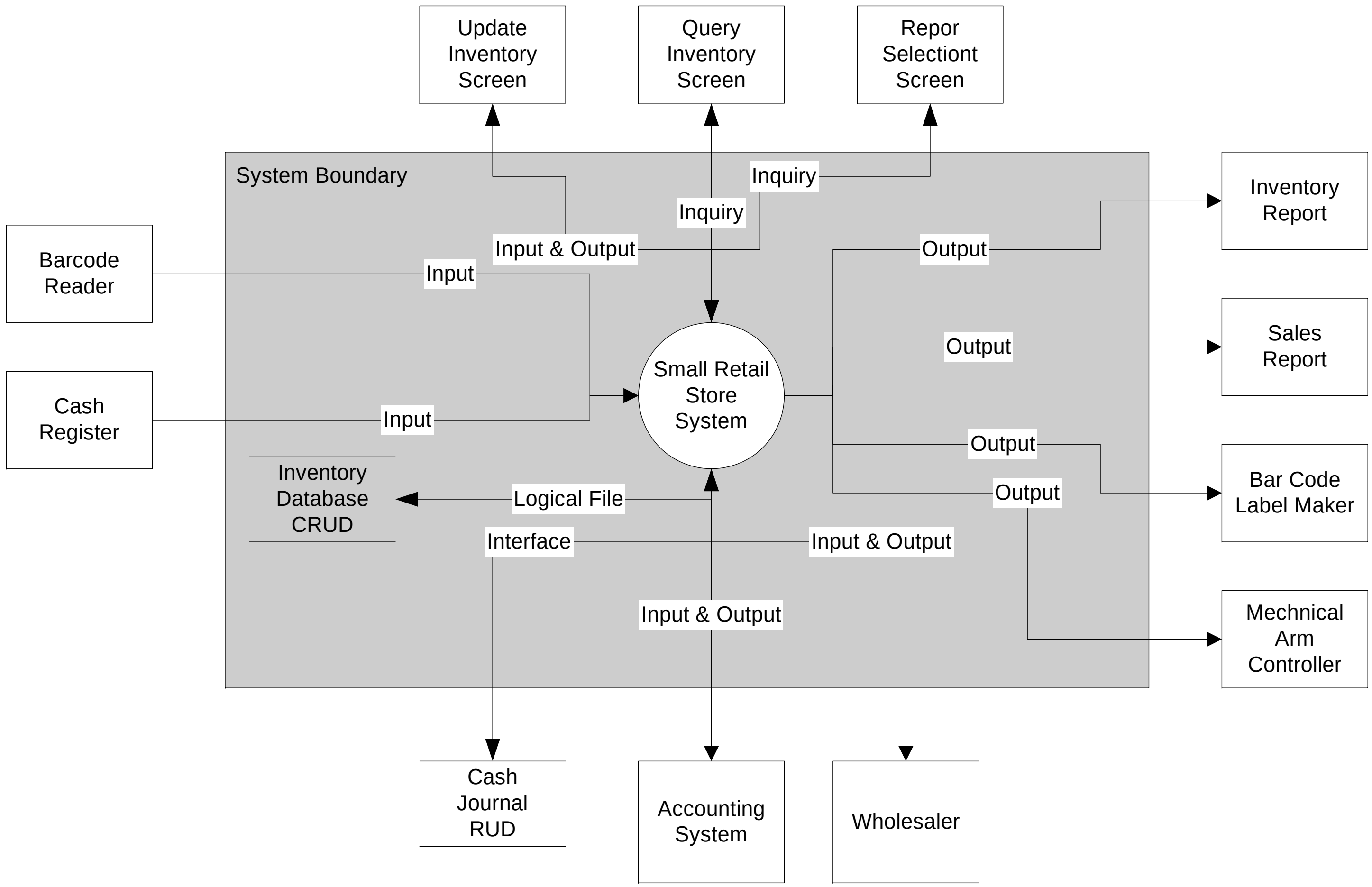
The Proposal

Often uses higher level PPMs

- False precision when using lower
- Better understanding of the risk in a bid

Results my be used for bid/no-bid decisions along with other criteria

Function Points



FP Type	#	Weight	Total
Inputs	5	4	20
Outputs	7	5	35
Inquiries	2	4	8
Logical Files	1	10	10
Interfaces	1	7	7
Total			80

Function Point Estimate

The function point estimate based on the context diagram results in

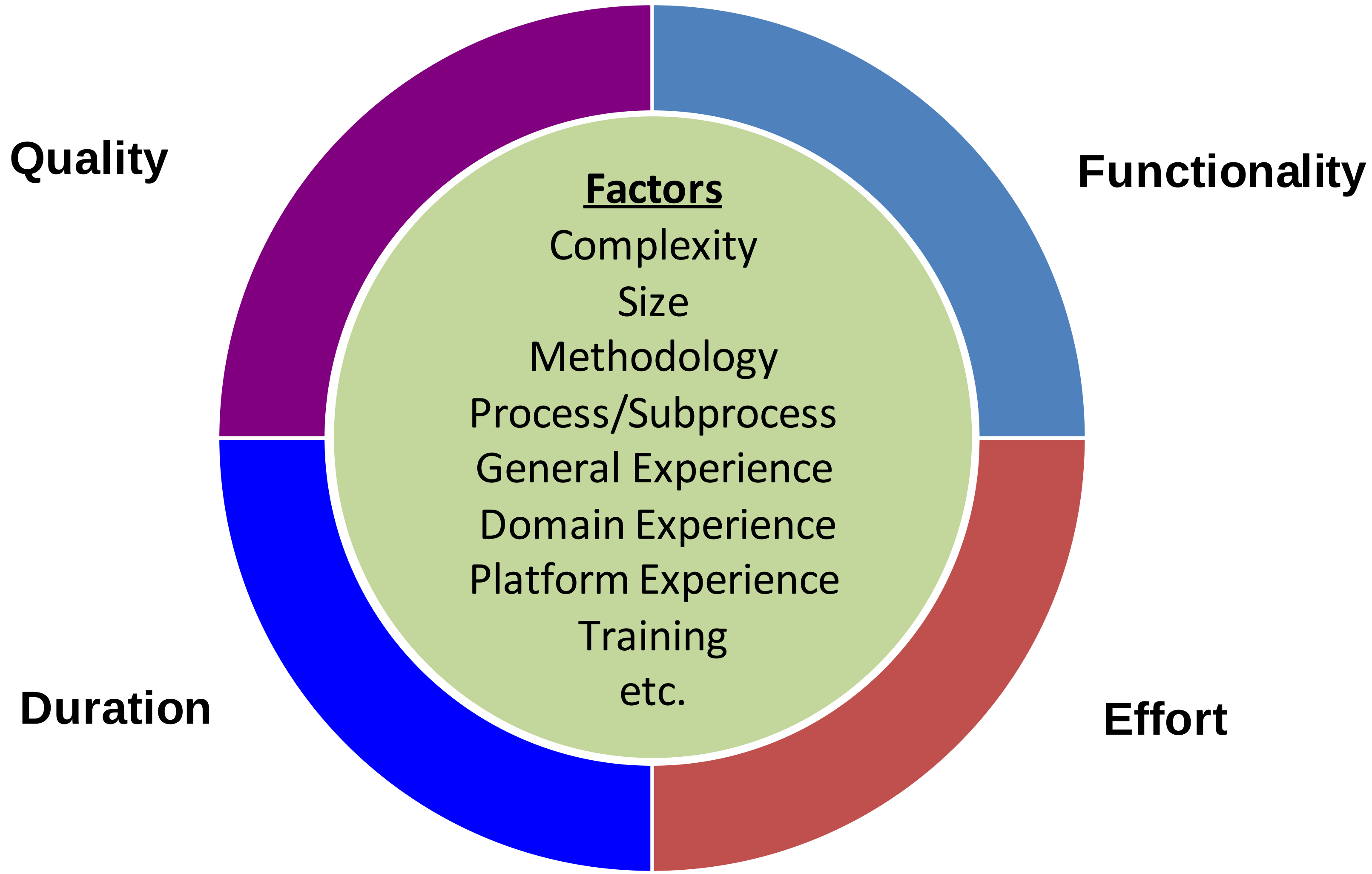
- 80 function points or 10,240 lines of code

However, two other context diagrams based on the proposal information resulted in estimates of

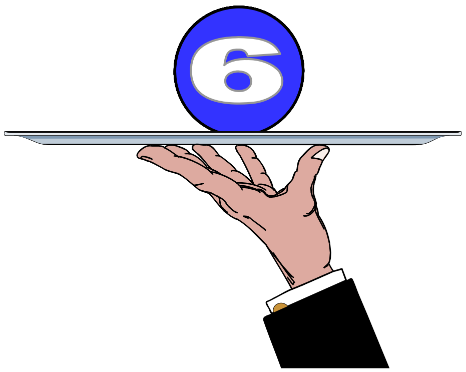
- 73 function points, or 9,344 lines of code and
- 96 function points or 12,288 lines of code

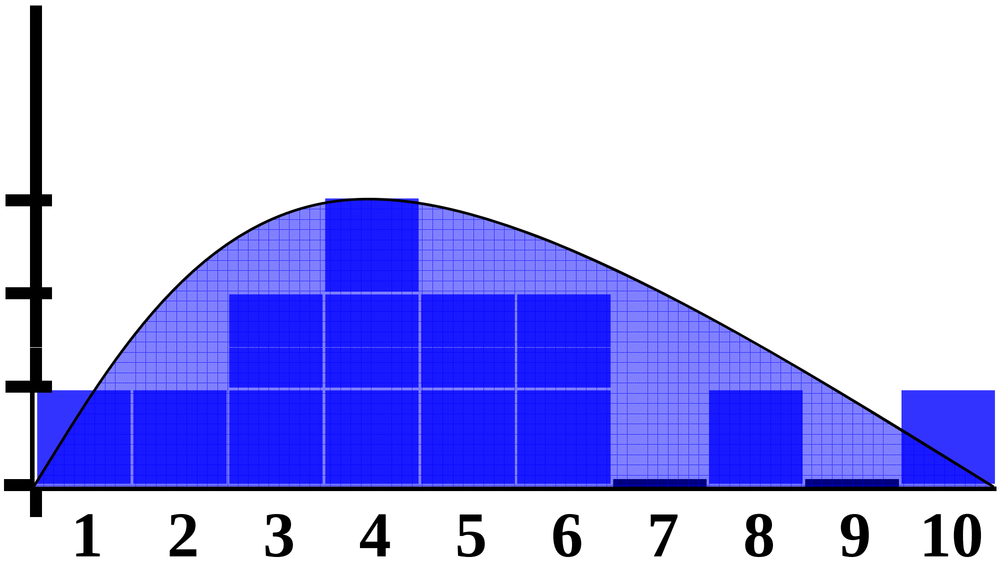
This variation in the estimates for the proposed system will be used for the process performance model (PPM) based predictions for the proposal and managing the project

Composition Trade-offs and PPM Factors



Understanding Distributions – Key to Informed Decisions



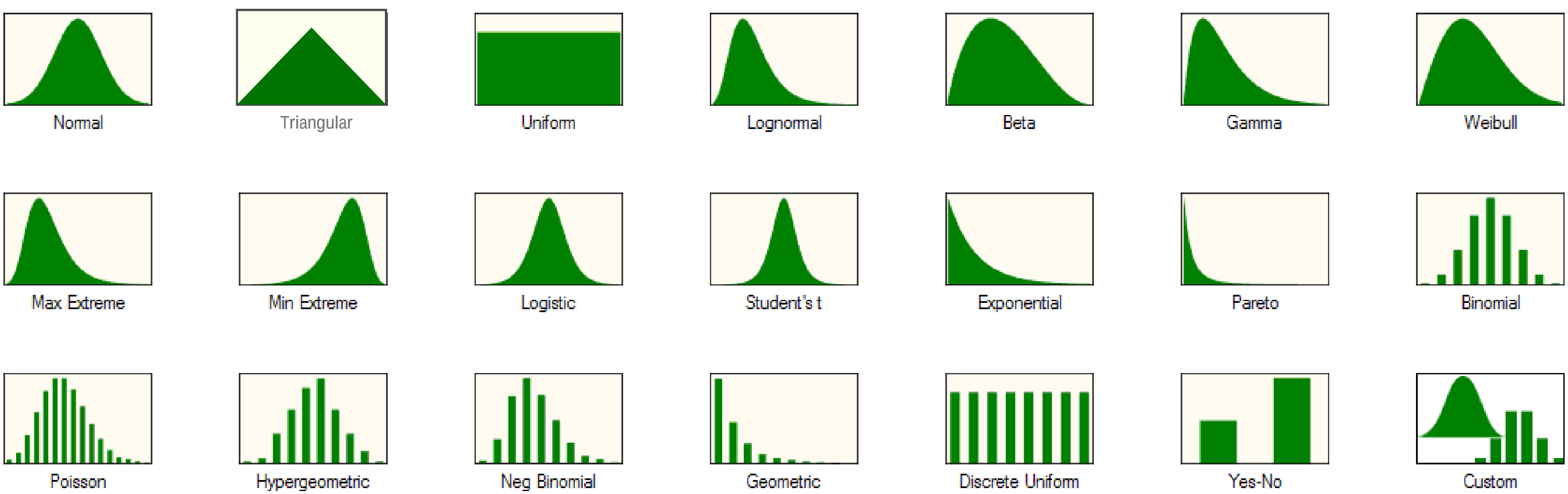


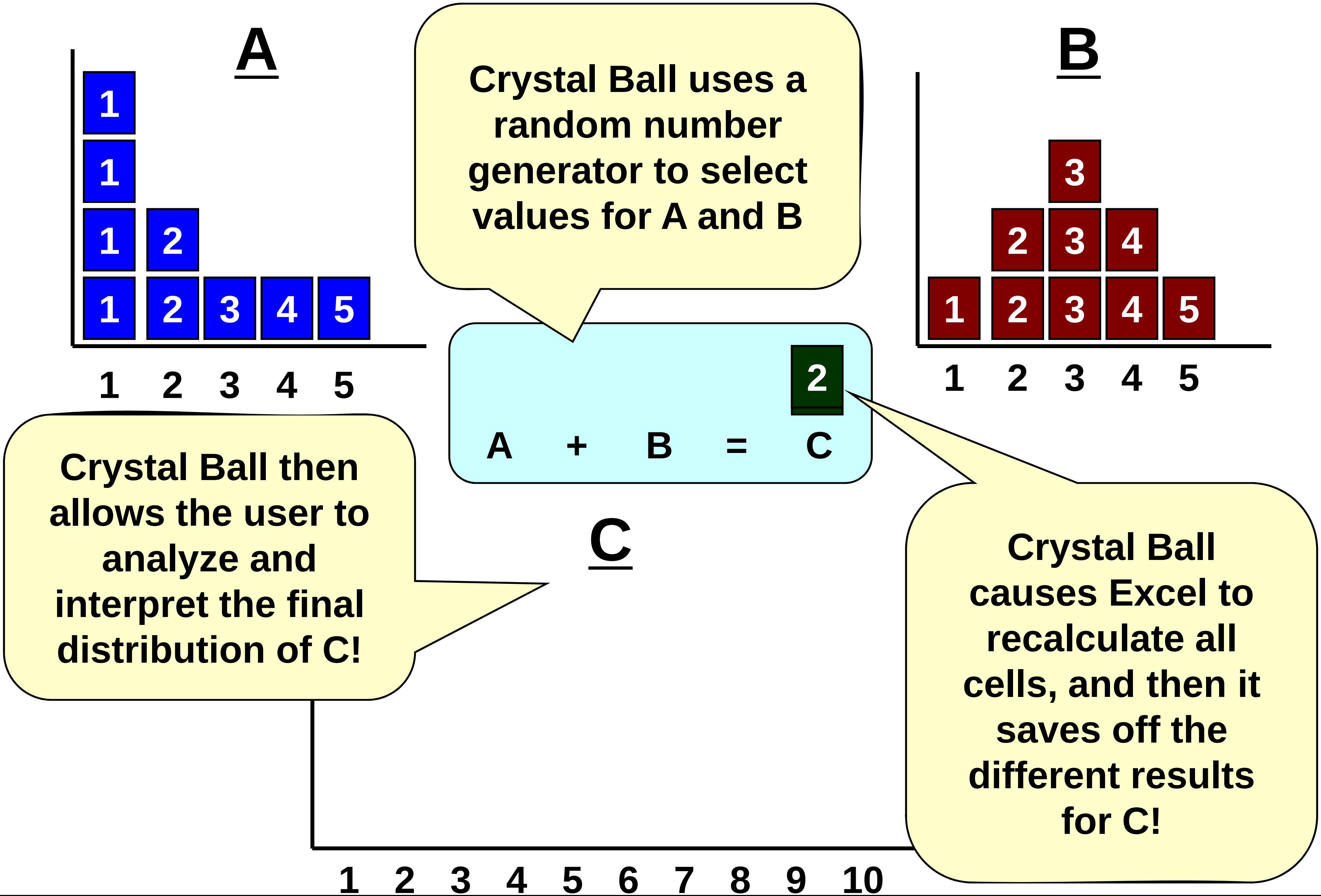
Distributions Describe Variation

Populations of data are characterized as distributions in most statistical procedures:

- expressed as an assumption for the procedure
- can be represented using an equation

The following are examples of distributions you may come across:

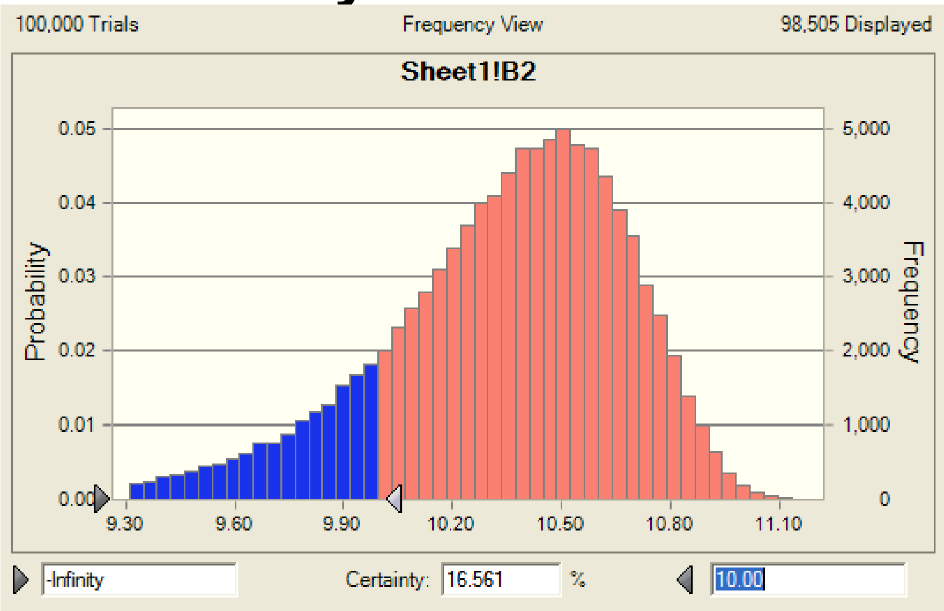



A
1
1
1 2
1 2 3 4 5
1 2 3 4 5
Crystal Ball uses a random number generator to select values for A and B
B
3
2 3 4
1 2 3 4 5
1 2 3 4 5
2
A + B = C
Crystal Ball then allows the user to analyze and interpret the final distribution of C!
C
Crystal Ball causes Excel to recalculate all cells, and then it saves off the different results for C!
1 2 3 4 5 6 7 8 9 10

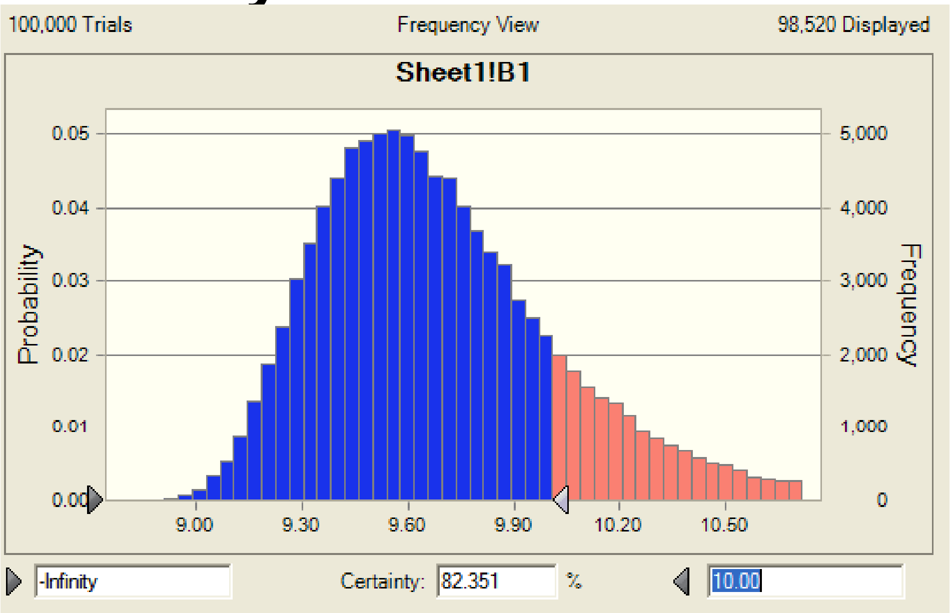
Why is Understanding Variation Important?

Customer wants the product in 10 weeks
Historical range is 9-11 weeks
Should the job be accepted?

Probably Not



Probably Should



Variation, Trade-offs, and PPMs

Function Pt Est. SLOC	0
Calculated KDSI	0

The shaded cells are where the effects of variation are incorporated using a Monte Carlo simulation

Nominal Effort =	0.0
Effort Multiplier =	0.0
Effort (MM) =	0.0
Nominal Schedule =	0.0
Staff =	#DIV/0!
Potential # Defects	0

Robert Stoddard
Rusty Young

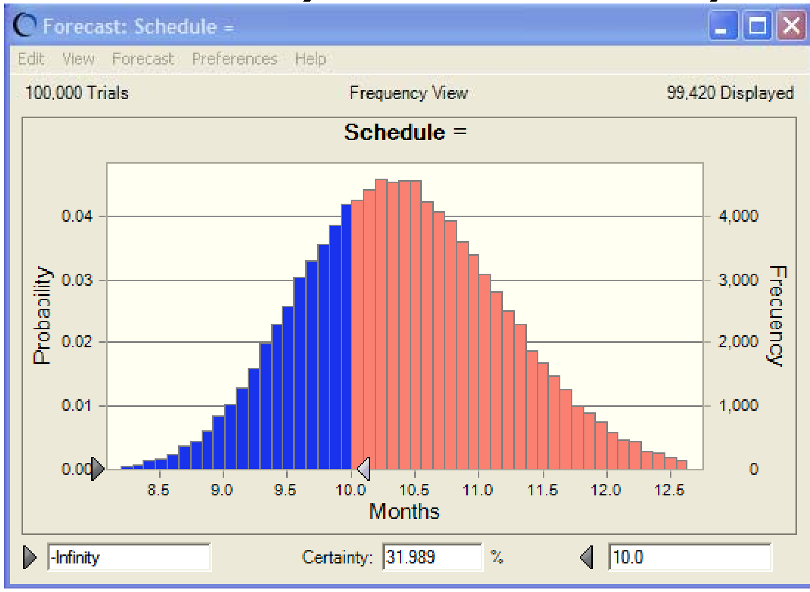
Evaluate Proposal Risk and Negotiate

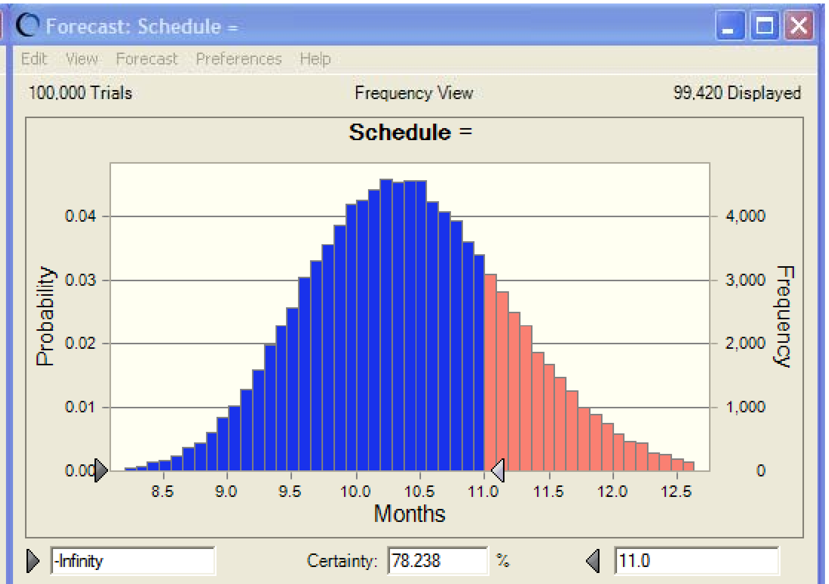
Run “what if” exercises holding one or more values constant

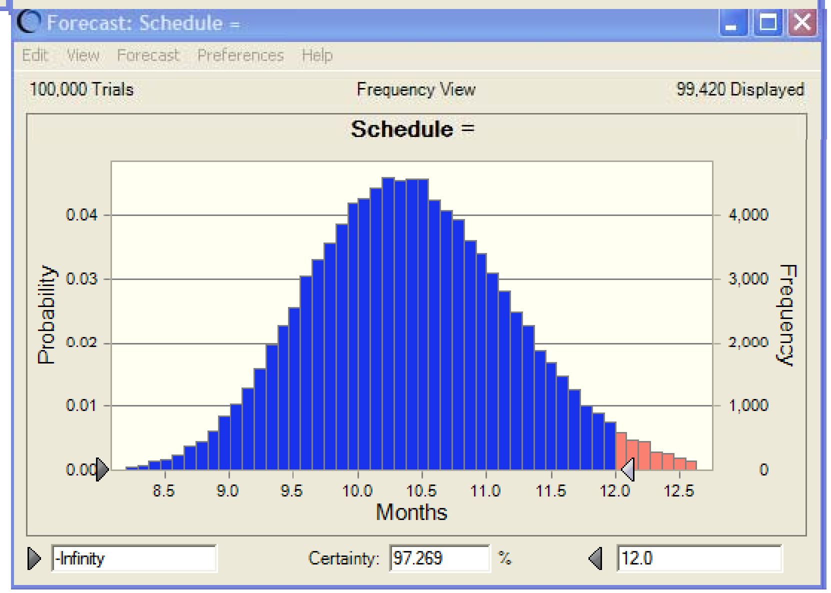
See effects of trade-offs between

- Schedule
- Effort
- Defects
- Staff
- Functionality

Variation, Trade-offs, and PPMs – Schedule



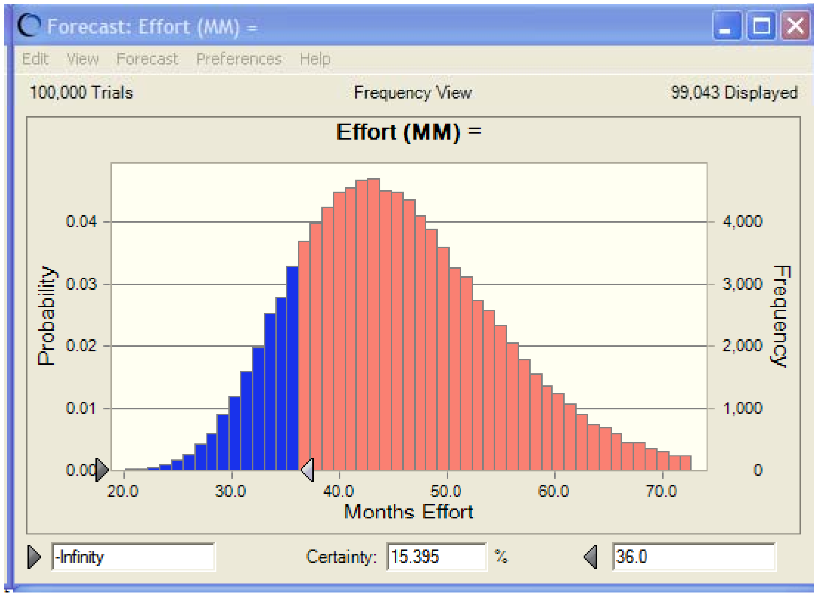


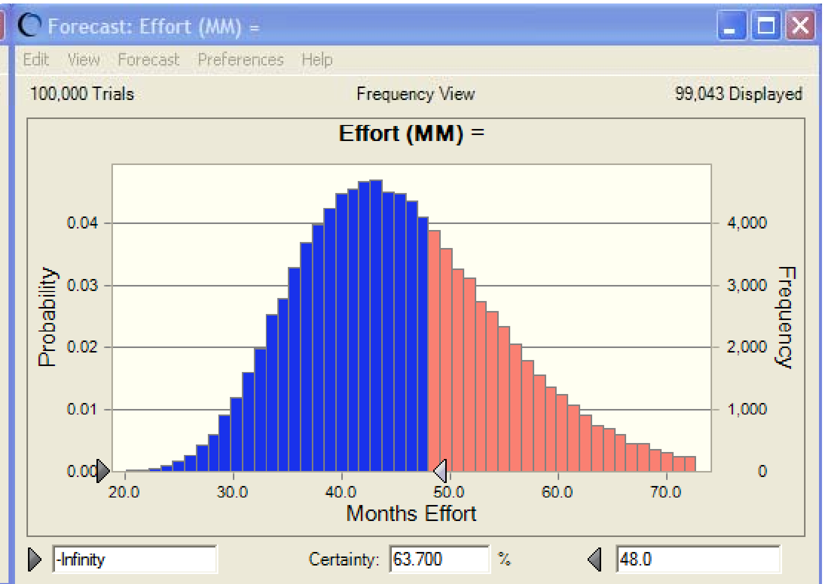


Forecast: Schedule =

Percentile	Forecast values
0%	7.6
10%	9.4
20%	9.7
30%	10.0
40%	10.2
50%	10.4
60%	10.6
70%	10.8
80%	11.1
90%	11.5
100%	14.0

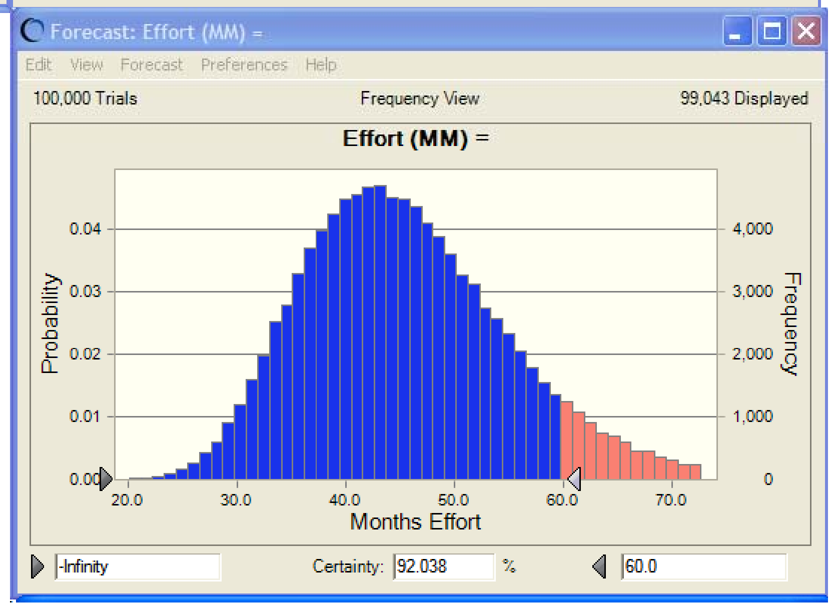
Variation, Trade-offs, and PPMs – Effort



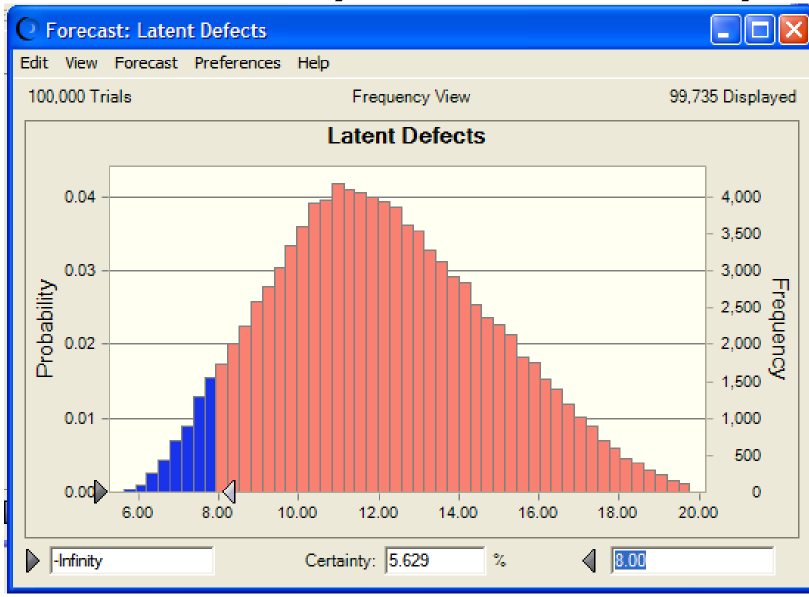


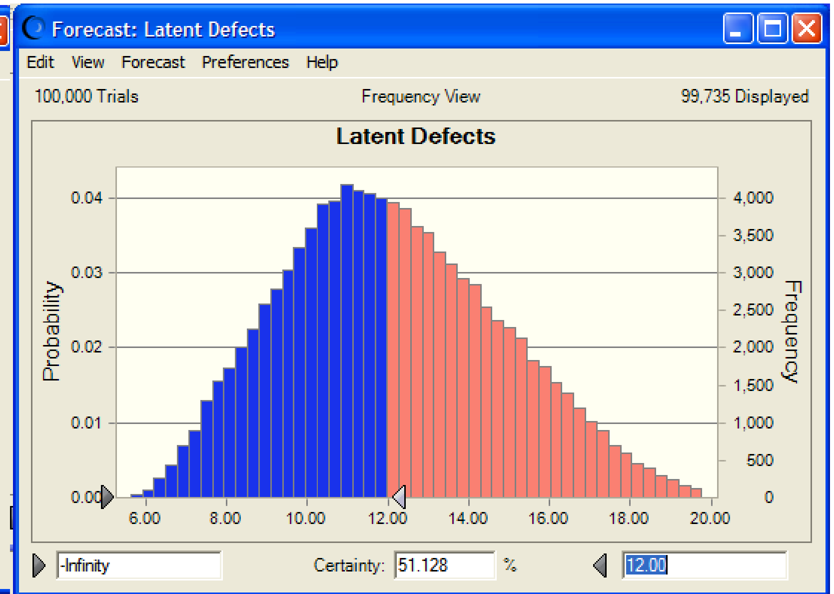
Forecast: Effort (MM) =

Percentile	Forecast values
0%	18.9
10%	34.1
20%	37.4
30%	40.0
40%	42.3
50%	44.6
60%	47.1
70%	49.8
80%	53.2
90%	58.4
100%	99.8



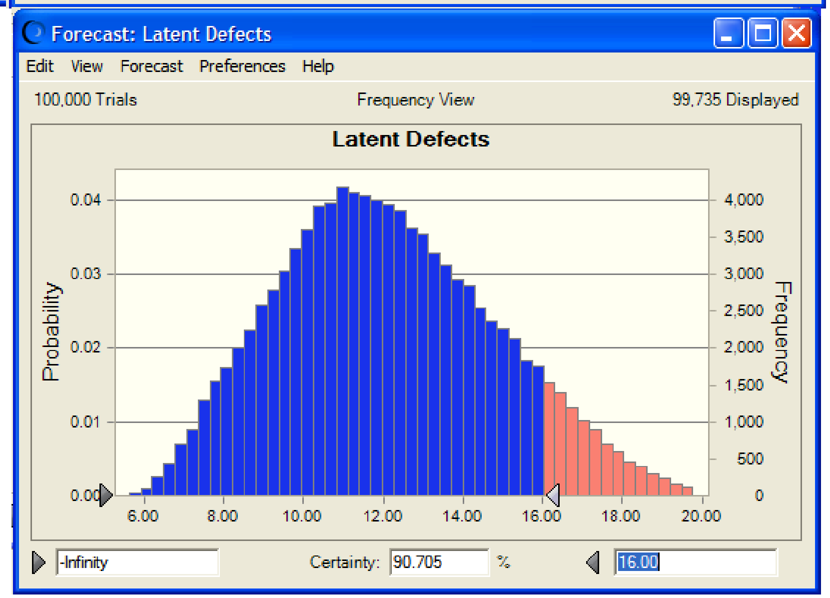
Variation, Trade-offs, and PPMs – Defects



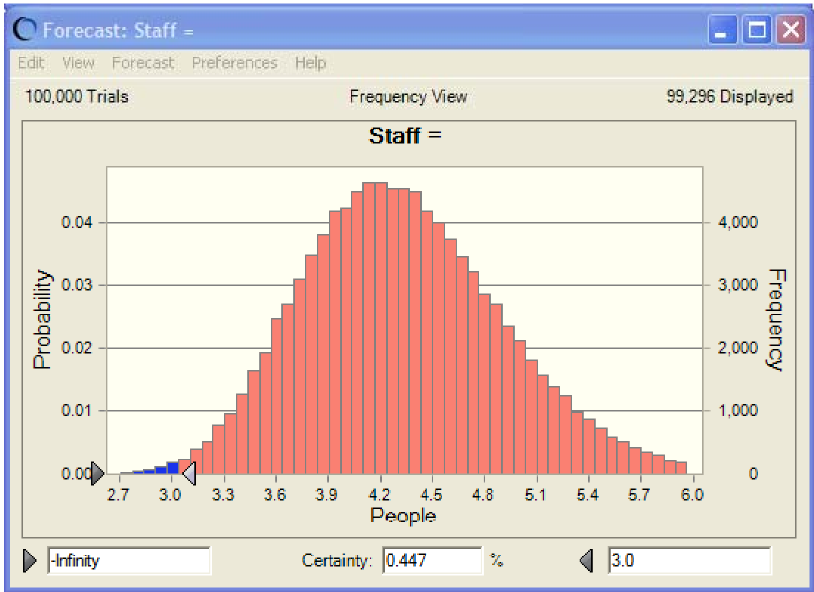


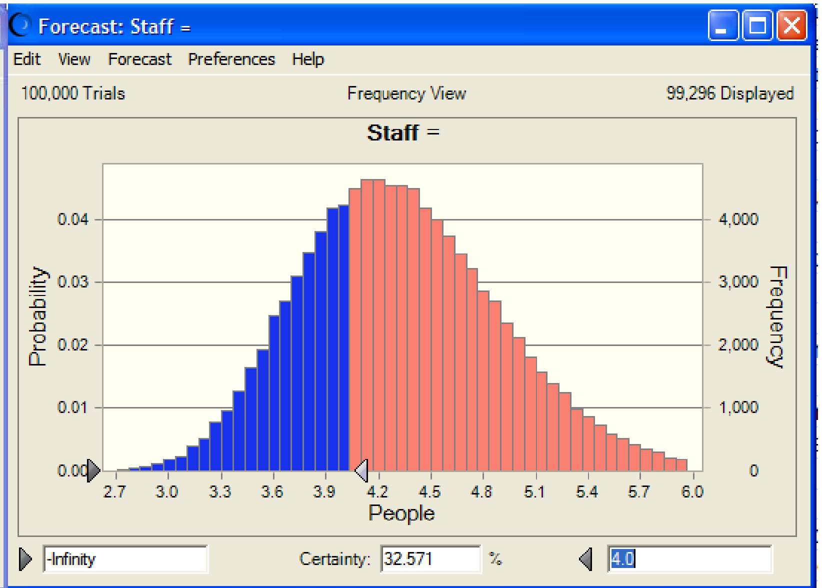
Forecast: Latent Defects

Percentile	Forecast values
0%	5.63
10%	8.64
20%	9.69
30%	10.50
40%	11.20
50%	11.92
60%	12.67
70%	13.51
80%	14.52
90%	15.88
100%	21.99



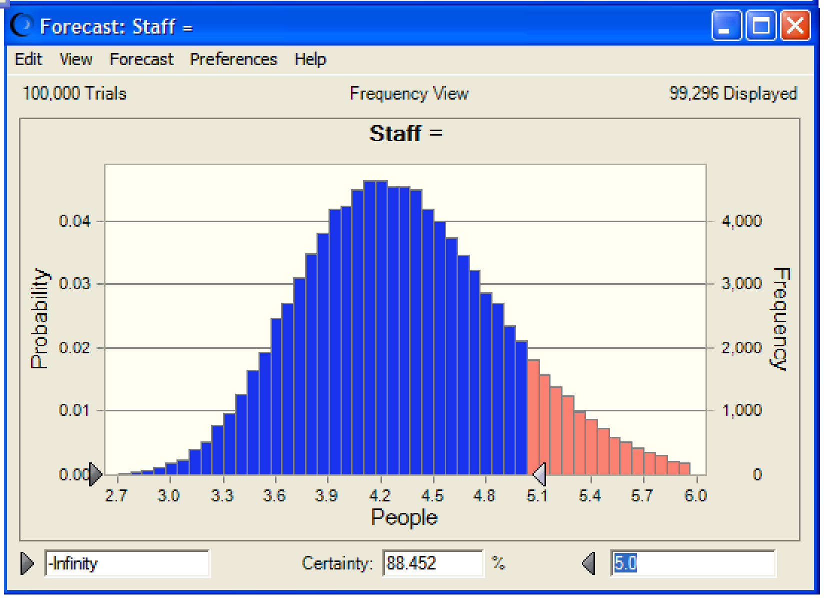
Variation, Trade-offs, and PPMs – Staff





Forecast: Staff =

Percentile	Forecast values
0%	2.5
10%	3.6
20%	3.8
30%	4.0
40%	4.2
50%	4.3
60%	4.4
70%	4.6
80%	4.8
90%	5.1
100%	7.2



Proposal CAR/OID to Mitigate Risk

Seeing if there are new technologies that if employed will reduce risk

May build/modify PPM to evaluate impact and ROI

- May involve a brief pilot
- May involve industry data
- May involve professional
- Each brings their own level of uncertainty to the prediction

Typically involves detailed project planning PPMs

- Results at micro-
- Extrapolate to macro

Proposal CAR/OID

New technology will increase coding productivity by 10%

- May want to verify with
  - pilot
  - in-depth testing
- Measured results

Adjust proposal model with results

Re-predict and evaluate resulting risks

Plan Project

Like proposal

- More detail
- Interim as well as end state

Compose a PDP and construct an initial PPM to ensure it will meet our goals and aid us managing the project

Initial PPM

Note: the greenish shaded cells on this and succeeding slides are where variations will be accounted for using a Monte Carlo simulation

Phase	UoM	Size	Effort
Proposal/Early Planning	Function Points	80	154
Elicit Requirements	User Requirements	110	723
URD Review	Defects		65
Analyze Requirements	Requirements	176	1809
SRS Review	Defects		98
Design	Components	124	2236
Design Review	Defects		72
Code	Components	110	2950
Code Review	Defects		65
Test	Test Cases	229	2633
Deliver	Defects		10806

Initial PPM

Size and Effort are predicted functions:

$SRS_{size} = \int(URD_{size}$, method, review type. Etc.)

$Effort = \int(Document_{size}$, *method, experience, etc.*)

Phase	UoM	Size	Effort
Proposal/Early Planning	Function Points	80	154
Elicit Requirements	User Requirements	110	723
URD Review	Defects		65
Analyze Requirements	Requirements	176	1809
SRS Review	Defects		98
Design	Components	124	2236
Design Review	Defects		72
Code	Components	110	2950
Code Review	Defects		65
Test	Test Cases	229	2633
Deliver	Defects		10806

Initial PPM

Size and Effort are predicted functions:

$SRS_{size} = \int(URD_{size}$, method, review type. etc.)

$Effort = \int(Document_{size}$, *method, experience, etc.*)

Phase	UoM	Size	Effort
Proposal/Early Planning	Function Points		154
Code Review	Defects		65
Test	Test Cases	229	2633
Deliver	Defects		10806

Effort = constant + multiplier * size$^{(method+experience+training)}$

Experience = ∫(domain, customer, platform general)

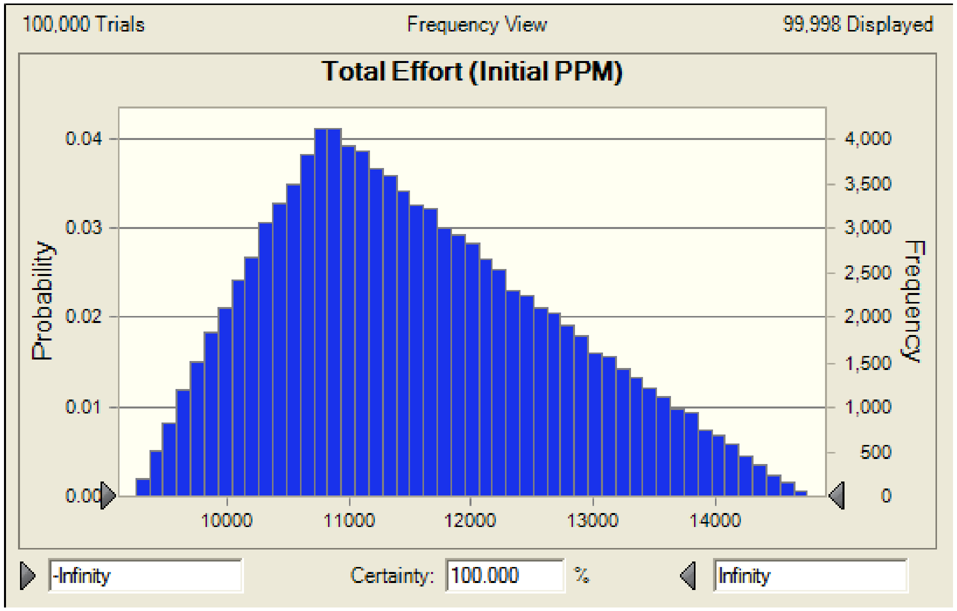
Initial PPM

Predicted Defects				
URD Defects	SRS Defects	DES Defects	Code Defects	Latent Defects
223				223
89				89
85	296			381
34	118			152
32	112	158		302
13	45	63		121
12	43	60	137	252
5	17	24	55	101
1	5	7	16	30

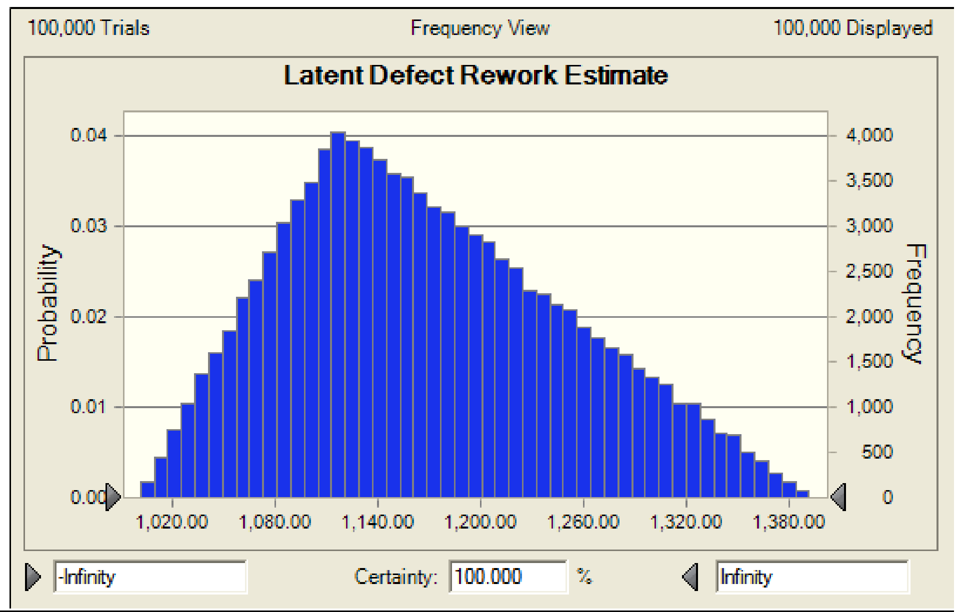
Initial PPM

Rework Effort				
URD	SRS	DES	Code	Latent
134				134
4				4
153	178			330
5	6			11
174	202	95		471
6	7	3		16
198	231	108	82	620
278	323	151	115	867
357	415	194	148	1115

PDP Risk

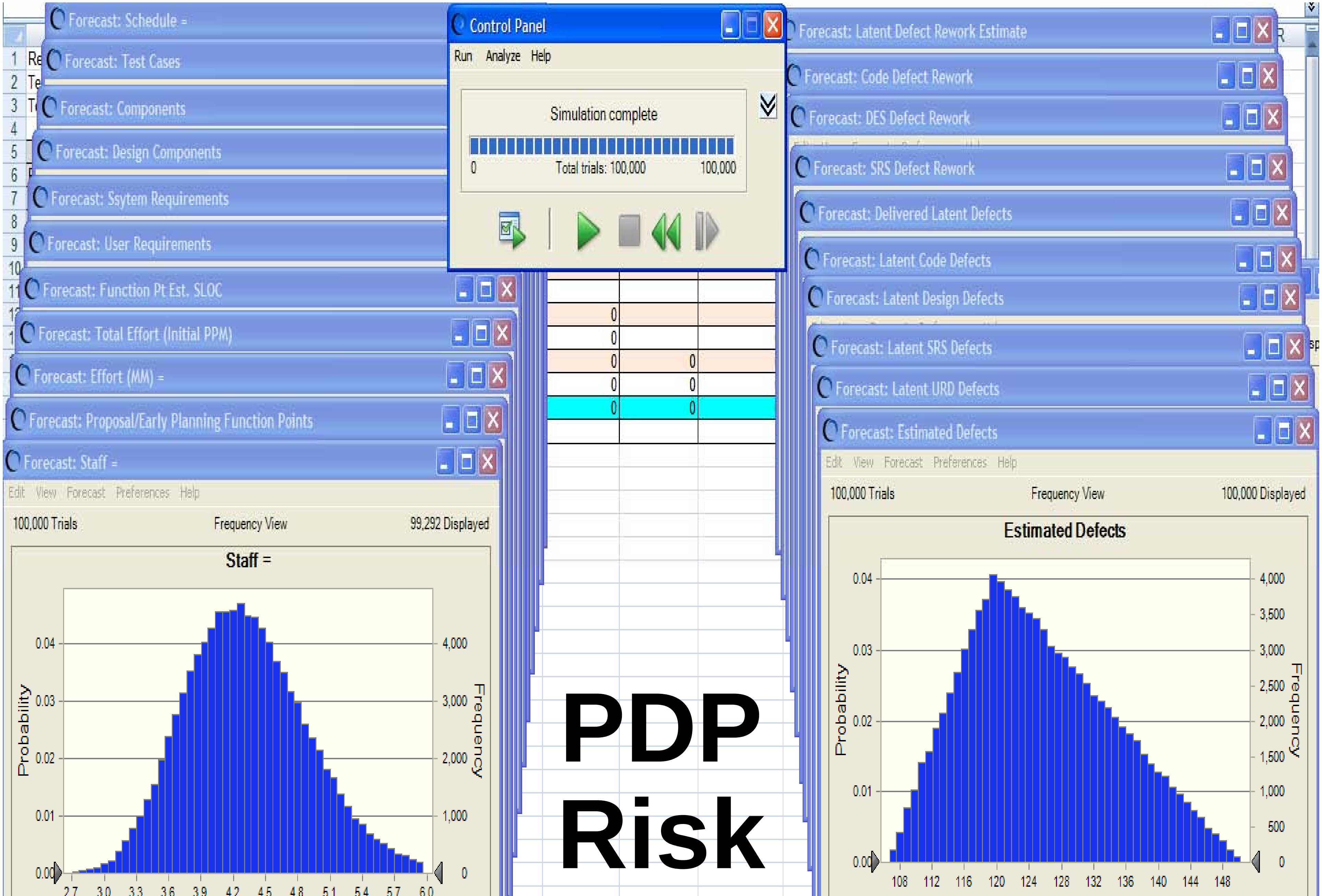


100,000 Trials
Frequency View
99,998 Displayed
Total Effort (Initial PPM)
Probability
Frequency
0.04
0.03
0.02
0.01
0.00
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
10000
11000
12000
13000
14000
-Infinity
Certainty: 100.000 %
Infinity



100,000 Trials
Frequency View
100,000 Displayed
Latent Defect Rework Estimate
Probability
Frequency
0.04
0.03
0.02
0.01
0.00
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
1,020.00
1,080.00
1,140.00
1,200.00
1,260.00
1,320.00
1,380.00
-Infinity
Certainty: 100.000 %
Infinity

PDP Risk



Forecast: Schedule =
Forecast: Test Cases
Forecast: Components
Forecast: Design Components
Forecast: Sytem Requirements
Forecast: User Requirements
Forecast: Function Pt Est. SLOC
Forecast: Total Effort (Initial PPM)
Forecast: Effort (MM) =
Forecast: Proposal/Early Planning Function Points
Forecast: Staff =
Edit View Forecast Preferences Help
100,000 Trials
Frequency View
99,292 Displayed
Staff =
Probability
Frequency
Control Panel
Run Analyze Help
Simulation complete
0
Total trials: 100,000
100,000
Forecast: Latent Defect Rework Estimate
Forecast: Code Defect Rework
Forecast: DES Defect Rework
Forecast: SRS Defect Rework
Forecast: Delivered Latent Defects
Forecast: Latent Code Defects
Forecast: Latent Design Defects
Forecast: Latent SRS Defects
Forecast: Latent URD Defects
Forecast: Estimated Defects
Edit View Forecast Preferences Help
100,000 Trials
Frequency View
100,000 Displayed
Estimated Defects
Probability
Frequency

An Alternate Example Process Performance Model (PPM) to support Escaped Defect Analysis and Monitoring

The Situation during Development

Defects escaping from one development phase to the next are very expensive to find, diagnose and fix. Some industrial studies suggest the increasing cost may be exponential.

OUR NEED: A PPM used by the software project manager and quality team to analyze escaping defect rates by type to support more informed decisions on where to target dynamic project corrective action, as well as, changes to organizational processes!

Details of the Escaping Defect PPM

The outcome, Y, is the amount of escaped defects by type within each phase of development

The x factors used in this model will be the various injection and detection rates by type of defect across the phases of development

Not only will this model focus on phase containment of Req'ts, Design, and Code phases, but on the phase screening of defects by type within the different types of testing

Background Information on the Data

Historical data on escaped defects, by type, across lifecycle phases was recorded.

For each historical project, software size was recorded, as well, to help normalize the defects injected and found, thereby producing injection and detection rates.

All numbers per 1MSLOC	Reqts Activity	Design Activity	Code Activity	Integration Test	System Test	User Test	Total Injection Rate by	Escape Rate by Activity	Escape Rate by All Activities	Activity Injection Rate by	Escape Rate by Activity %	Escape Rate by All Activities %
	Phase Found											
Phase Injected												
Reqts Activity	500	80	10	60	150	30	830	330	330	7%	40%	40%
Design Activity		2000	300	1200	600	100	4200	2200	2450	36%	52%	54%
Code Activity			3800	2000	400	140	6340	2540	4680	54%	40%	53%
Integration				300	50	5	355	55	1475	3%	15%	29%
System Test					100	0	100	0	275	1%	0%	17%
User Test						5	5	0	0	0%	0%	0%
Phase Containment Rate	500	2080	4110	3560	1300	280	11830	5125	9210	100%		
Screening Rate	500	2000	3800	300	100	5						
Phase Containment Rate %	60%	46%	47%	71%	83%	100%						
Screening Rate %	60%	48%	60%	85%	100%	100%						

43% Of all defects escaped at least one phase

1.8 For defects not caught in phase originally injected, this is the average number of times they escaped a phase

A modern spreadsheet for escaped defect analysis before being transformed into a CMMI Process Performance Model

All numbers per 1MSLOC	Phase Found: Reqts Activity	Design Activity	Code Activity	Integration Test	System Test	User Test	Total Injection Rate by	Escape Rate by Activity	Escape Rate by All Activities	Activity Injection Rate by	Escape Rate by Activity %	Escape Rate by All Activities %
Phase Injected: Reqts Activity	500	80	10	60	150	30	830	330	330	7%	40%	40%
Design Activity		2000	300	1200	600	100	4200	2200	2450	36%	52%	54%
Code Activity			3800	2000	400	140	6340	2540	4680	54%	40%	53%
Integration				300	50	5	355	55	1475	3%	15%	29%
System Test					100	0	100	0	275	1%	0%	17%
User Test						5	5	0	0	0%	0%	0%
Phase Containment Rate	500	2080	4110	3560	1300	280	11830	5125	9210	100%		
Screening Rate	500	2000	3800	300	100	5						
Phase Containment Rate %	60%	46%	47%	71%	83%	100%						
Screening Rate %	60%	48%	60%	85%	100%	100%						

43% Of all defects escaped at least one phase

1.8 For defects not caught in phase originally injected, this is the average number of times they escaped a phase

Let's look at the matrix showing "Phase Injected" vs "Phase Found"

Phase Injected \ Phase Found (All numbers per 1MSLOC)	Reqts Activity	Design Activity	Code Activity	Integration Test	System Test	User Test
Reqts Activity	500	80	10	60	150	30
Design Activity		2000	300	1200	600	100
Code Activity			3800	2000	400	140
Integration Test				300	50	5
System Test					100	0
User Test						5

For example, an average of 2000 design defects were found during the design activity

All numbers per 1MSLOC	Phase Found											
	Reqts Activity	Design Activity	Code Activity	Integration Test	System Test	User Test	Total Injection Rate by	Escape Rate by Activity	Escape Rate by All Activities	Activity Injection Rate by	Escape Rate by Activity %	Escape Rate by All Activities %
Phase Injected												
Reqts Activity	500	80	10	60	150	30	830	330	330	7%	40%	40%
Design Activity		2000	300	1200	600	100	4200	2200	2450	36%	52%	54%
Code Activity			3800	2000	400	140	6340	2540	4680	54%	40%	53%
Integration				300	50	5	355	55	1475	3%	15%	29%
System Test					100	0	100	0	275	1%	0%	17%
User Test						5	5	0	0	0%	0%	0%
Phase Containment Rate	500	2090	4110	3560	1300	280	11830	5125	9210	100%	43%	
Screening Rate	500	2000	3800	300	100	5						
Phase Containment Rate %	60%	46%	47%	71%	83%	100%						
Screening Rate %	60%	48%	60%	85%	100%	100%						

43% Of all defects escaped at least one phase

1.8 For defects not caught in phase originally injected, this is the average number of times they escaped a phase

Let's look at the "Phase Containment" & "Phase Screening" rates

Robert Stoddard
Rusty Young

	Phase Found					
	Reqts Activity	Design Activity	Code Activity	Integration Test	System Test	User Test
Phase Containment Rate	500	2080				280
Phase Screening Rate	500	2000				5
Phase Containment Rate %	60%	46%				
Phase Screening Rate %	60%	48%				

Here, 2080 Requirements and Design defects were caught during Design

Here, 2000 Design defects were caught during Design

% of all defects entering and injected in Design caught in Design

% of Design defects caught in Design

All numbers per 1MSLOC	Reqts Activity	Design Activity	Code Activity	Integration Test	System Test	User Test	Total Injection Rate by	Escape Rate by Activity	Escape Rate by All Activities	Activity Injection Rate by	Escape Rate by Activity %	Escape Rate by All Activities %
	Phase Found											
Phase Injected												
Reqts Activity	500	80	10	60	150	30	830	330	330	7%	40%	40%
Design Activity		2000	300	1200	600	100	4200	2200	2450	36%	52%	54%
Code Activity			3800	2000	400	140	6340	2540	4680	54%	40%	53%
Integration				300	50	5	355	55	1475	3%	15%	29%
System Test					100	0	100	0	275	1%	0%	17%
User Test						5	5	0	0	0%	0%	0%
Phase Containment Rate	500	2080	4110	3560	1300	280	11830	5125	9210	100%		
Screening Rate	500	2000	3800	300	100	5						
Phase Containment Rate %	60%	46%	47%	71%	83%	100%						
Screening Rate %	60%	48%	60%	85%	100%	100%						

43% Of all defects escaped at least one phase

1.8 For defects not caught in phase originally injected, this is the average number of times they escaped a phase

Let's look at the Phase Injection and Escape rates

Here, 4200 Design defects were injected with 2200 of them escaping the Design activity; Additionally, 2450 total defects (injected during Design or inherited from upstream activities) escaped past the Design activity

Phase Injected	All numbers per 1MSLOC	Total Injection Rate by Activity	Escape Rate by Activity	Escape Rate by All Activities	Activity Injection Rate by Total Defects %	Escape Rate by Activity %	Escape Rate by All Activities %
	Reqts Activity	830	330	330	7%	40%	40%
	Design Activity	4200	2200	2450	36%	52%	54%
	Code Activity	6340	2540	4680	54%	40%	53%
	Integration Test	355	55	1475	3%	15%	29%
	System Test	100	0	275	1%	0%	17%
	User Test	5	0	0	0%	0%	0%

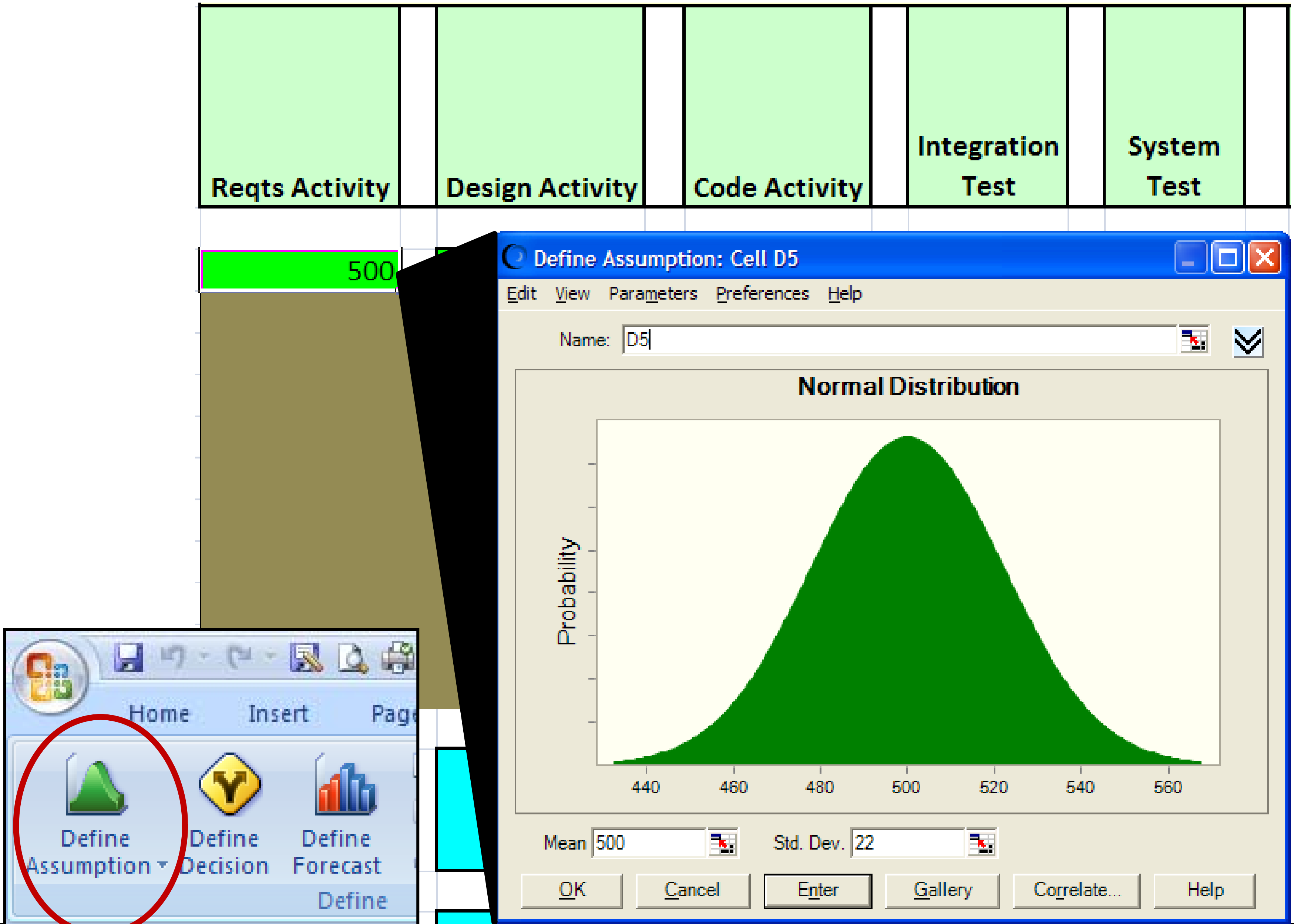
Here, 36% of all defects in a project are expected to be Design defects; 52% of Design defects are expected to escape past Design; and 54% of all types of defects in the Design activity (injected during Design or inherited from upstream activities) are escaping past the Design activity

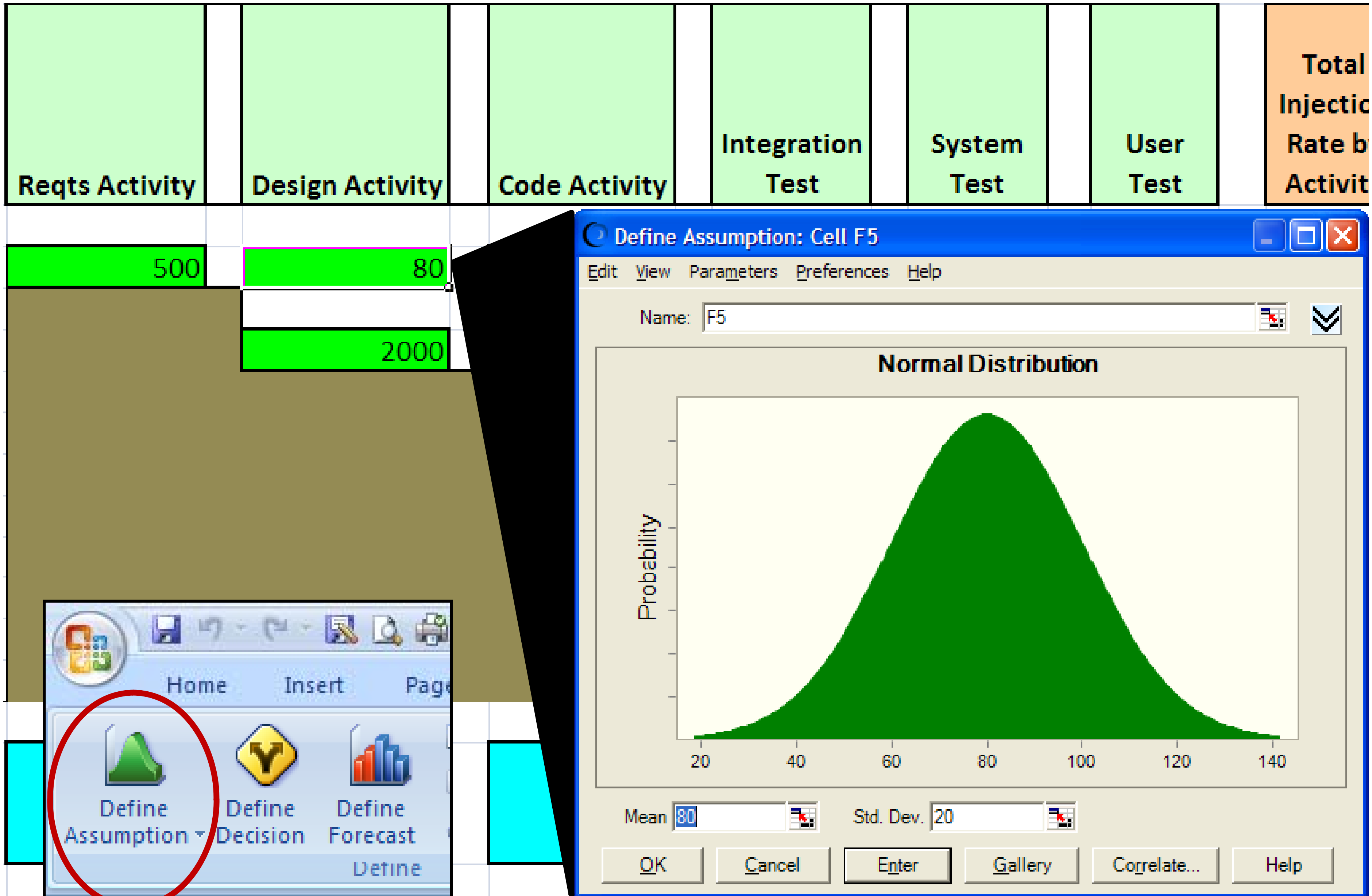
Phase Injected (All numbers per 1MSLOC)	Total Injection Rate by Activity	Escape Rate by Activity	Escape Rate by All Activities	Activity Injection Rate by Total Defects %	Escape Rate by Activity %	Escape Rate by All Activities %
Reqts Activity	830	330	330	7%	40%	40%
Design Activity	4200	2200	2450	36%	52%	54%
Code Activity	6340	2540	4680	54%	40%	53%
Integration Test	355	55	1475	3%	15%	29%
System Test	100	0	275	1%	0%	17%
User Test	5	0	0	0%	0%	0%

All numbers per 1MSLOC	Phase Found: Reqts Activity	Design Activity	Code Activity	Integration Test	System Test	User Test	Total Injection Rate by	Escape Rate by Activity	Escape Rate by All Activities	Activity Injection Rate by	Escape Rate by Activity %	Escape Rate by All Activities %
Phase Injected: Reqts Activity	500	80	10	60	150	30	830	330	330	7%	40%	40%
Design Activity		2000	300	1200	600	100	4200	2200	2450	36%	52%	54%
Code Activity			3800	2000	400	140	6340	2540	4680	54%	40%	53%
Integration				300	50	5	355	55	1475	3%	15%	29%
System Test					100	0	100	0	275	1%	0%	17%
User Test						5	5	0	0	0%	0%	0%
Phase Containment Rate	500	2080	4110	3560	1300	280	11830	5125	9210	100%	43%	Of all defects escaped at least one phase
Screening Rate	500	2000	3800	300	100	5						
Phase Containment Rate %	60%	46%	47%	71%	83%	100%						
Screening Rate %	60%	48%	60%	85%	100%	100%						

1.8 For defects not caught in phase originally injected, this is the average number of times they escaped a phase

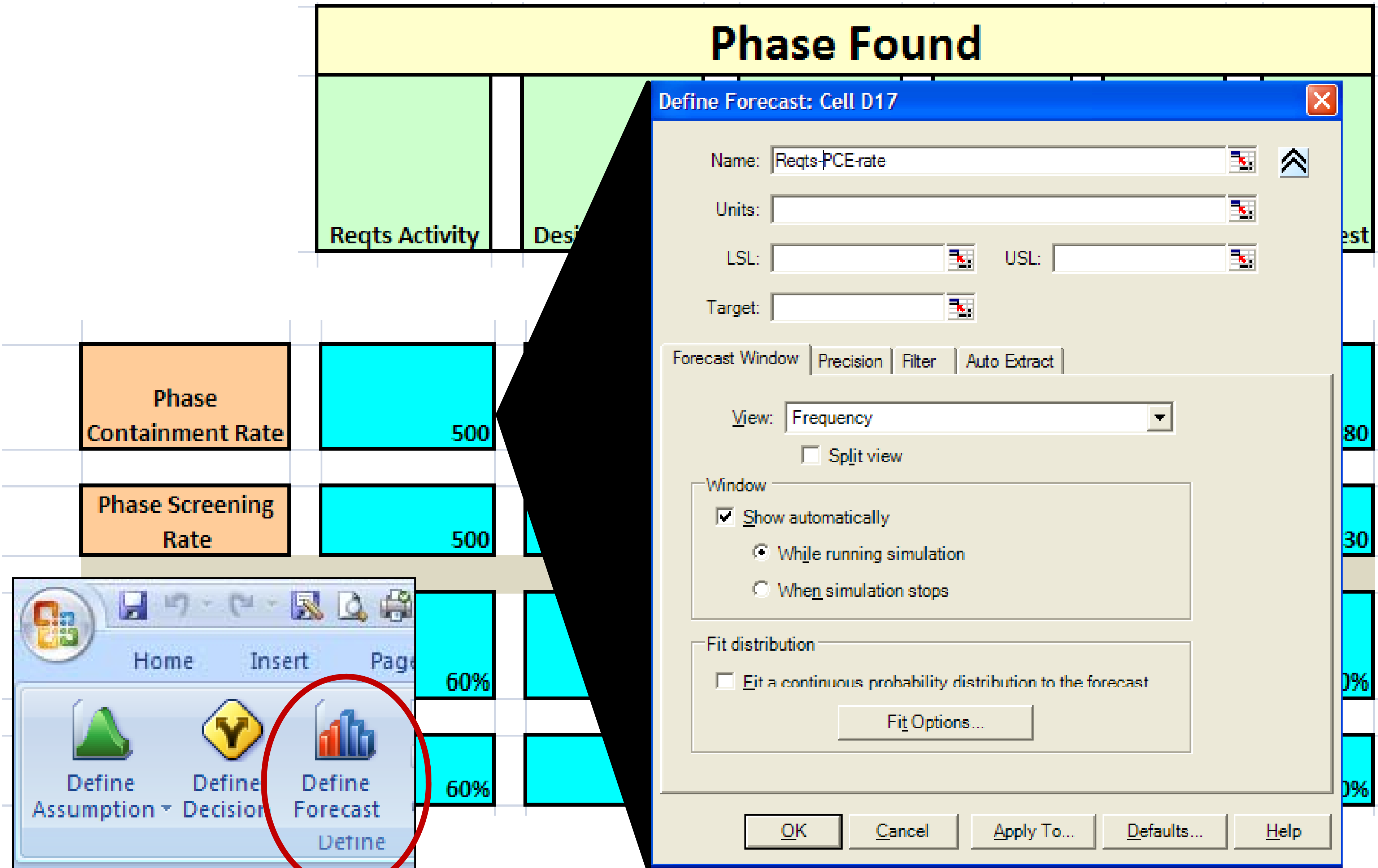
Now, let's transform this spreadsheet model into a valid CMMI process performance model!

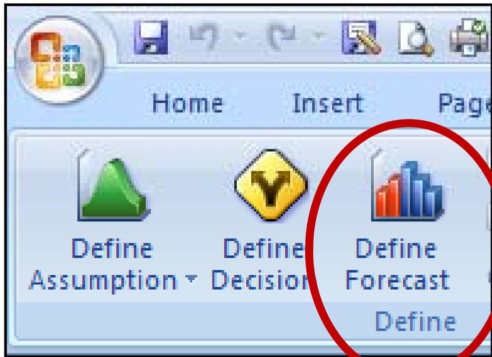

Reqts Activity
Design Activity
Code Activity
Integration Test
System Test
500
Define Assumption: Cell D5
Edit View Parameters Preferences Help
Name: D5
Normal Distribution
Probability
440 460 480 500 520 540 560
Mean 500
Std. Dev. 22
OK Cancel Enter Gallery Correlate... Help
Home Insert Page
Define Assumption
Define Decision
Define Forecast
Define


Reqts Activity
Design Activity
Code Activity
Integration Test
System Test
User Test
Total Injectic Rate b Activit
500
80
2000
Define Assumption: Cell F5
Edit View Parameters Preferences Help
Name: F5
Normal Distribution
Probability
20 40 60 80 100 120 140
Mean 80
Std. Dev. 20
OK Cancel Enter Gallery Correlate... Help
Home Insert Pag
Define Assumption
Define Decision
Define Forecast
Define

All numbers per 1MSLOC	Phase Found					
Phase Injected	Reqts Activity	Design Activity	Code Activity	Integration Test	System Test	User Test
Reqts Activity	500	80	10	60	150	30
Design Activity		2000	300	1200	600	100
Code Activity			3800	2000	400	140
Integration Test				300	50	5
System Test					100	0
User Test						5

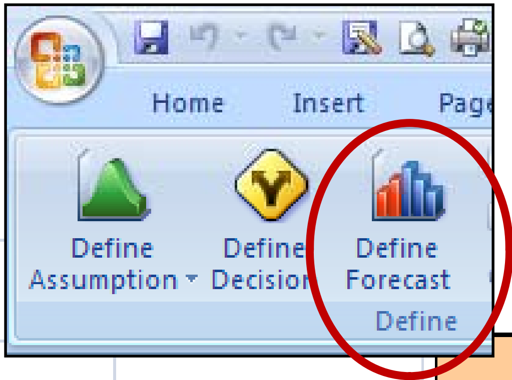
Each of the green cells have received uncertainty distributions based on historical data


Phase Found
Reqts Activity
Phase Containment Rate
500
Phase Screening Rate
500
60%
60%
Define Forecast: Cell D17
Name: Reqts-PCE-rate
Units:
LSL:
USL:
Target:
Forecast Window
Precision
Filter
Auto Extract
View: Frequency
Split view
Window
Show automatically
While running simulation
When simulation stops
Fit distribution
Fit a continuous probability distribution to the forecast
Fit Options...
OK
Cancel
Apply To...
Defaults...
Help
Home
Insert
Define Assumption
Define Decision
Define Forecast
Define



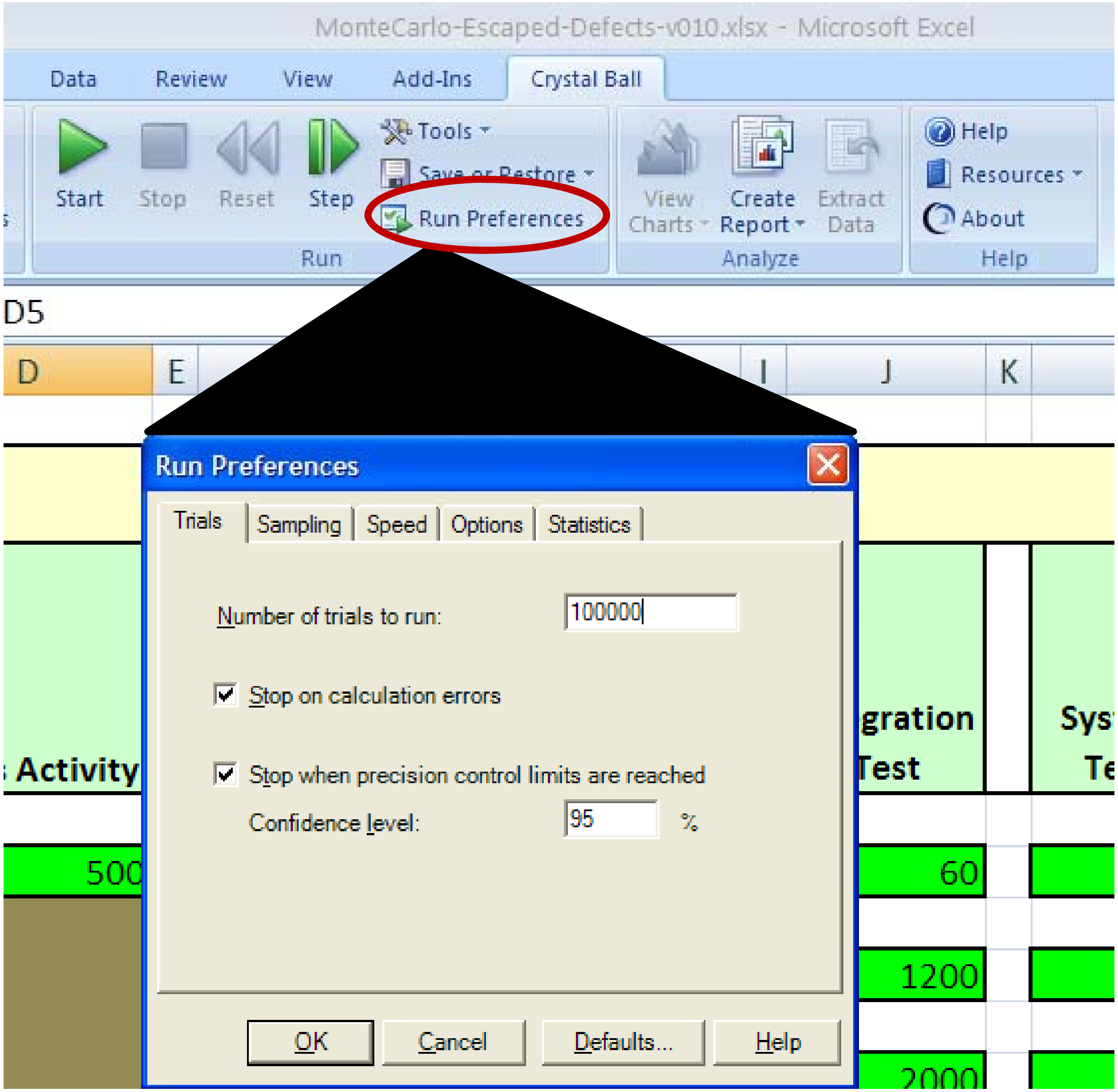
Phase Found	Reqts Activity	Design Activity	Code Activity	Integration Test	System Test	User Test
Phase Containment Rate	500	2080				80
Phase Screening Rate	500	2000				5
Phase Containment Rate %	60%	46%	47%	71%	83%	100%
Phase Screening Rate %	60%	48%	60%	85%	100%	100%

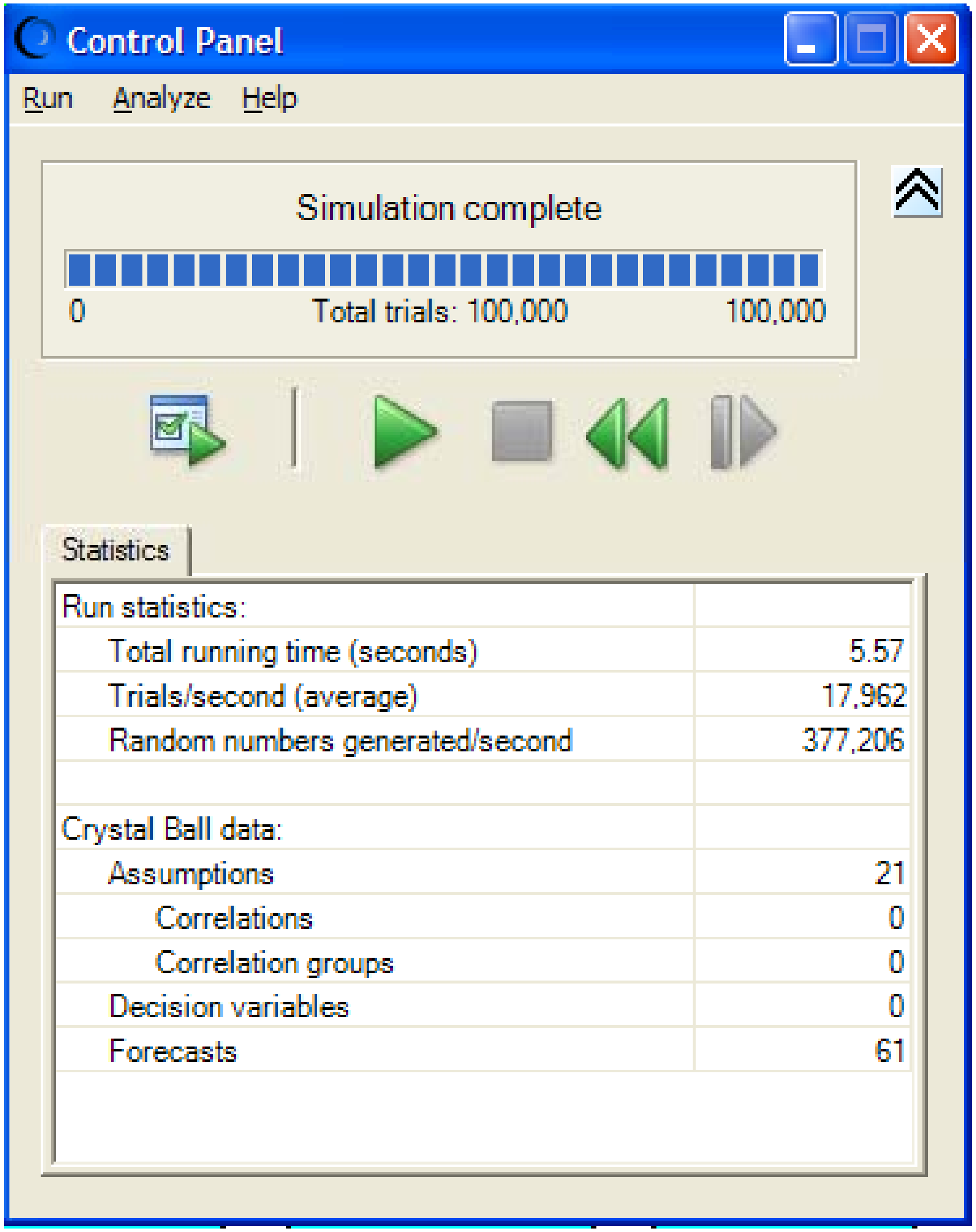
Each of these blue cells were identified as outcomes whose resulting distributions will be studied



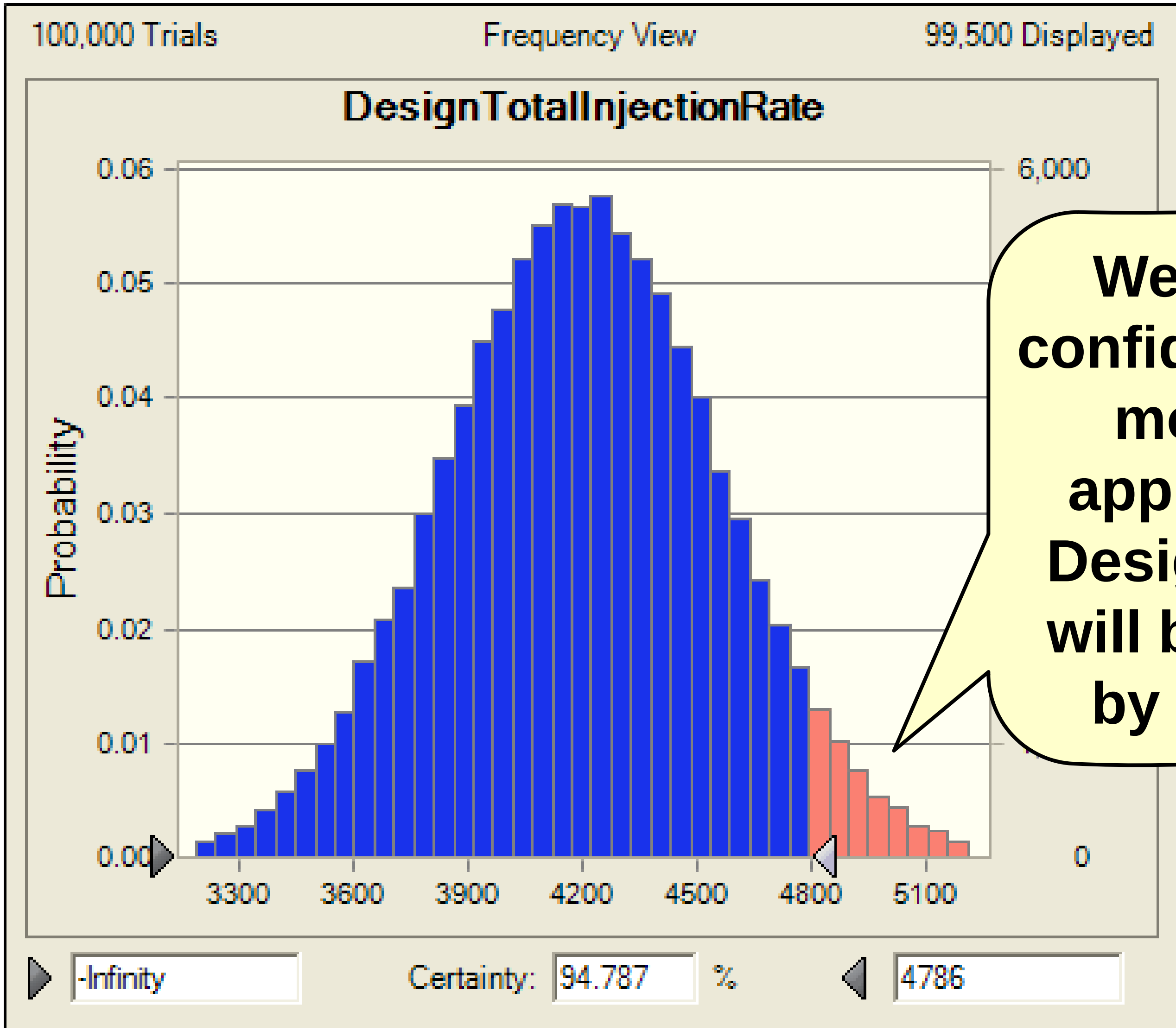
Each of these blue cells were identified as outcomes whose resulting distributions will be studied

Phase Injected	All numbers per 1MSLOC	Total Injection Rate by Activity	Escape Rate by Activity	Escape Rate by All Activities	Activity Injection Rate by Total Defects %	Escape Rate by Activity %	Escape Rate by All Activities %
	Reqts Activity	830	330	330	7%	40%	40%
	Design Activity	4200	2200	2450	36%	52%	54%
	Code Activity	6340	2540	4680	54%	40%	53%
	Integration Test	355	55	1475	3%	15%	29%
	System Test	100	0	275	1%	0%	17%
	User Test	5	0	0	0%	0%	0%

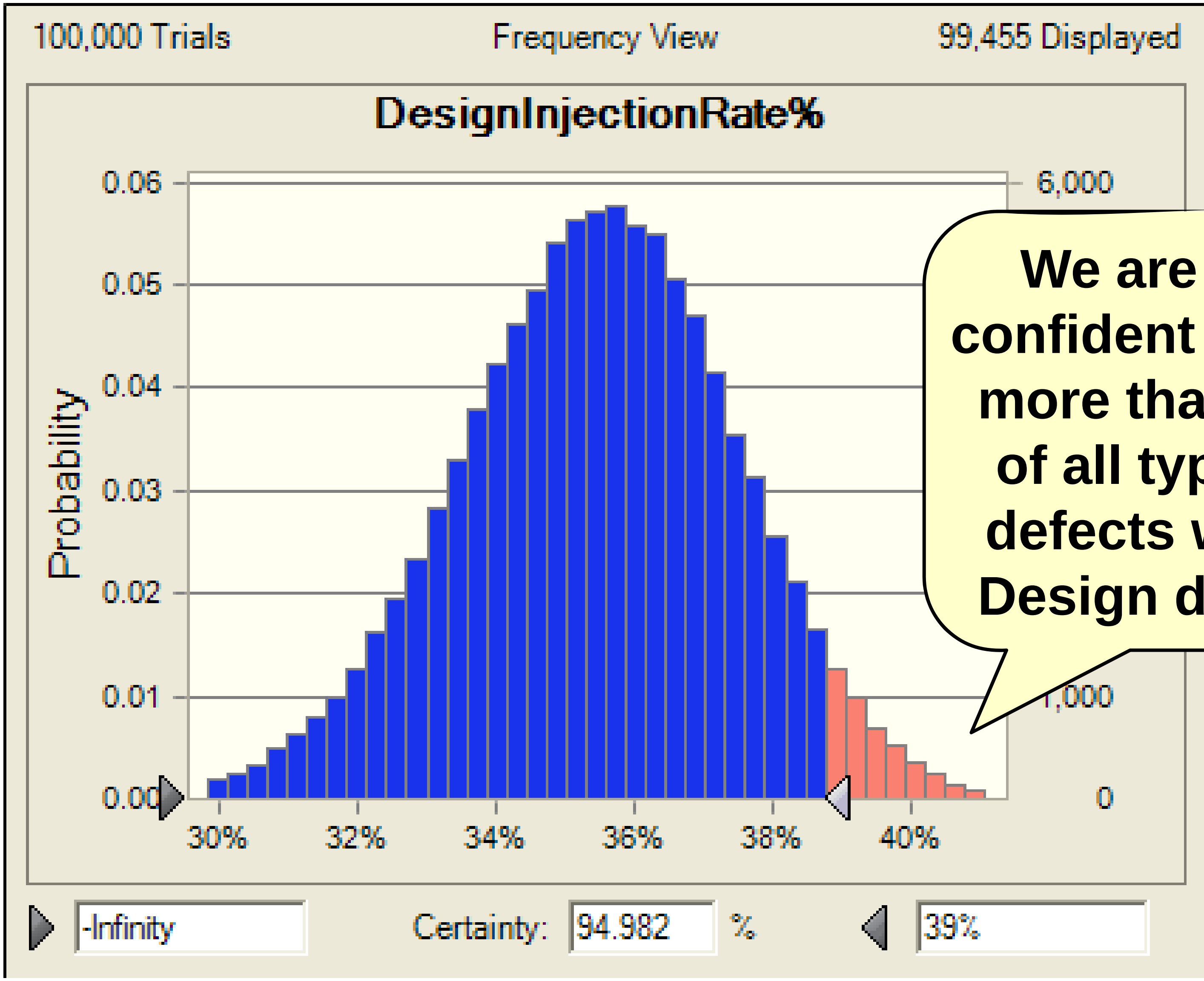

MonteCarlo-Escaped-Defects-v010.xlsx - Microsoft Excel
Data
Review
View
Add-Ins
Crystal Ball
Start
Stop
Reset
Step
Tools
Save or Restore
Run Preferences
Run
View Charts
Create Report
Extract Data
Analyze
Help
Resources
About
Help
Run Preferences
Trials
Sampling
Speed
Options
Statistics
Number of trials to run:
100000
Stop on calculation errors
Stop when precision control limits are reached
Confidence level:
95
%
OK
Cancel
Defaults...
Help



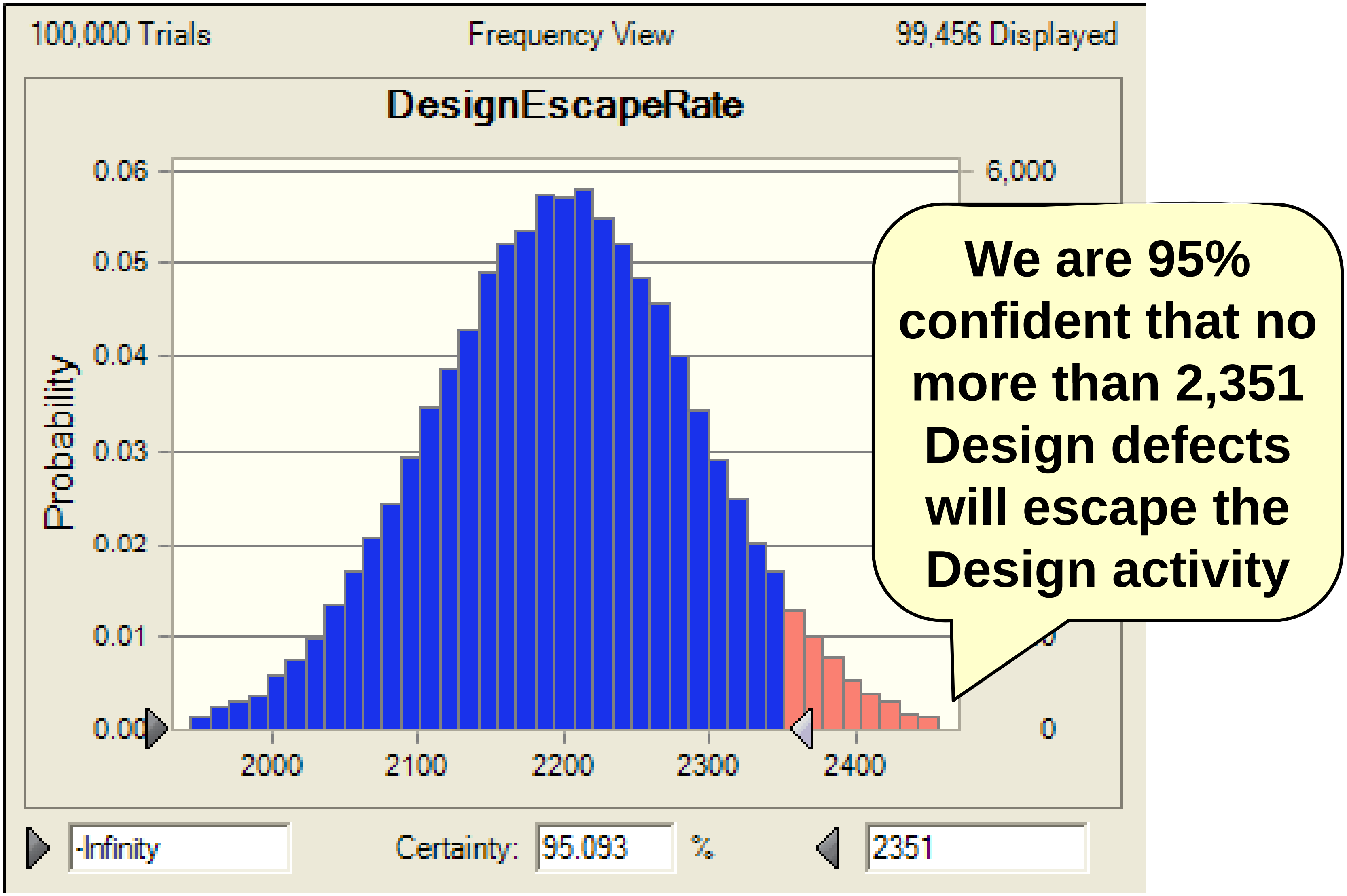
Standard simulation summary results

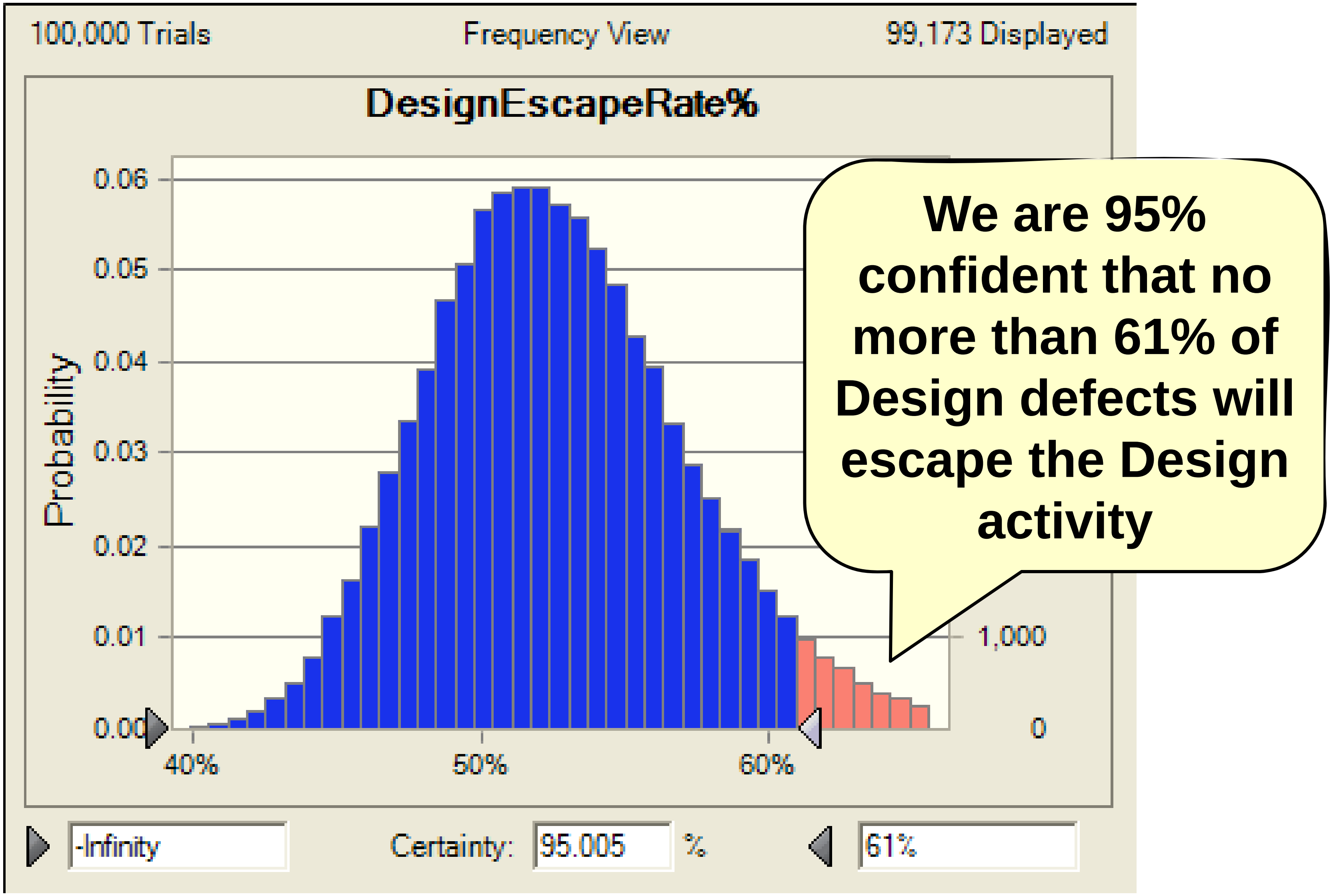


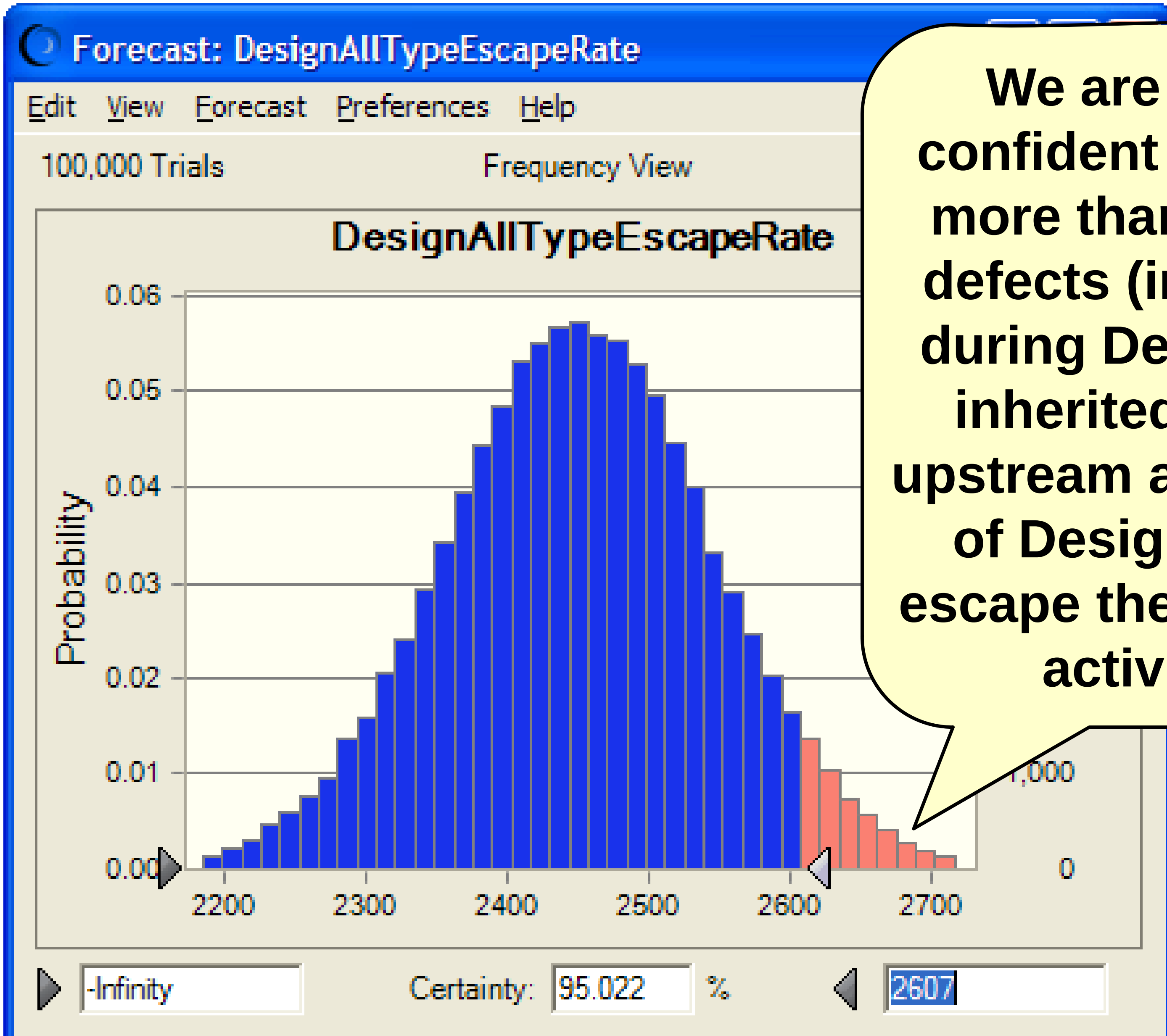
We are 95% confident that no more than approx. 4,786 Design defects will be injected by a project



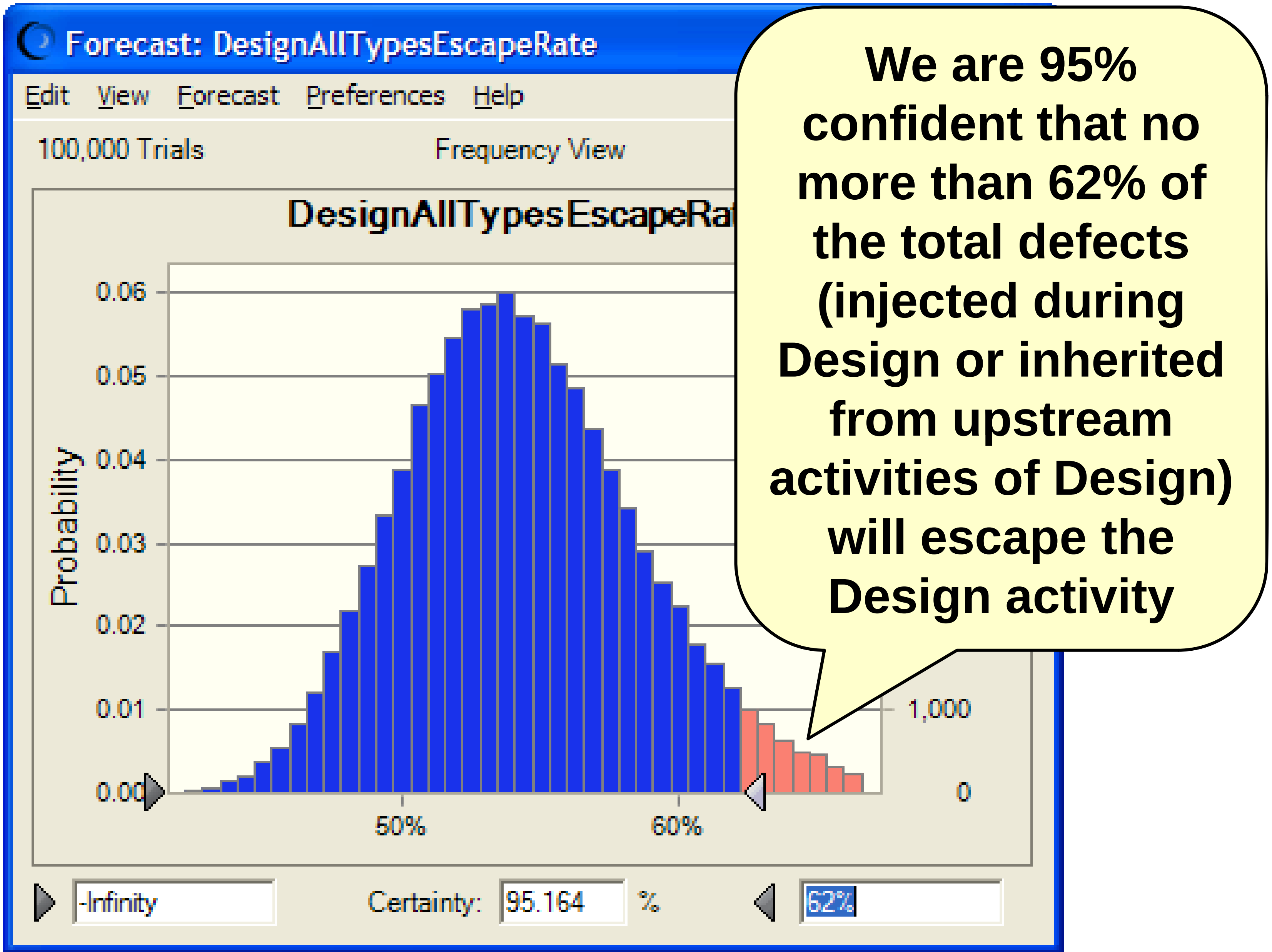
We are 95% confident that no more than 39% of all types of defects will be Design defects


100,000 Trials
Frequency View
99,456 Displayed
DesignEscapeRate
Probability
0.06
0.05
0.04
0.03
0.02
0.01
0.00
6,000
0
2000
2100
2200
2300
2400
-Infinity
Certainty: 95.093 %
2351
We are 95% confident that no more than 2,351 Design defects will escape the Design activity

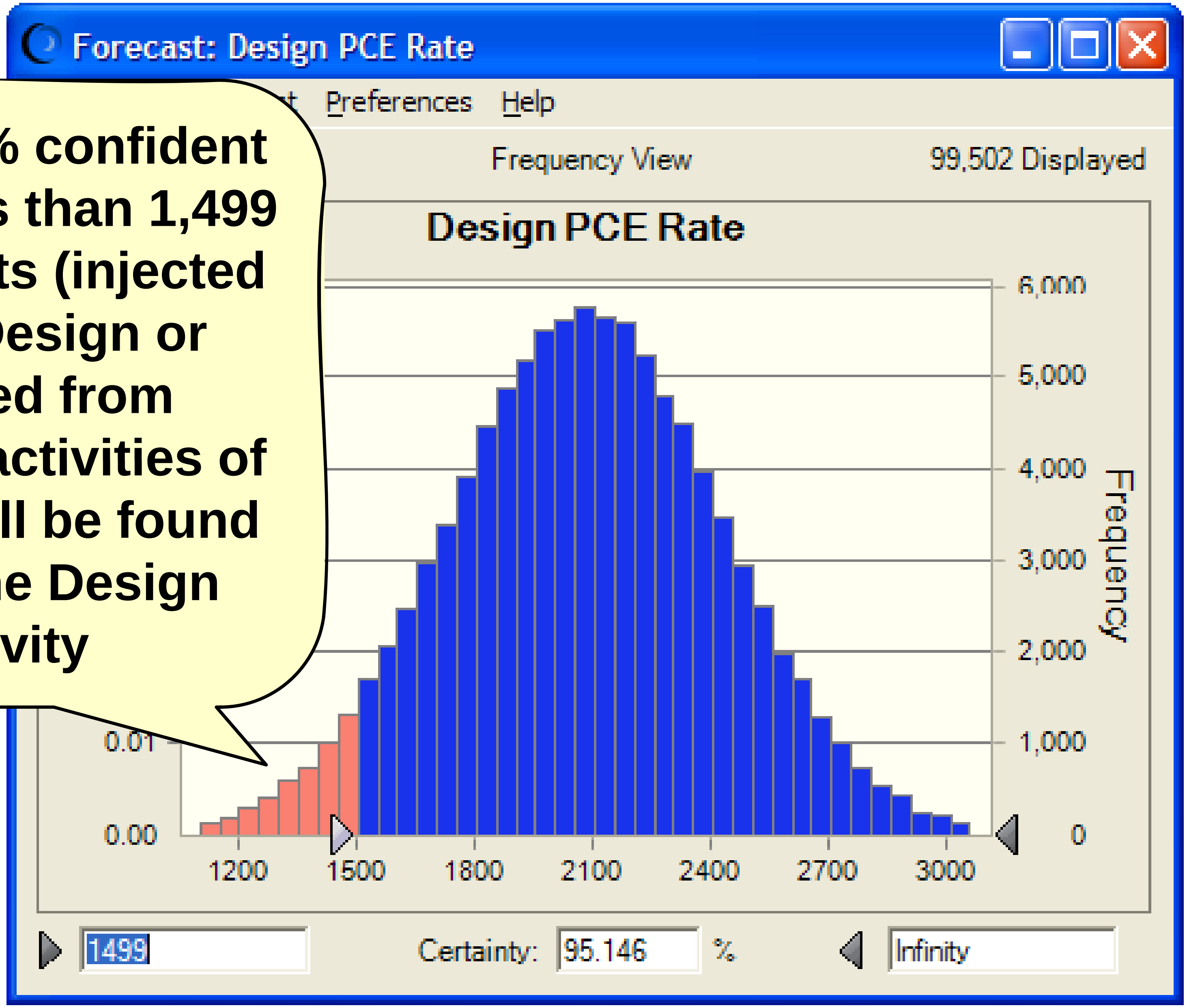

100,000 Trials
Frequency View
99,173 Displayed
DesignEscapeRate%
Probability
0.06
0.05
0.04
0.03
0.02
0.01
0.00
1,000
0
40%
50%
60%
-Infinity
Certainty: 95.005 %
61%
We are 95% confident that no more than 61% of Design defects will escape the Design activity



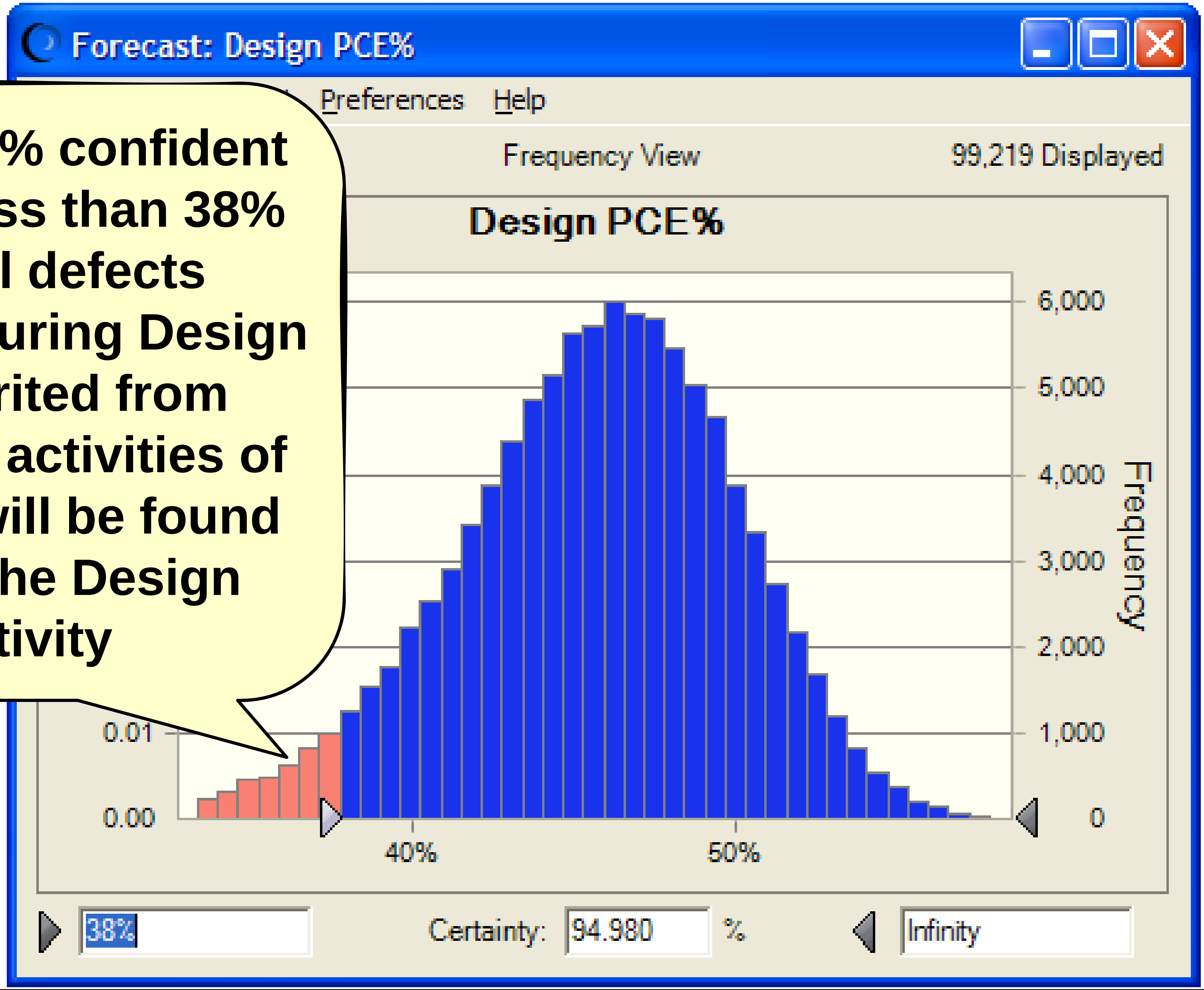
We are 95% confident that no more than 2,607 defects (injected during Design or inherited from upstream activities of Design) will escape the Design activity

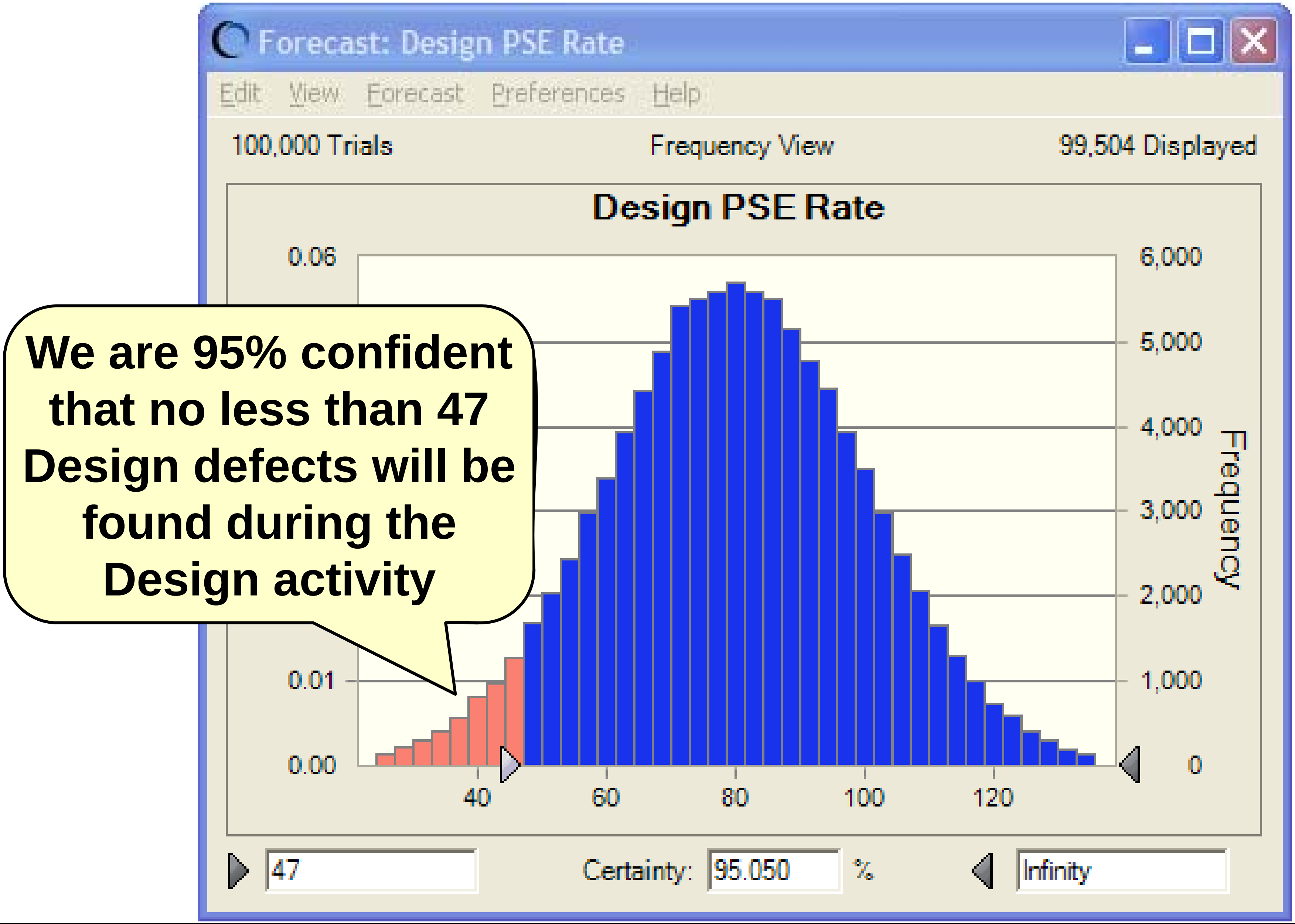

Forecast: DesignAllTypesEscapeRate
Edit View Forecast Preferences Help
100,000 Trials
Frequency View
DesignAllTypesEscapeRat
Probability
0.06
0.05
0.04
0.03
0.02
0.01
0.00
1,000
0
50%
60%
-Infinity
Certainty: 95.164 %
62%
We are 95% confident that no more than 62% of the total defects (injected during Design or inherited from upstream activities of Design) will escape the Design activity

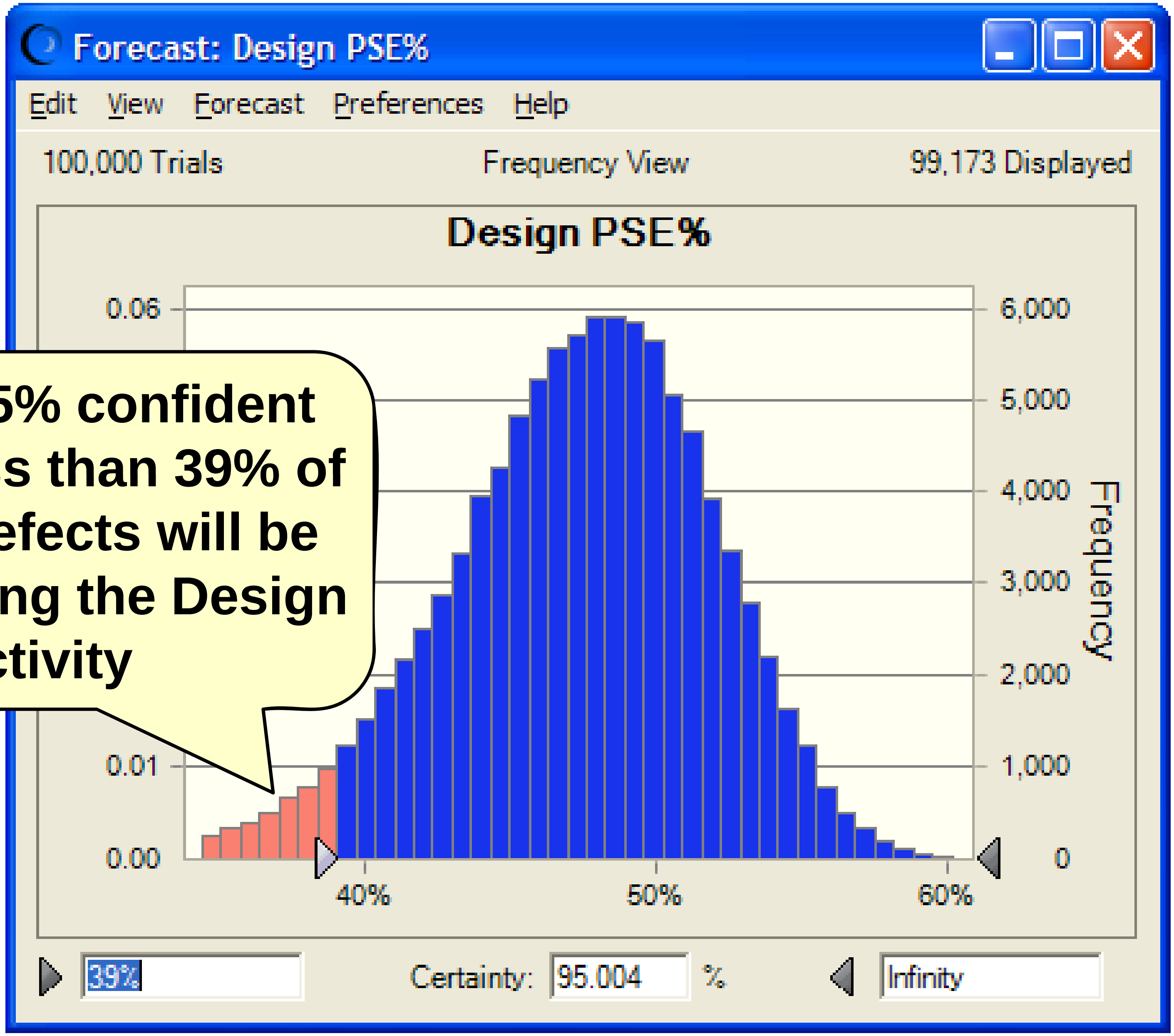
We are 95% confident that no less than 1,499 total defects (injected during Design or inherited from upstream activities of Design) will be found during the Design activity



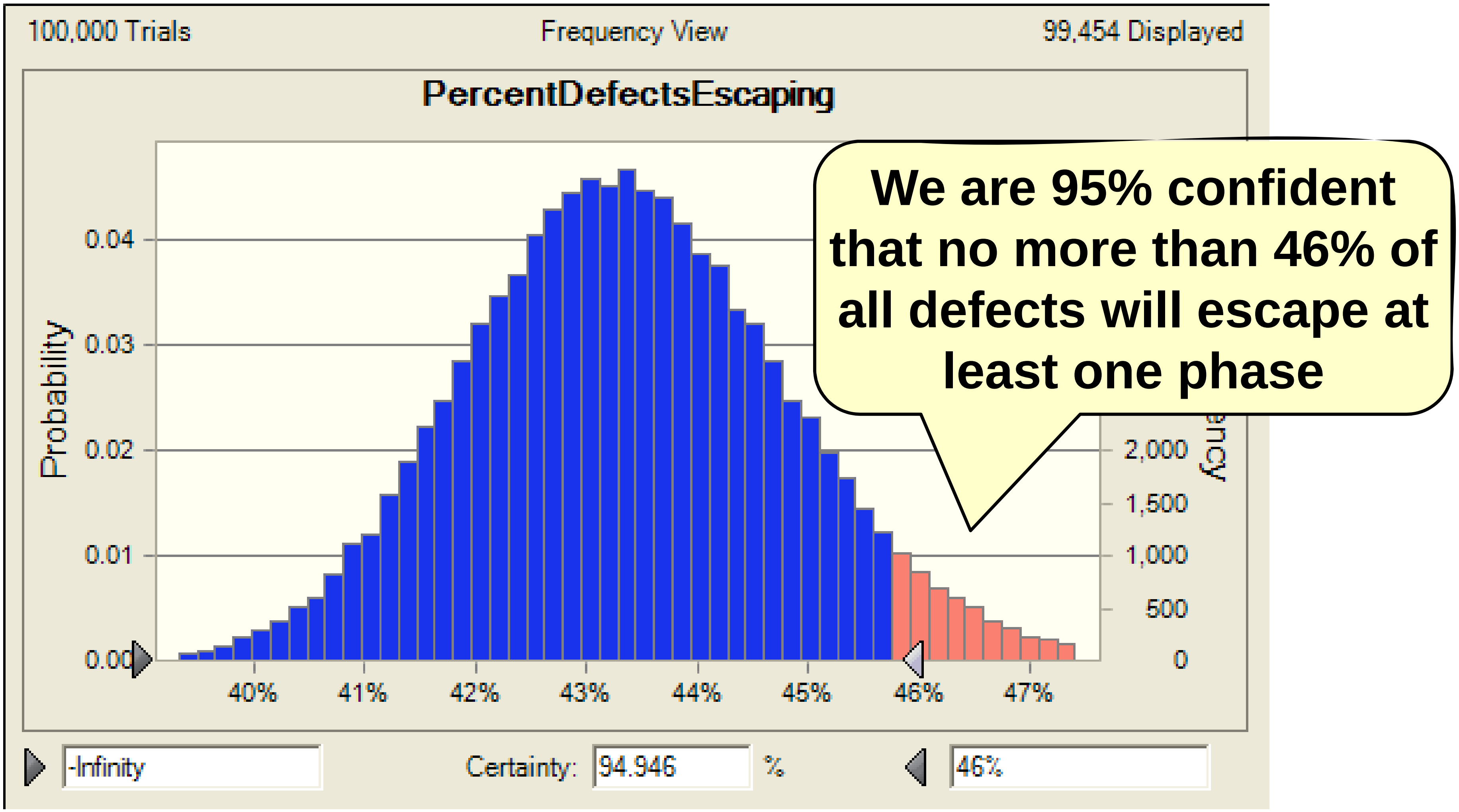
We are 95% confident that no less than 38% of total defects (injected during Design or inherited from upstream activities of Design) will be found during the Design activity

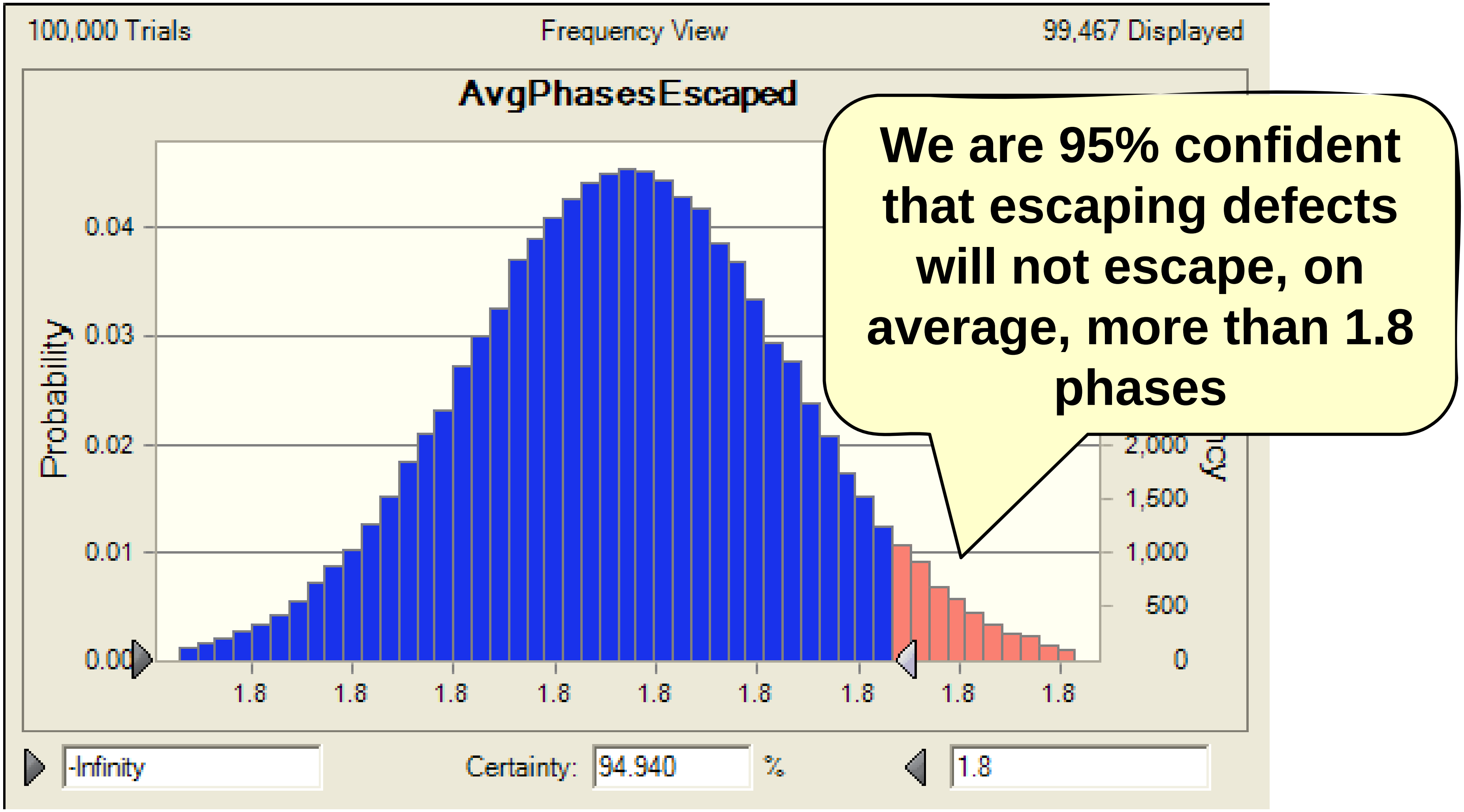


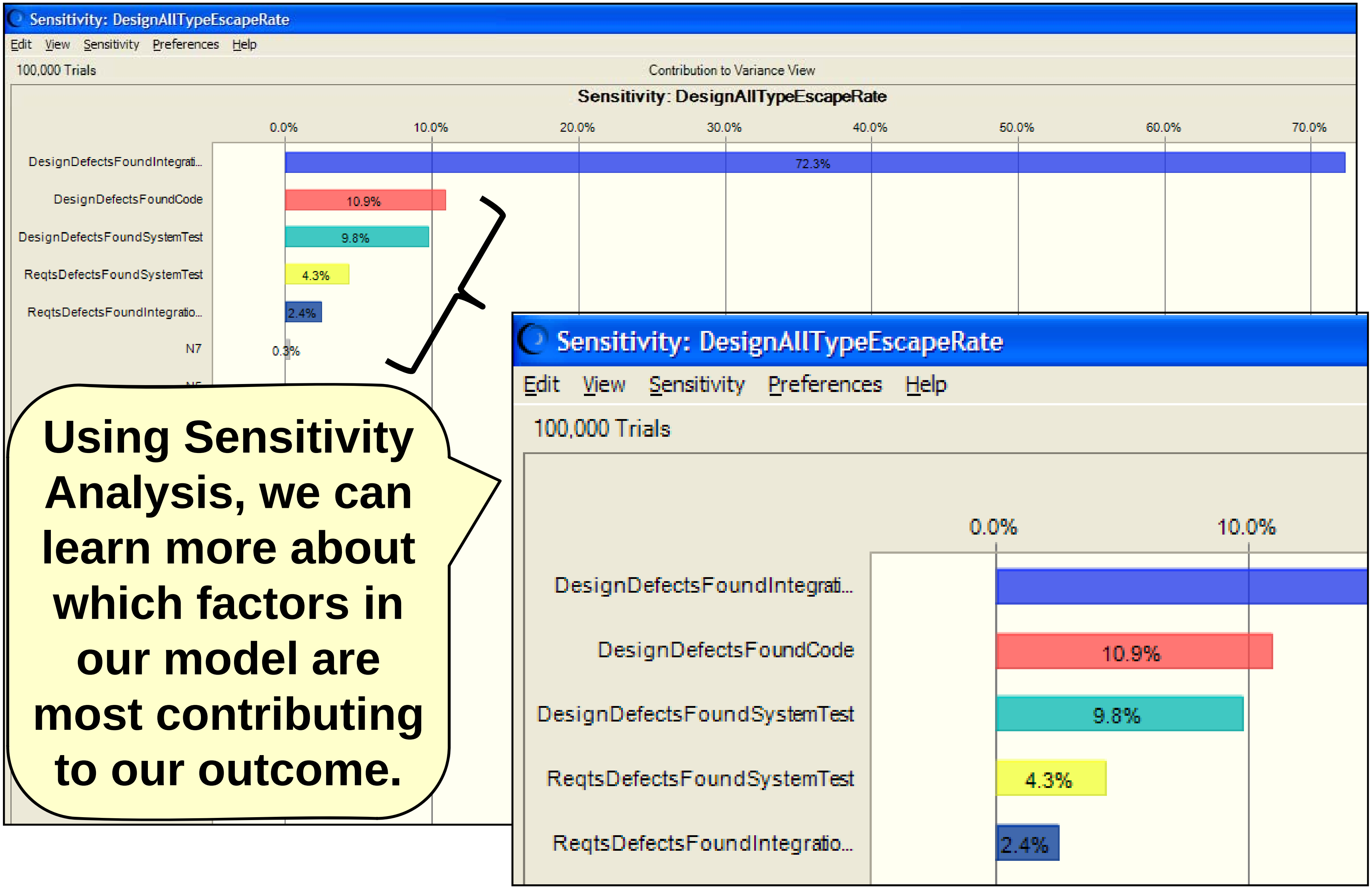

Forecast: Design PSE Rate
Edit View Forecast Preferences Help
100,000 Trials
Frequency View
99,504 Displayed
Design PSE Rate
0.06
0.01
0.00
6,000
5,000
4,000
3,000
2,000
1,000
0
Frequency
40
60
80
100
120
47
Certainty: 95.050 %
Infinity
We are 95% confident that no less than 47 Design defects will be found during the Design activity



We are 95% confident that no less than 39% of Design defects will be found during the Design activity


100,000 Trials
Frequency View
99,454 Displayed
PercentDefectsEscaping
Probability
0.04
0.03
0.02
0.01
0.00
2,000
1,500
1,000
500
0
40%
41%
42%
43%
44%
45%
46%
47%
-Infinity
Certainty: 94.946 %
46%
We are 95% confident that no more than 46% of all defects will escape at least one phase


100,000 Trials
Frequency View
99,467 Displayed
AvgPhasesEscaped
Probability
0.04
0.03
0.02
0.01
0.00
2,000
1,500
1,000
500
0
1.8
1.8
1.8
1.8
1.8
1.8
1.8
1.8
1.8
-Infinity
Certainty: 94.940 %
1.8
We are 95% confident that escaping defects will not escape, on average, more than 1.8 phases


Sensitivity: DesignAllTypeEscapeRate
Edit View Sensitivity Preferences Help
100,000 Trials
Contribution to Variance View
Sensitivity: DesignAllTypeEscapeRate
0.0% 10.0% 20.0% 30.0% 40.0% 50.0% 60.0% 70.0%
DesignDefectsFoundIntegrati... 72.3%
DesignDefectsFoundCode 10.9%
DesignDefectsFoundSystemTest 9.8%
ReqtsDefectsFoundSystemTest 4.3%
ReqtsDefectsFoundIntegratio... 2.4%
N7 0.3%
Using Sensitivity Analysis, we can learn more about which factors in our model are most contributing to our outcome.
Sensitivity: DesignAllTypeEscapeRate
Edit View Sensitivity Preferences Help
100,000 Trials
0.0% 10.0%
DesignDefectsFoundIntegrati...
DesignDefectsFoundCode 10.9%
DesignDefectsFoundSystemTest 9.8%
ReqtsDefectsFoundSystemTest 4.3%
ReqtsDefectsFoundIntegratio... 2.4%

What Have We Accomplished?

We transformed a model that used only historical averages and substituted uncertainty distributions for each of the injection and found rates

By doing this, we can establish confident conclusions about our outcomes using their resulting distributions:

- Defect Injection rates by Phase
- Phase Containment and Screening Effectiveness

We also used sensitivity analysis to decide which factors to tackle first to improve each outcome

Planning the Requirements Buildup

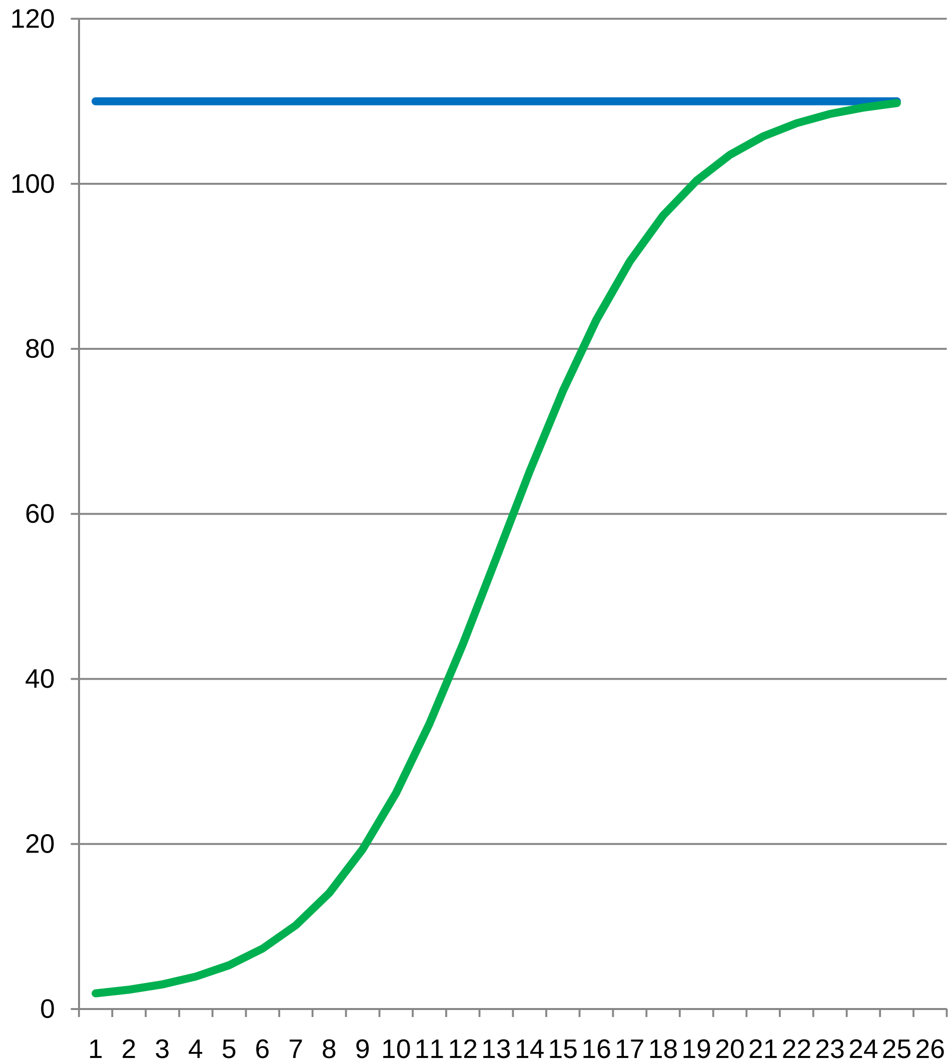
Generate Plan

Generate schedule based on higher level PPMs which help determine milestones and variation base slack

Generate detailed PPMs to predict performance during this phase

Note: these steps will be repeated periodically (such as at phase or other selected milestones) and on an as needed basis

Elicitation – Requirements Buildup – Predicted

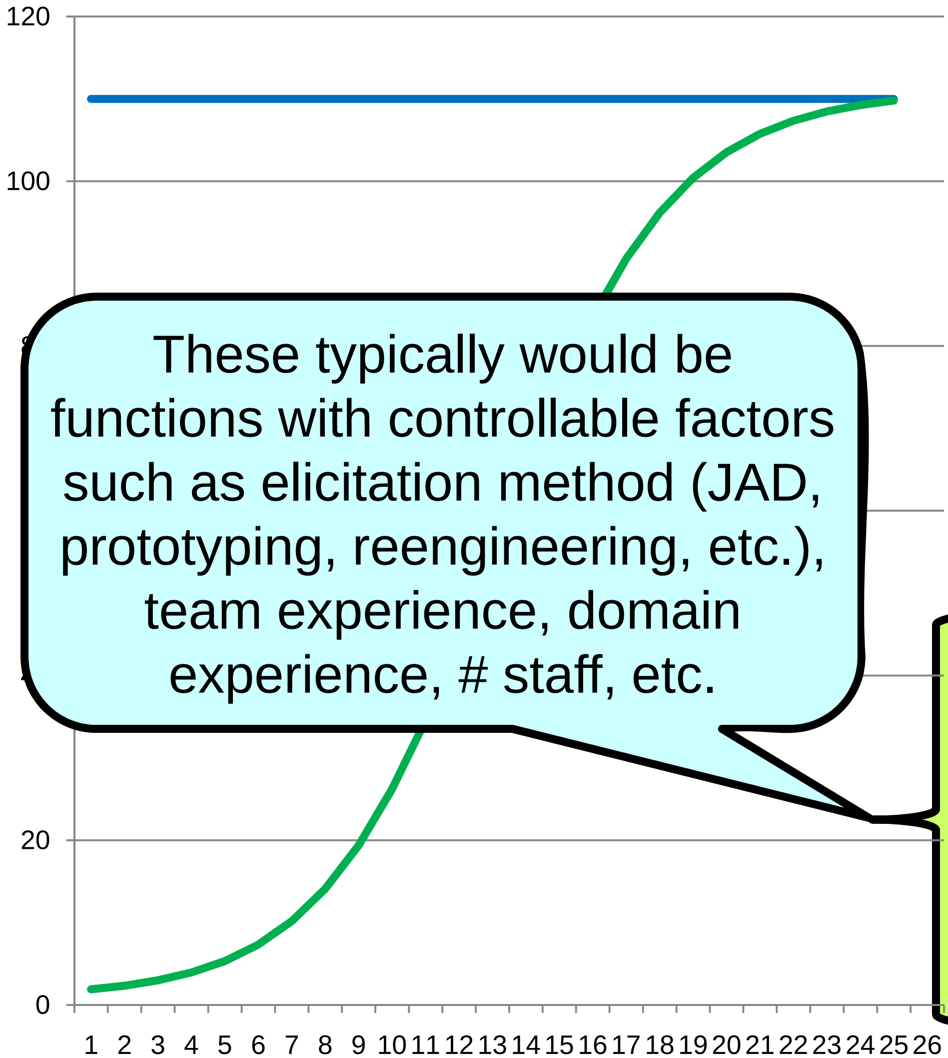


Generalized Logistic (or Richard's)

$$Y = A + \frac{C}{\left(1 + T e^{-B(x-M)}\right)^{1/T}}$$

- **x** = time.
- **A** controls the lower asymptote,
- **C** controls the upper asymptote,
- **M** controls the time of maximum growth,
- **B** controls the growth rate, and
- **T** controls where maximum growth occurs - nearer the lower or upper asymptote

Elicitation – Requirements Buildup – Predicted



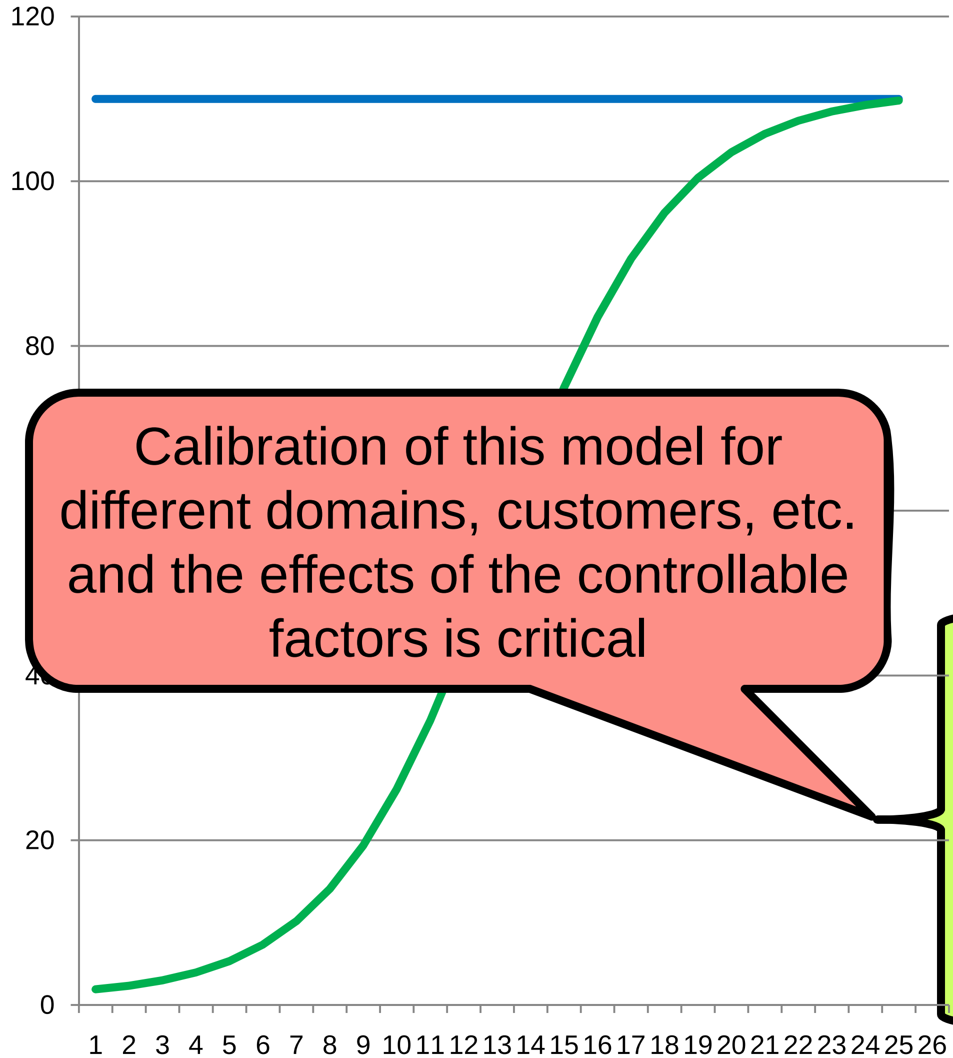
These typically would be functions with controllable factors such as elicitation method (JAD, prototyping, reengineering, etc.), team experience, domain experience, # staff, etc.

Generalized Logistic (or Richard's)

$$Y = A + \frac{C}{(1 + Te^{-B(x-M)})^{1/T}}$$

- **x** = time.
- **A** controls the lower asymptote,
- **C** controls the upper asymptote,
- **M** controls the time of maximum growth,
- **B** controls the growth rate, and
- **T** controls where maximum growth occurs - nearer the lower or upper asymptote

Elicitation – Requirements Buildup – Predicted



Generalized Logistic (or Richard's)

$$Y = A + \frac{C}{(1 + Te^{-B(x-M)})^{1/T}}$$

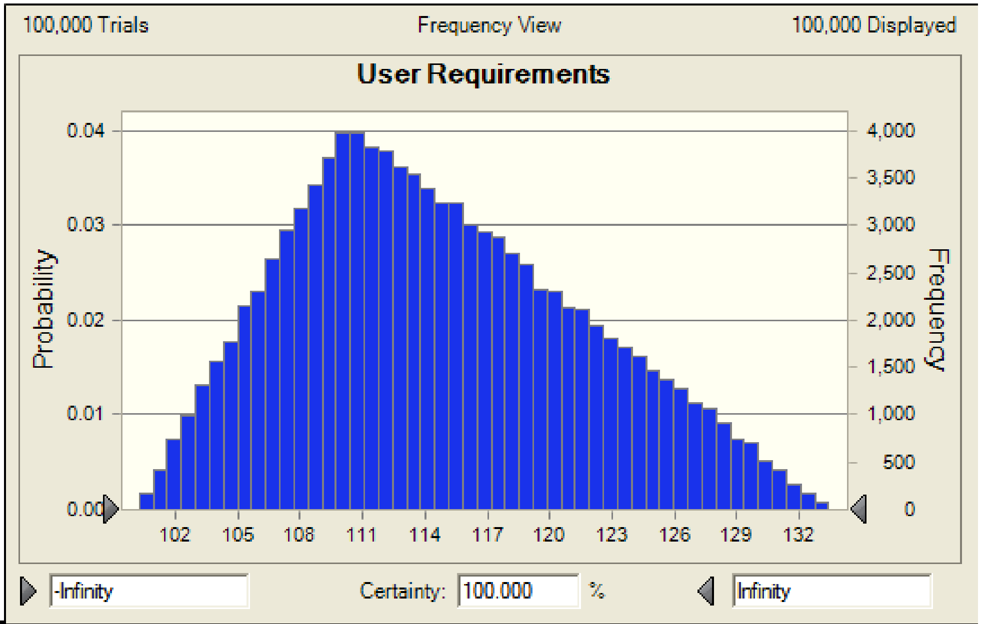
- **x** = time.
- **A** controls the lower asymptote,
- **C** controls the upper asymptote,
- **M** controls the time of maximum growth,
- **B** controls the growth rate, and
- **T** controls where maximum growth occurs - nearer the lower or upper asymptote

Risk in Plan

Use PPMs to judge overall risk in the plan

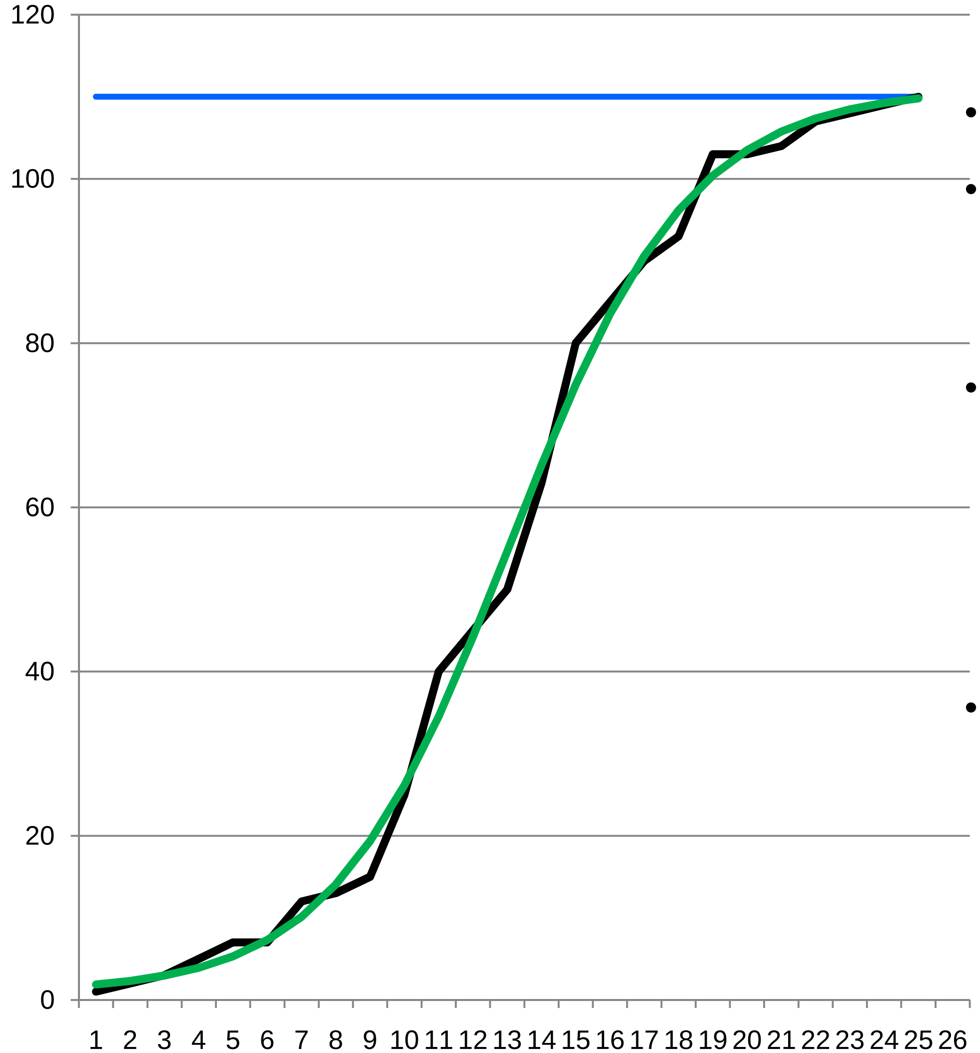
May use Monte Carlo simulation in the schedule to better understand

- Schedule based sources of risk
- Effects of risks on the schedule



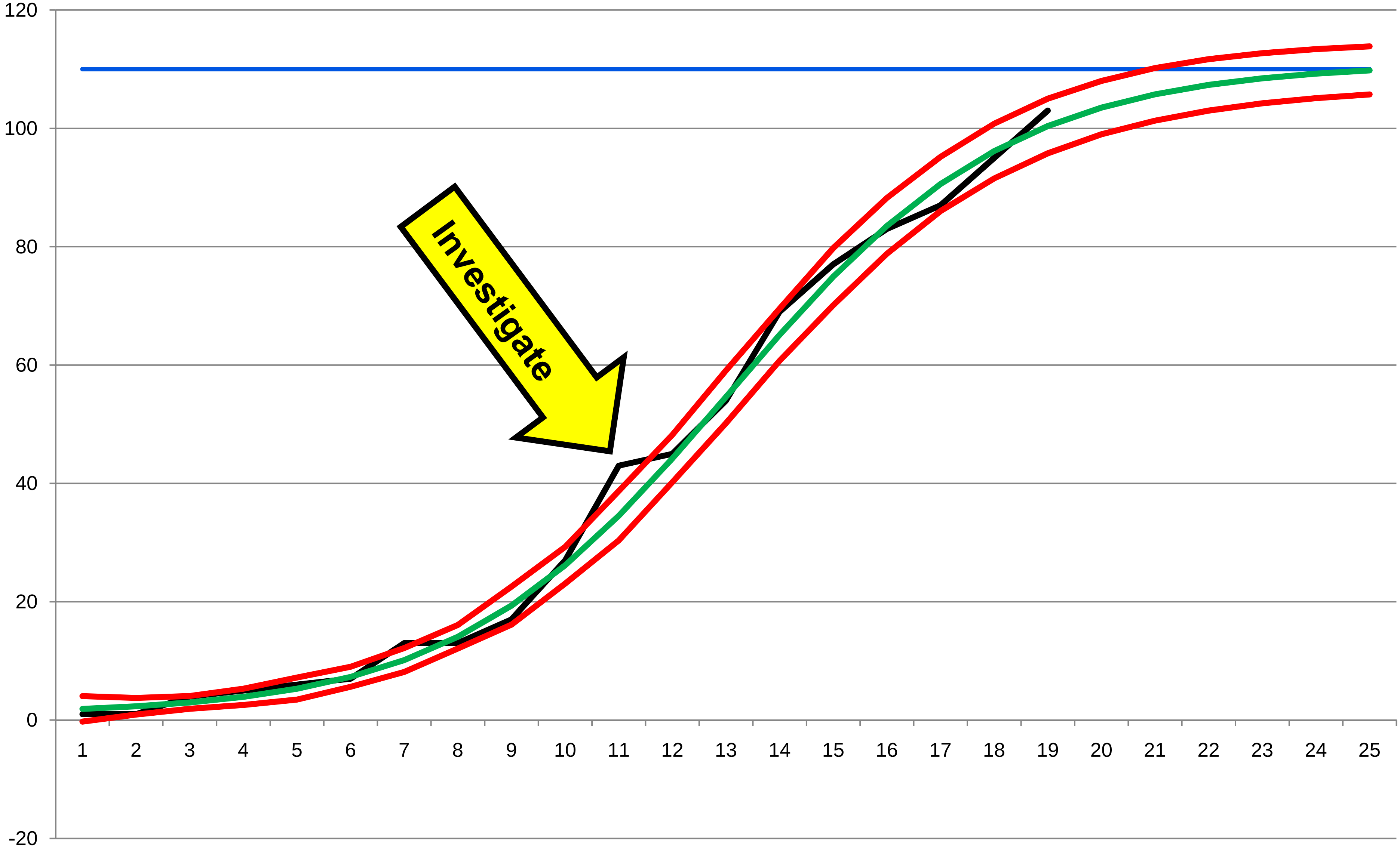
Robert Stoddard
Rusty Young

Elicitation – Requirements Buildup – Monitor

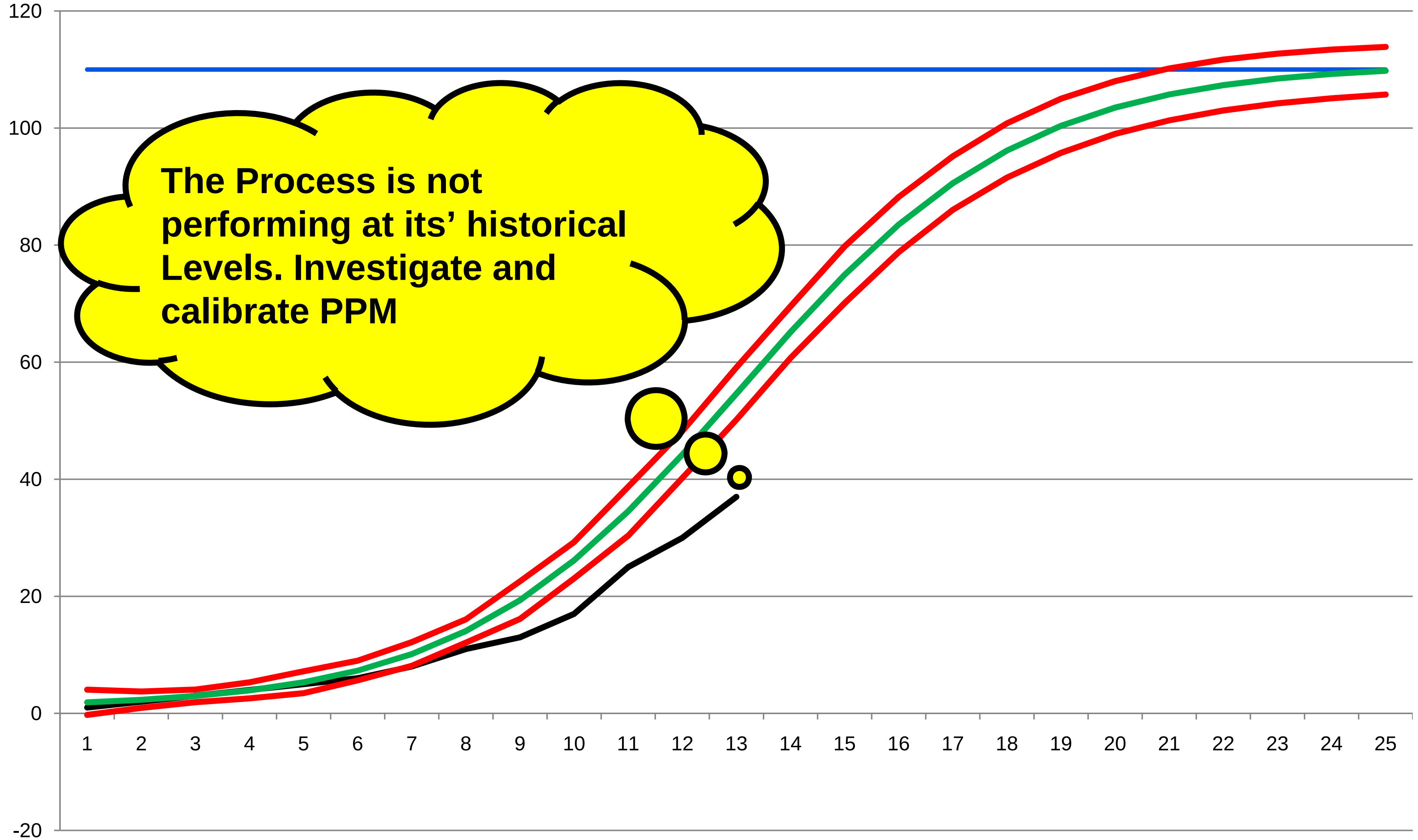


- Monitor the buildup
- Flattening means you are reaching the point of diminishing returns for elicitation
- Significant difference between predicted and actual upper asymptote indicate a potential misunderstanding of the system to be built
- If actuals show significant variation from predicted, re fit curve for new prediction
  - Calculate an appropriate Prediction Interval (PI) to aid in detection of anomalous conditions

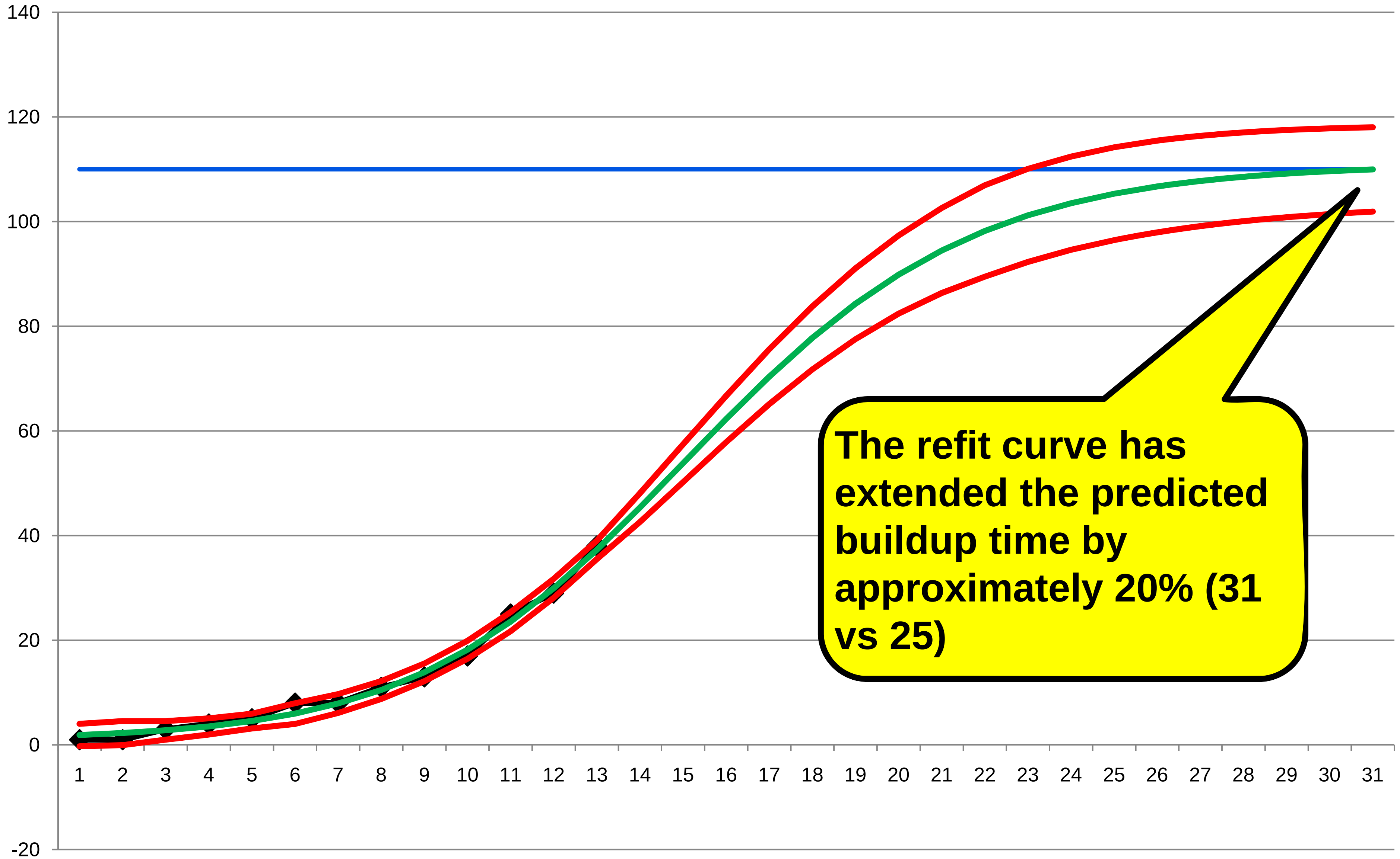
Elicitation – Requirements Buildup – Example 1



Elicitation – Requirements Buildup – Example 2a



Elicitation – Requirements Buildup – Example 2b



Elicitation – Requirements Buildup – Notes

Requires a strong consistent requirements elicitation process

- Different standard curves for different elicitation processes such as JAD, prototyping, etc.
- Curve shape parameters can be influenced by context -- re-engineering vs green field, well understood vs non-well understood domain, experience, etc.

Consider a measurement systems error analysis – perhaps using Gage R&R to ensure consistent buildup counts

Requires a good prediction of size

Can be beneficial with good size prediction and then fitting the curve as you gather data

- It will take time before variation in the buildup time minimizes

Another Example Process Performance Model (PPM) in the Requirements Phase

The Situation in the Requirements Phase

Our products are comprised of 40-60 features

We assign each feature a small development team to develop the feature “cradle to grave”

These feature teams operate in overlapping lifecycles within an overall product incremental waterfall lifecycle model (thus, different features will be added in each new increment)

<u>OUR NEED</u>: A PPM that will let each feature team predict the number of requirements defects to be experienced throughout the lifecycle of the feature development

Details of the Requirements Phase PPM

The outcome, Y, is the predicted number of Requirements defects for a given feature team

The x factors used to predict the Requirements defects are:

- x1: Req'ts Volatility (continuous data)
- x2: Risk of Incomplete Req'ts (nominal data)
- x3: Risk of Ambiguous Req'ts (nominal data)
- x4: Risk of Non-Testable Req'ts (nominal data)
- x5: Risk of Late Req'ts (nominal data)

Background Information on the Data

We collected historical data (of the Y and the x's) for a large volume of feature teams

For the x2 thru x5 factors, the feature team leader would historically check off as "yes" or "no" depending on whether they felt that the specific risk significantly impacted the feature team cost, schedule or quality

Operational definitions and training were conducted to ensure consistency and repeatability among the feature team leads

Development of the Req’ts Phase PPM - 1

Volatility	RiskofIncompleteness	RiskofAmbiguity	RiskofNonTestability
0.12	1	1	0

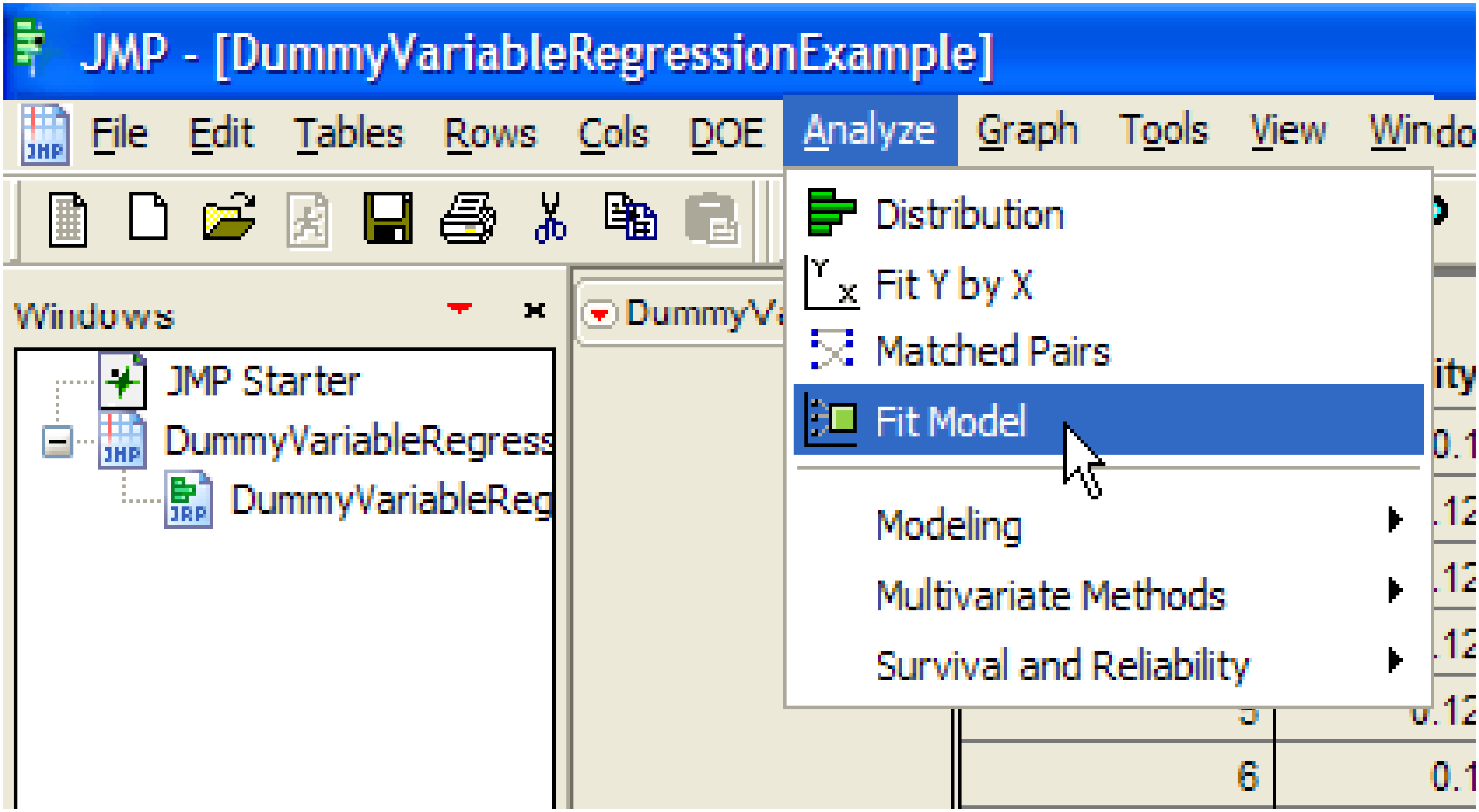
X1: Volatility shown in decimal form

All of the risks are rated either 0 or 1, with 0 being the absence of the risk and 1 being the presence of the risk

The Y outcome Is the Number Of Reqts Defects

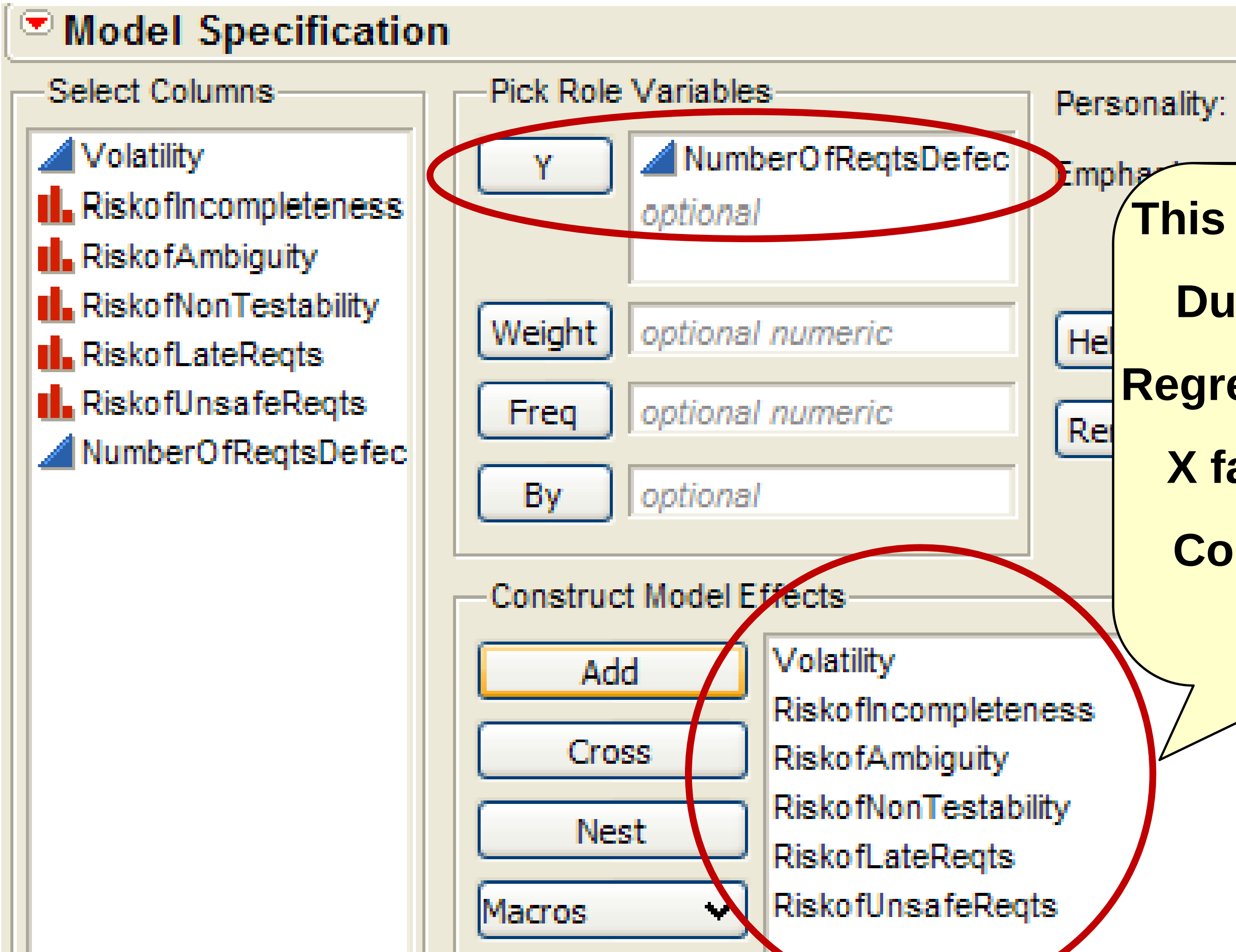
RiskofLateReqts	RiskofUnsafeReqts	NumberOfReqtsDefects
0	0	12

Development of the Req’ts Phase PPM - 2



Robert Stoddard
Rusty Young

Development of the Req'ts Phase PPM - 3



This will accomplish Dummy Variable Regression to handle X factors that are Continuous and Discrete

Development of the Req'ts Phase PPM - 4

Summary of Fit

RSquare	0.940697
RSquare Adj	0.935386
Root Mean Square Error	0.28853
Mean of Response	13.83784
Observations (or Sum Wgts)	74

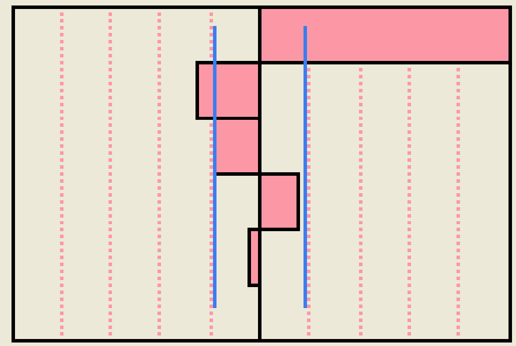
Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	6	88.476347	14.7461	177.1312
Error	67	5.577707	0.0832	Prob > F
C. Total	73	94.054054		<.0001*

Development of the Req'ts Phase PPM - 5

Sorted Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Volatility	32.860031	2.9708	11.06	<.0001*
RiskofIncompleteness[0]	-0.240651	0.088552	-2.72	0.0084*
RiskofLateReqts[0]	-0.107577	0.053802	-2.00	0.0496*
RiskofUnsafeReqts[0]	0.1647047	0.095776	1.72	0.0901
RiskofAmbiguity[0]	-0.031284	0.078746	-0.40	0.6924
RiskofNonTestability[0]	0.0017969	0.058641	0.03	0.9756



Development of the Req’ts Phase PPM - 6

Prediction Expression

$$
\begin{aligned}
&7.5937439072846 \\
&+32.8600305113136 * \text{Volatility} \\
&+\text{Match}\left(\text{RiskofIncompleteness}\right)\begin{pmatrix} \text{"0"} \Rightarrow -0.2406507338728 \\ \text{"1"} \Rightarrow 0.24065073387281 \\ \text{else} \Rightarrow . \end{pmatrix} \\
&+\text{Match}\left(\text{RiskofAmbiguity}\right)\begin{pmatrix} \text{"0"} \Rightarrow -0.0312842155195 \\ \text{"1"} \Rightarrow 0.03128421551947 \\ \text{else} \Rightarrow . \end{pmatrix} \\
&+\text{Match}\left(\text{RiskofNonTestability}\right)\begin{pmatrix} \text{"0"} \Rightarrow 0.00179694498247 \\ \text{"1"} \Rightarrow -0.0017969449825 \\ \text{else} \Rightarrow . \end{pmatrix} \\
&+\text{Match}\left(\text{RiskofLateReqts}\right)\begin{pmatrix} \text{"0"} \Rightarrow -0.107577311748 \\ \text{"1"} \Rightarrow 0.10757731174804 \\ \text{else} \Rightarrow . \end{pmatrix} \\
&+\text{Match}\left(\text{RiskofUnsafeReqts}\right)\begin{pmatrix} \text{"0"} \Rightarrow 0.16470472563596 \\ \text{"1"} \Rightarrow -0.164704725636 \\ \text{else} \Rightarrow . \end{pmatrix}
\end{aligned}
$$

Intended Use of this Req'ts PPM

Once we decide on the final form of the PPM, we will use it in two primary ways:

1) At the beginning of each feature team kickoff, the team will anticipate the values for the x factors (x1 … x5). They will evaluate the PPM at these values to predict the number of Req'ts defects. If this prediction is unacceptable, they will take immediate action to address one or more of the x factors.

2) During the development, the feature team will periodically re-assess the anticipated values of the x factors and repeat the actions of step 1 above.

Updating this Req’ts PPM

As more feature teams develop features, they will continue to record the data for the x factors and the resulting Y outcome of number of Req’ts Defects

When a group of feature teams have finished the lifecycle and have recorded their data, the organization may choose to add their data to the existing data set and then repeat the exercise of developing the dummy variable regression equation.

Ultimately, the organization may want to segment the feature teams by type and conduct this analysis for each segment.

An Example Process Performance Model (PPM) during the Design Phase

Software Engineering Institute | Carnegie Mellon

The Situation in Design

The Design team is faced with modifying legacy software in addition to developing new software.

A major issue that can have disastrous effects on projects is the idea of “brittleness” of software. In a nutshell, software becomes more “brittle” over time as it is changed and experiences a drifting usage model.

<u>**OUR NEED**</u>: A PPM used by each feature team during design to predict how “brittle” software is, and subsequently to make the correct design decisions regarding degree of modification vs rewrite from scratch.

Details of the Software Brittleness PPM

The outcome, Y, is the measure of software brittleness, measured on an arbitrary scale of 0 (low) to 100 (high), which will be treated as continuous data

The x factors used in this prediction example are the following:

- Unit path complexity
- Unit data complexity
- Number of times the unit code files have been changed
- Number of unit code changes not represented in Design document updates

Background Information on the Data

We collected historical data from feature teams on their code units.   The data, related to the first four x factors, are maintained by the CM system using automated tools tracking this data each time new code file versions are checked in.

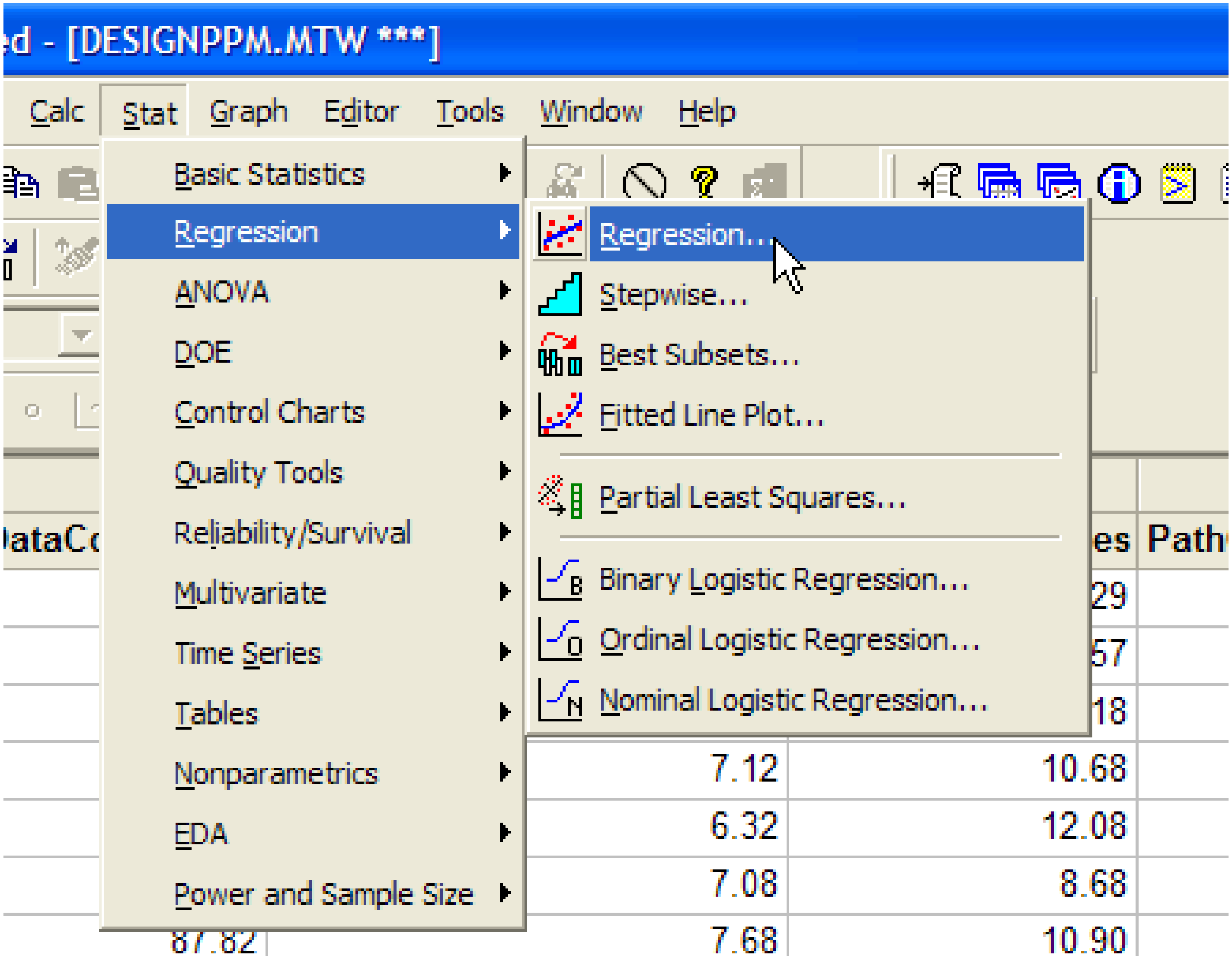
- Unit path complexity
- Unit data complexity
- Number of times the unit code files have been changed

We also have access to problem reporting and inspection databases which provide us with a number of issues reported against individual code units. Finally, we have “Brittleness” values for each unit of code that were assigned by a different empirical exercise with domain experts.

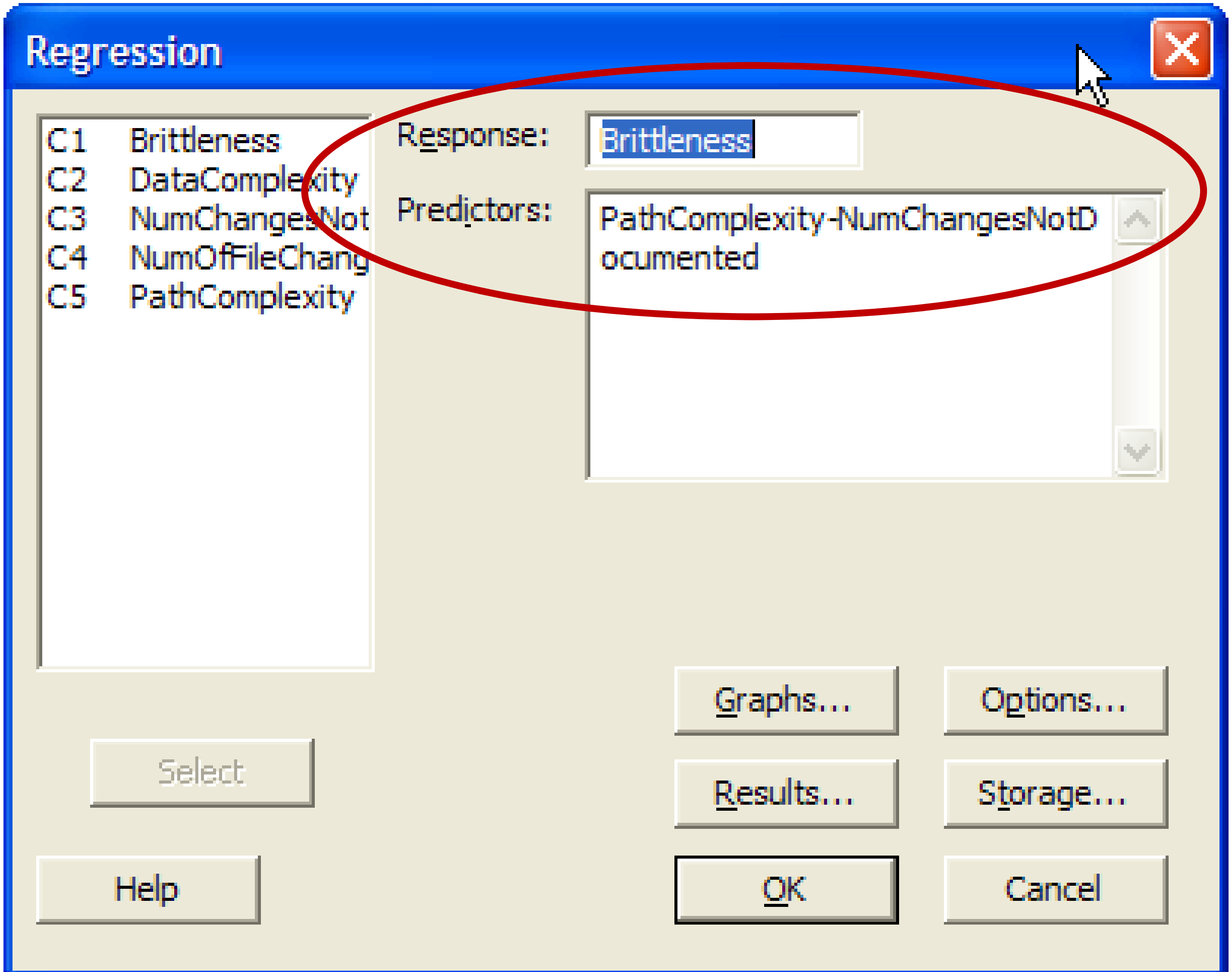
Development of the Brittleness PPM - 1

C1	C2	C3	C4	C5
Brittleness	DataComplexity	NumChangesNotDocumented	NumOfFileChanges	PathComplexity
69.62	45.68	8.13	10.29	19.98
67.59	98.09	5.95	11.57	22.88
64.59	37.02	6.12	16.18	22.52
62.95	36.57	7.12	10.68	20.55
60.58	24.28	6.32	12.08	20.98
67.44	72.53	7.08	8.68	19.75
73.75	87.82	7.68	10.90	19.52
67.00	33.06	6.77	8.97	21.46
63.06	32.84	5.70	12.81	19.06
64.15	38.08	6.92	12.90	20.95
74.27	164.08	6.46	13.85	19.72
77.70	110.71	7.81	9.33	22.20
63.63	31.07	6.25	11.05	20.43
61.71	25.43	6.89	11.84	19.76
73.46	28.92	8.38	12.59	23.18
73.54	143.56	6.70	9.80	25.91
61.58	88.74	6.99	5.29	19.38
66.22	16.32	7.16	11.78	19.75

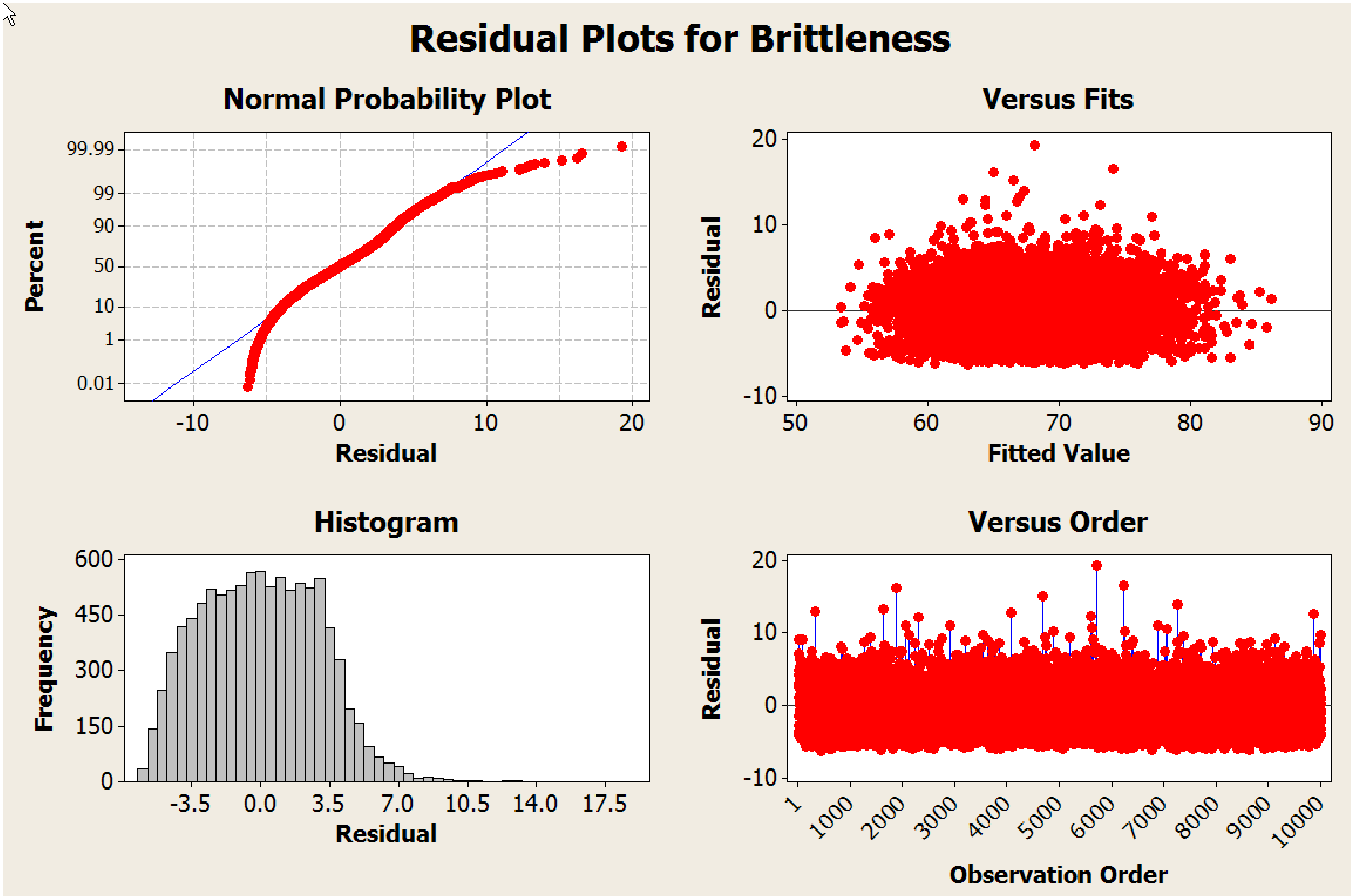
Development of the Brittleness PPM - 2



Development of the Brittleness PPM - 3



Development of the Brittleness PPM - 4



Robert Stoddard
Rusty Young

Development of the Brittleness PPM - 5

Regression Analysis: Brittleness versus PathComplexi, NumOfFileCha, ...

```
The regression equation is
Brittleness = 6.62 + 0.793 PathComplexity + 0.743 NumOfFileChanges
              + 5.04 NumChangesNotDocumented
```

Predictor	Coef	SE Coef	T	P
Constant	6.6168	0.4192	15.78	0.000
PathComplexity	0.79281	0.01173	67.58	0.000
NumOfFileChanges	0.74298	0.01197	62.07	0.000
NumChangesNotDocumented	5.04283	0.04320	116.75	0.000

```
S = 2.99998   R-Sq = 69.0%   R-Sq(adj) = 69.0%
```

Development of the Brittleness PPM - 6

Analysis of Variance

Source	DF	SS	MS	F	P
Regression	3	200594	66865	7429.50	0.000
Residual Error	9996	89963	9		
Total	9999	290557			

Source	DF	Seq SS
PathComplexity	1	41056
NumOfFileChanges	1	36875
NumChangesNotDocumented	1	122663

Intended Use of this Brittleness PPM - 1

Once we decide on the final form of the PPM, we will use it in two primary ways:

1) As each feature team begins software design, the team will collect the x factor information for the software units to be worked on, and then evaluate the PPM at these values to predict the brittleness of the individual software units. If the brittleness prediction is too high, they will decide whether they should continue with modifying legacy code units or rewrite them from scratch.

Intended Use of this Brittleness PPM - 2

Once we decide on the final form of the PPM, we will use it in two primary ways:

2) Ideally, management would have access to this PPM during proposal and planning activities so that predictions of high vs low brittleness may appropriately influence the early estimates.

Updating this Brittleness PPM

As more code units are inspected and tested with corresponding x factor information recorded, the organization will periodically add this new data to the original data set and re-run the regression analysis.

Over time, an analysis relating predicted brittleness of code units to the actual experienced ripple effects of changes to the same code units would be warranted, to ensure the PPM is accurate.

Continuing into the Build Phase

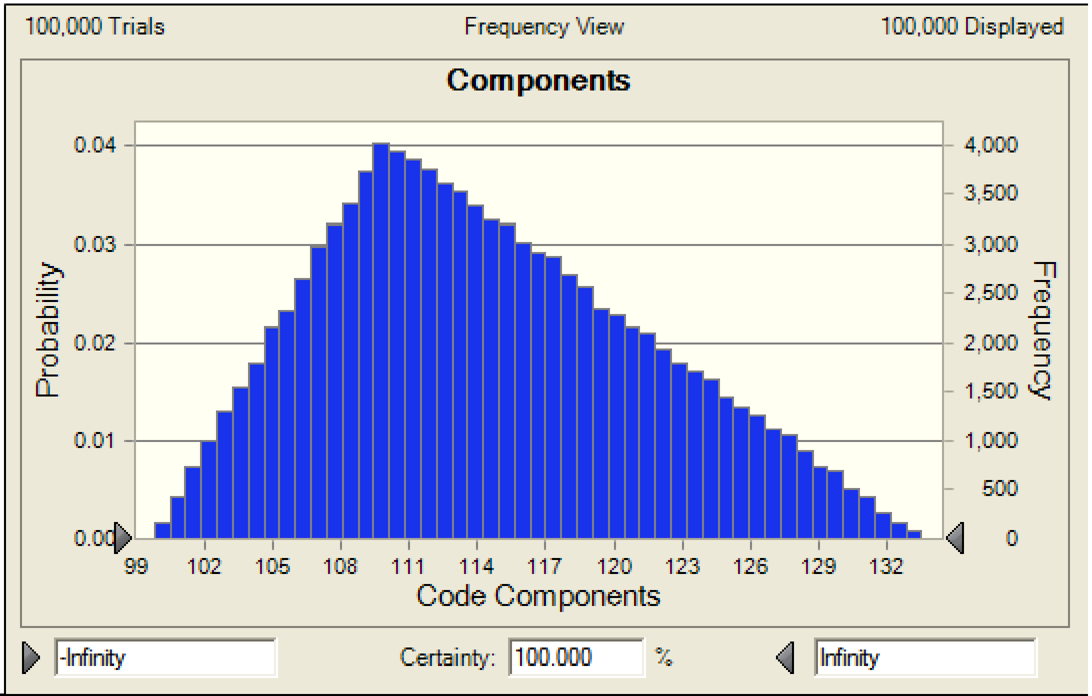
Updating PPM – Post Design PPM

Update PPM with actual values

Rerun predictions

Reevaluate risks and take appropriate actions

- Mitigation plans
- Contingency plans
- CAR/OID actions



Robert Stoddard
Rusty Young

Build PPM

Updates PPM with most recent actuals

Adjusts for changes in

- Staff
- Process definitions
- PDP
- Scope/requirements/design

Produce detailed phase PPMs for detailed monitoring during build phase

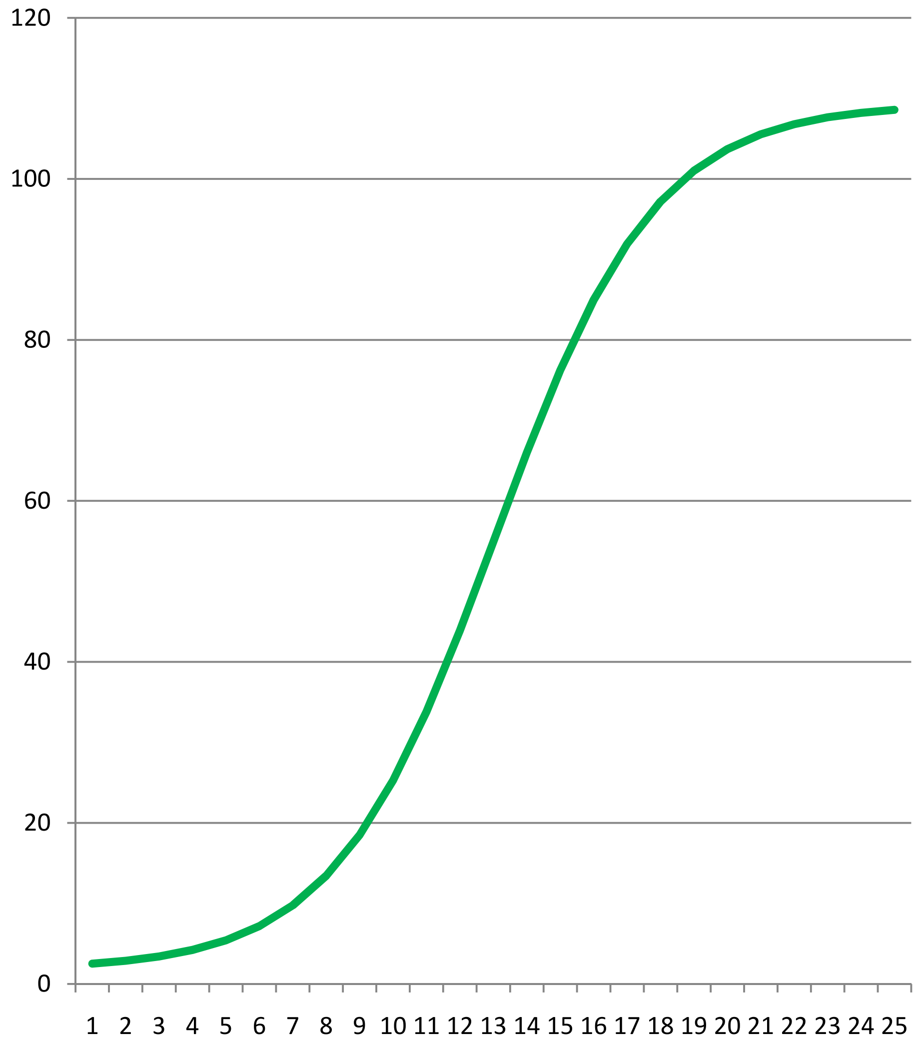
Monitoring the Build

Multi-mode monitoring (not just PPMs)

- SPC – Micro level performance
- PPM – Macro level performance
- EVMS – Cost/Schedule implications
- Schedule – Critical path/Dependency effects

Each monitors different aspects of the product build (similar strategy/tactics can be used during any phase or for maintenance

Build – Component Buildup – Predicted

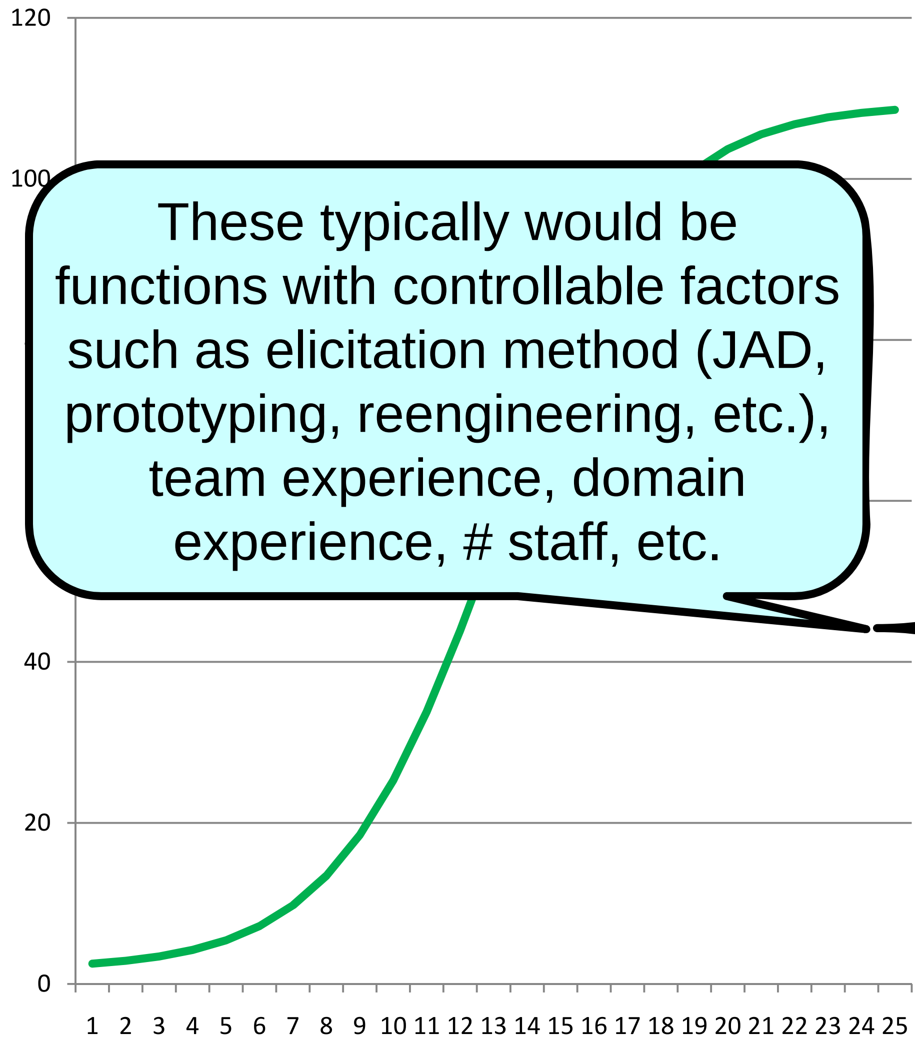


Sigmoid Curve

Y = L+((U–L)/(1+exp[–1*((x–M)/W)])

- **x** time
- **L** = lower asymptote
- **U** = upper asymptote
- **M** = Middle point of growth
- **W** = width from leaves bottom and gets to top

Build – Component Buildup – Predicted



These typically would be functions with controllable factors such as elicitation method (JAD, prototyping, reengineering, etc.), team experience, domain experience, # staff, etc.

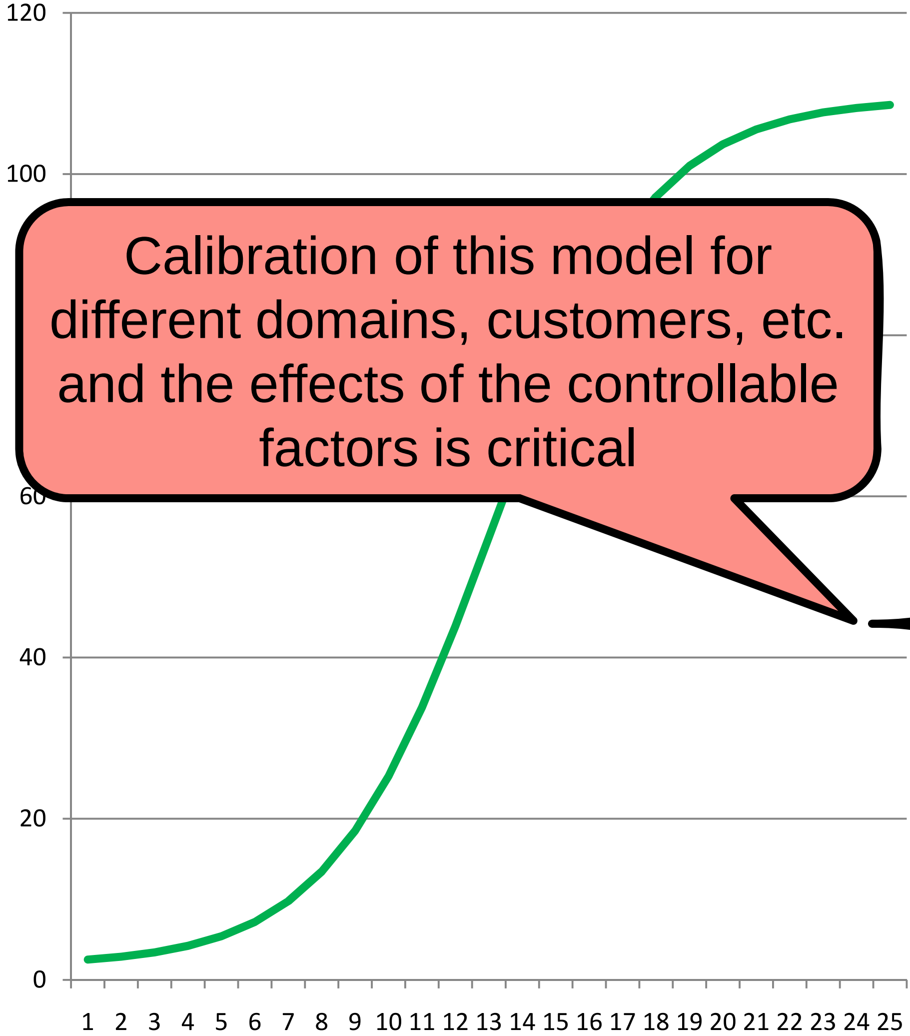
Sigmoid Curve

Y = L+((U–L)/(1+exp[–1*((x–M)/W)])

- **x** time
- **L** = lower asymptote
- **U** = upper asymptote
- **M** = Middle point of growth
- **W** = width from leaves bottom and gets to top

Robert Stoddard
Rusty Young

Build – Component Buildup – Predicted



Calibration of this model for different domains, customers, etc. and the effects of the controllable factors is critical

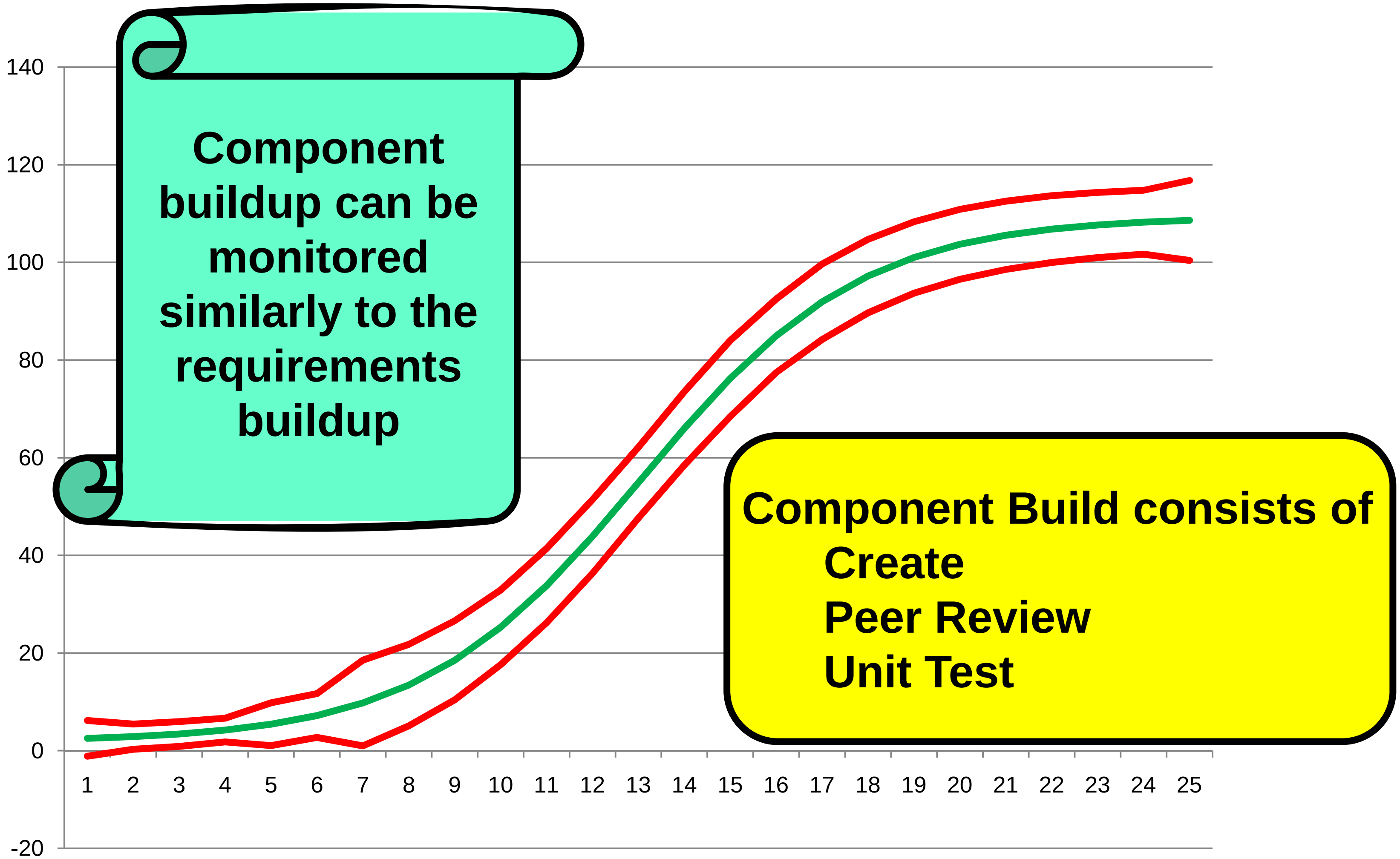
Sigmoid Curve

Y = L+((U–L)/(1+exp[–1*((x–M)/W)])

- **x** time
- **L** = lower asymptote
- **U** = upper asymptote
- **M** = Middle point of growth
- **W** = width from leaves bottom and gets to top

Robert Stoddard
Rusty Young

Build – Component Buildup – Example



Monitor Component Build – SPC

Components designed to be homogeneous or segment into homogeneous subsets (note, may result in an S curve for each sub-group)

Stabilize the build process

- Control chart
  - Size
  - Complexity
  - Effort
  - Duration
- May also chose to stabilize
  - Unit test
  - Build peer review

May also perform CAR/OID activies to optimize process

- May cause a modification to the PPM

Monitor Component Build – EVMS

EVMS, PPMs, and SPC are not incompatible

- Each manages different aspects
- Although there is some overlap

Is value earned at

- End of Build
- End of Unit test
- End of Peer review
- Or a percent at each step

When using EVMS, may want to emphasize monitoring

- Cost variance
- CPI

To monitor quality aspects, PPMs can supplement EVMS

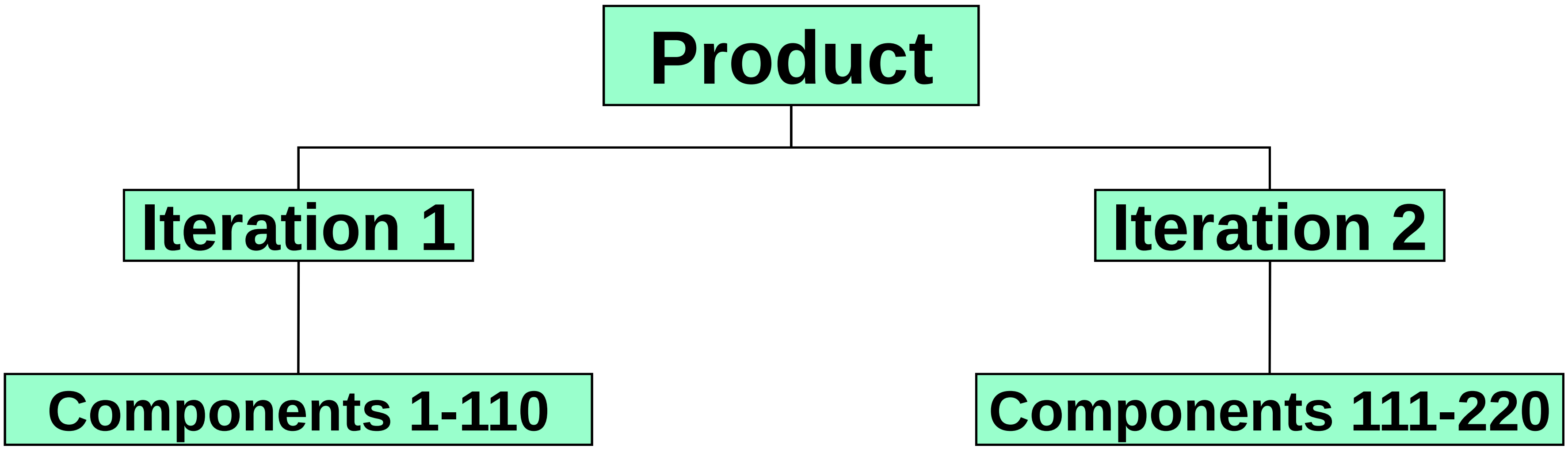
Monitor Component Build – Schedule

Don’t forget the critical path

The models and EVMS may indicate no problem

- But the slippage of a single small task on the critical path can be a source of project trouble
- Sometimes the issue will be picked up as a special cause on a control chart, but the CPM implications are unseen and ignored, thus not detecting that the project may be in trouble

The Model – 2 Partitions, Iterative Build



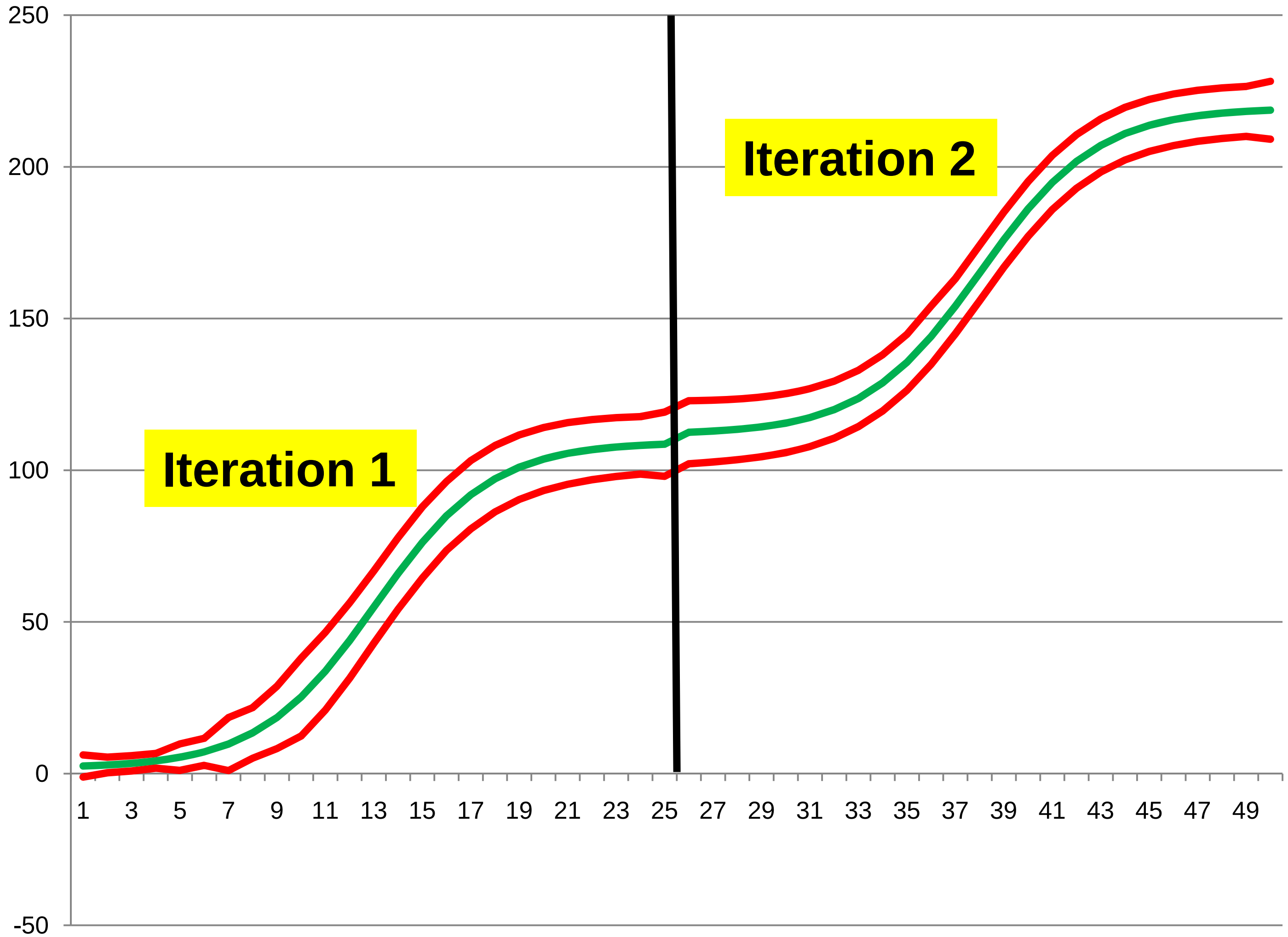
There is support for iterative forms of development such as iterative builds and agile methods

- Use sets of linked sigmoid curves
- Refactoring models

Points to the use of multiple linked models

- Don’t have to stick to a single equation

Iterative Build – Component Buildup – Example



An Example Process Performance Model (PPM) as an input to System Testing

The Situation in Systems Testing

The System Test team receives baselines from various feature teams during the series of system test baselines within each of the product incremental baselines

The Systems Test team does not have the schedule nor the resources to conduct 100% coverage of all possible test cases

<u>**OUR NEED**</u>: A PPM used by each feature team during the handoff of a feature baseline to System Test, whereby the feature team will predict the relative likelihood of a list of defect types.   This will enable prioritized, efficient testing!

Details of the System Testing PPM

The outcome, Y, is the relative likelihood of occurrence of the different standard defect types (e.g. nominal categories such as: logical, data, and algorithmic)

The x factor used in this prediction example is a measure of staff turnover of the feature development team prior to System Test (e.g. continuous data as a percentage)

This x factor was chosen because it historically surfaced as a significant factor in explaining types of defects found in System Test.

Background Information on the Data

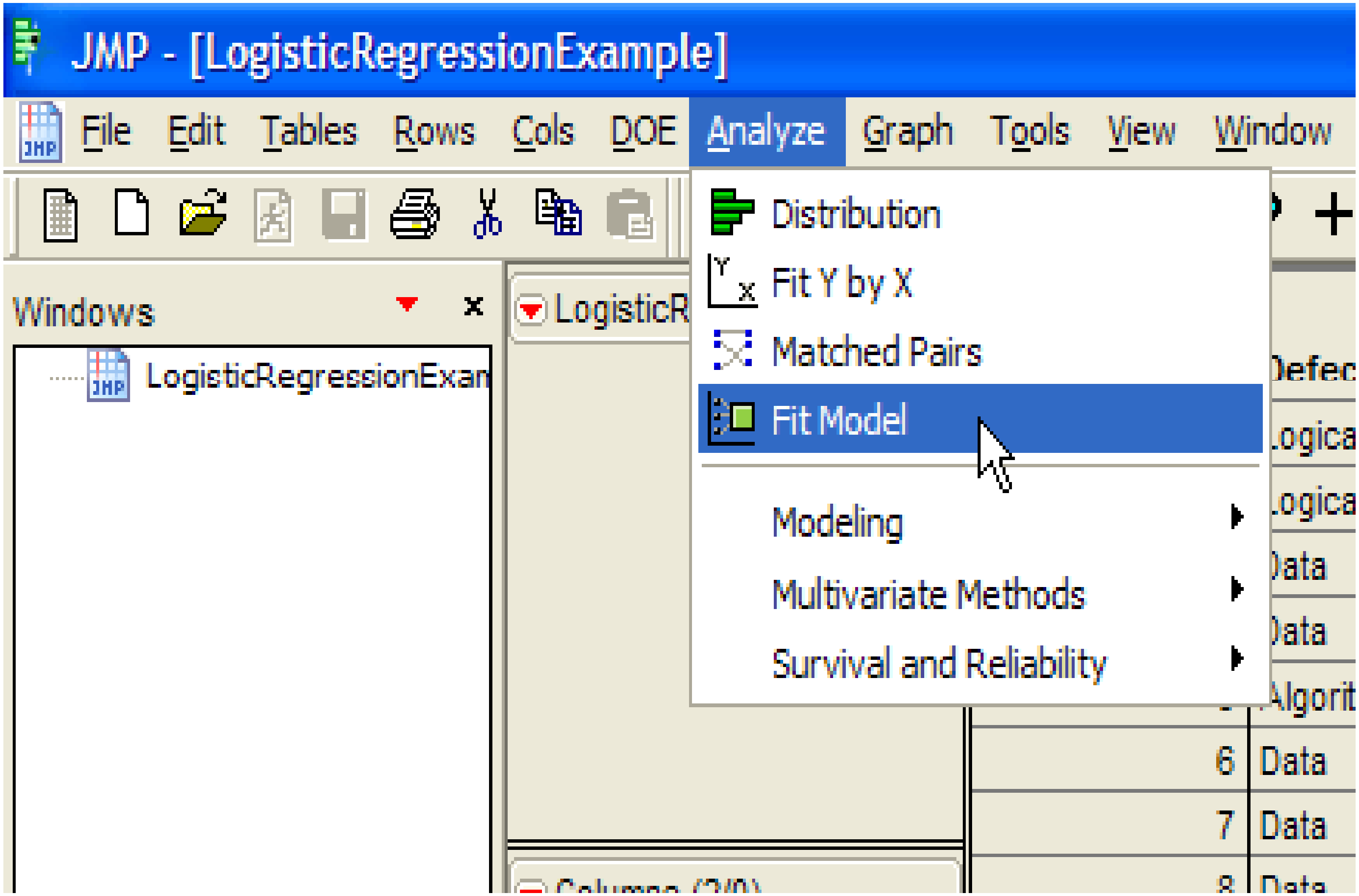
We collected historical data (of the defect type and the staff turnover of the responsible feature team) for a large volume of defects found in System Test

Operational definitions and training were conducted to ensure consistency and repeatability of defect types found in System Test, as well as, the staff turnover rates for feature teams reaching System Test

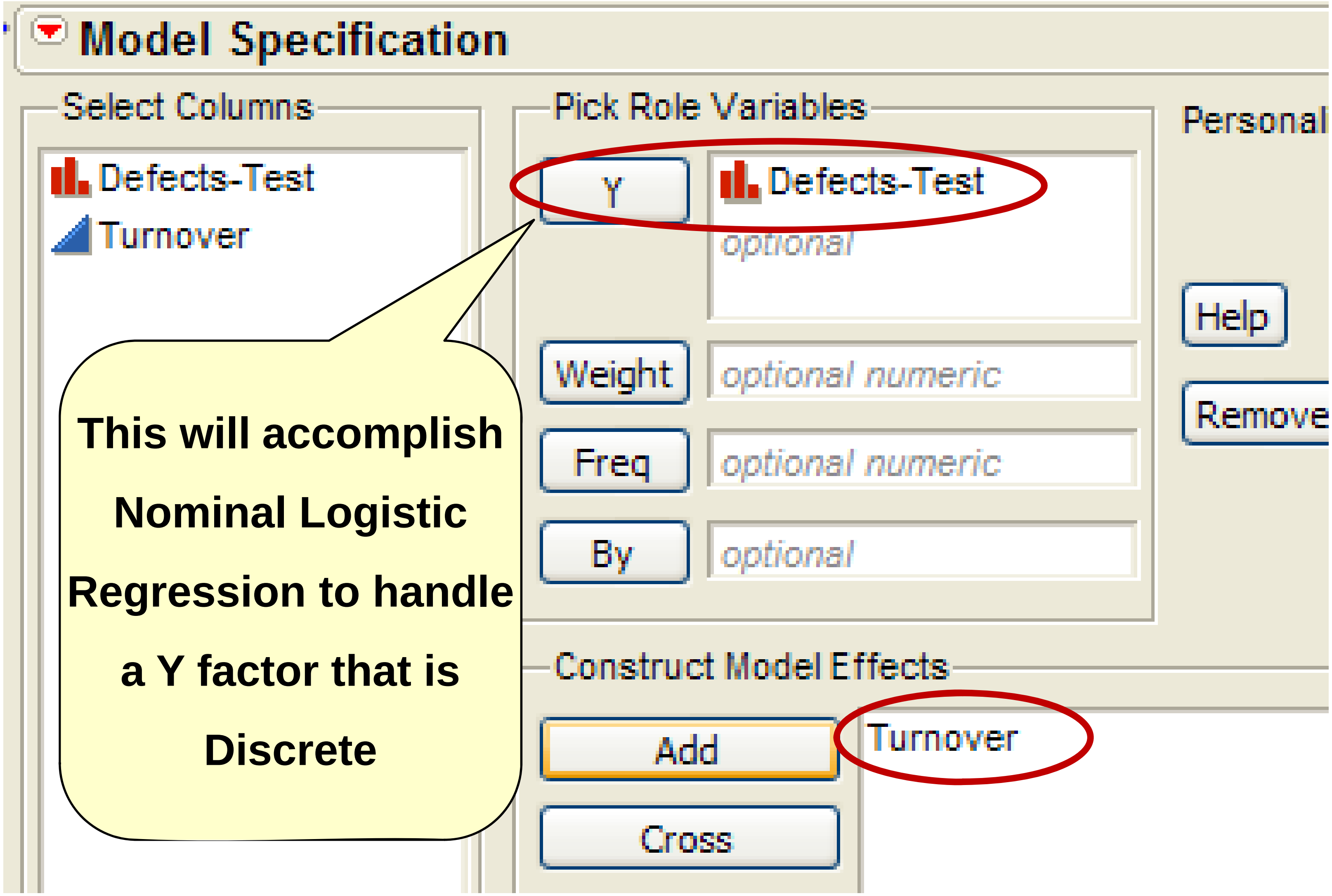
Development of the System Test PPM - 1

	Defects-Test	Turnover
1	Logical	19
2	Logical	23
3	Data	11
4	Data	16
5	Algorithmic	9
6	Data	18
7	Data	11
8	Data	14
9	Logical	28
10	Data	9
11	Data	11

Development of the System Test PPM - 2



Development of the System Test PPM - 3



Development of the System Test PPM - 4

Whole Model Test

Model	-LogLikelihood	DF	ChiSquare	Prob>ChiSq
Difference	8.925626	2	17.85125	0.0001*
Full	56.245396			
Reduced	65.171022			

RSquare (U)	0.1370
Observations (or Sum Wgts)	61

Converged by Gradient

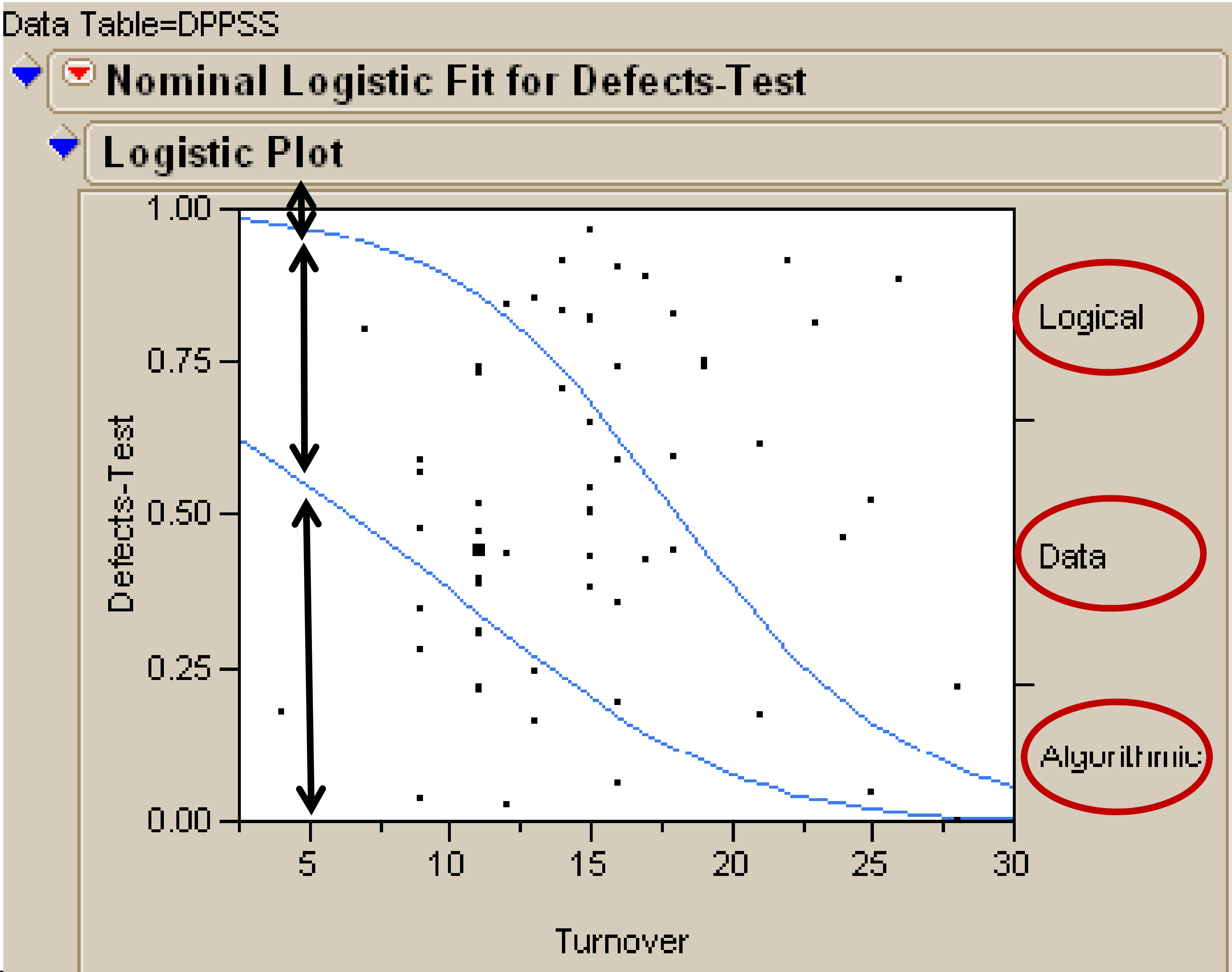
Lack Of Fit

Source	DF	-LogLikelihood	ChiSquare
Lack Of Fit	34	18.169720	36.33944
Saturated	36	38.075676	Prob>ChiSq
Fitted	2	56.245396	0.3602

Parameter Estimates

Term	Estimate	Std Error	ChiSquare	Prob>ChiSq
Intercept	4.49519702	1.5437487	8.48	0.0036*
Turnover	-0.3296347	0.1064655	9.59	0.0020*
Intercept	3.68084598	1.239601	8.82	0.0030*
Turnover	-0.218423	0.076182	8.22	0.0041*

Development of the System Test PPM - 5



Intended Use of this System Test PPM - 1

Once we decide on the final form of the PPM, we will use it in two primary ways:

1) As each feature team proceeds through development, the team will anticipate the final staff turnover rate at the point of System Test based on turnover to-date, and then evaluate the PPM at these updated values to predict the likelihood of the types of defects. If this prediction is unacceptable, they will take immediate action to address the staff turnover rate while still possible.

Intended Use of this System Test PPM - 2

Once we decide on the final form of the PPM, we will use it in two primary ways:

2)System Test will use the prediction to decide if the feature is suitable to enter System Test.   If not, the feature team may have to take risk reduction actions.

Updating this System Test PPM

As more features are tested and defects are found, they will continue to record the type of defects found and the corresponding information of the staff turnover of the responsible feature team

When a group of defects from System Test have been recorded, the organization may choose to add this additional data to the existing data set and then repeat the exercise of developing the nominal logistic regression.

Ultimately, the organization may pursue inclusion of additional x factors that are believed to explain the defect types surfacing in System Test.

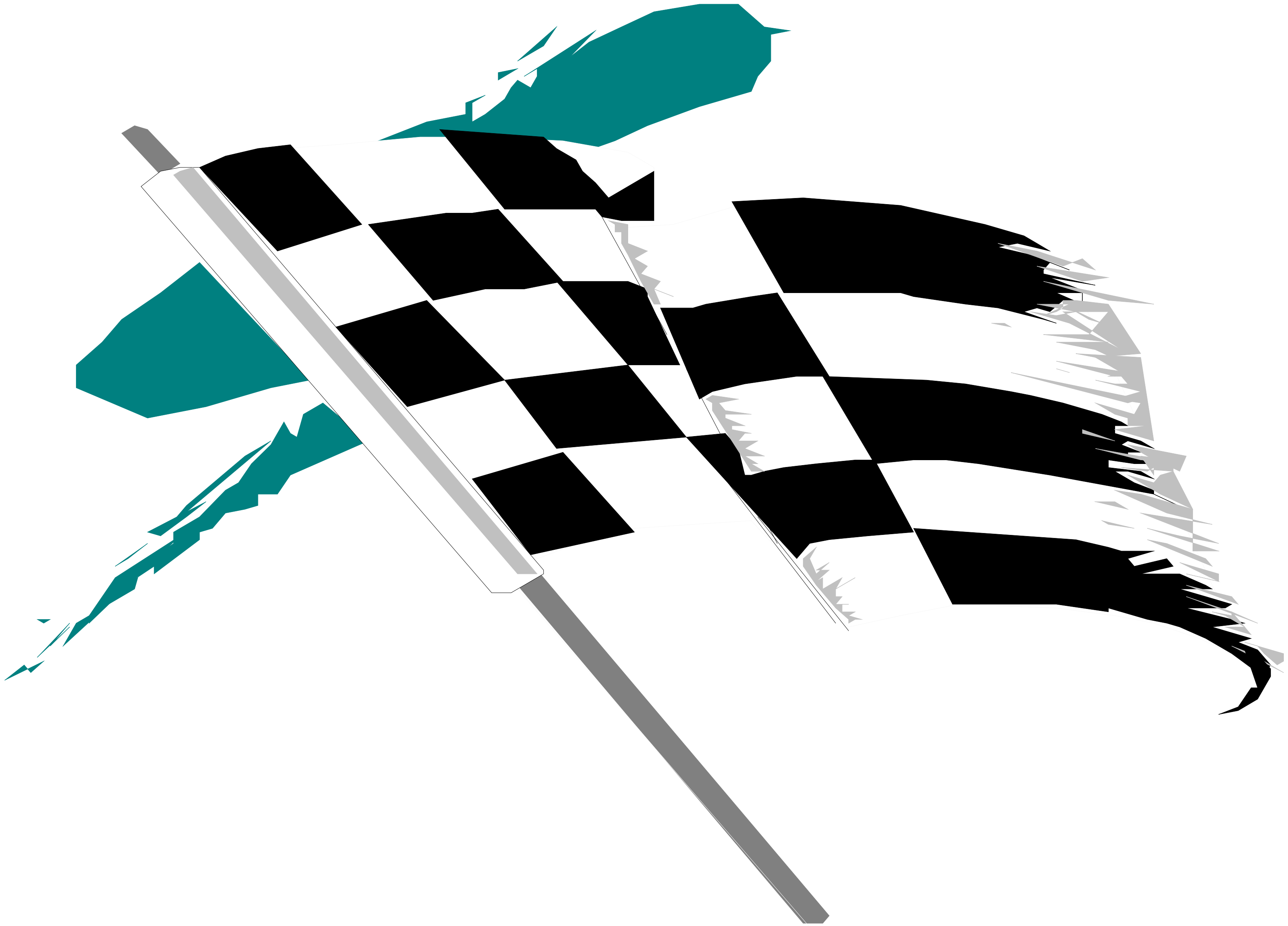
Tutorial Summary

We have shown practical examples of process performance models used in a variety of ways across the lifecycle

The methods depicted are not rocket science and may be performed by practitioners, without becoming statisticians

The greatest challenge in implementing process performance models is not the quantitative or statistical science, but rather…

It is the domain expertise to decide what are the business-important outcomes worthy of prediction, and what are the controllable, sub-process factors likely to prove significant in predicting that outcome



Contact Information

Robert W. Stoddard
Telephone: +1 412-268-1121
Email: rws@sei.cmu.edu

Rawdon Young
Telephone: +1 412-268-2584
Email: rry@sei.cmu.edu

U.S. mail:
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

World Wide Web:
www.sei.cmu.edu
www.sei.cmu.edu/contact.html
www.sei.cmu.edu/sema/presentations.html
To get a pdf and data files related to this presentation

Customer Relations
Email: customer-relations@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257