

Softrel, LLC
20 Towne Drive #130
Bluffton, SC 29910
<http://www.softrel.com>
321-514-4659
1/13/15

SOFTWARE RELIABILITY ASSESSMENT

About Ann Marie Neufelder

- Chairperson of the IEEE 1633 Recommended Practices for Reliable Software working group
- Since 1983 has been a software engineer or manager for DoD and commercial software applications
- Co-Authored first DoD Guidebook on software reliability
- Developed NASAs webinar on Software FMEA and FTA
- Has trained every NASA center and every major defense contractor on software reliability
- Has patent pending for model to predict software defect density
- Has conducted software reliability predictions for numerous military and commercial vehicles and aircraft, space systems, medical devices and equipment, semiconductors, commercial electronics.


Two types of software reliability models exist in industry

○ Predictive models are

- Used before testing
- Supports sensitivity analysis
- Used for determining if and how objective can be met early in process
- Accuracy depends on level of detail of assessment and accuracy of size estimate

○ Estimative models are

- Used during any post integration test
- Extrapolates defect rate during testing into the future
- Determines testing hours needed to reach a reliability objective
- Accuracy depends on test coverage



This presentation focuses on the predictive models

Based on assessment survey and size estimates

Based on extrapolating current defect rate into the future

Software reliability assessment

- A prerequisite for predicting failure rate, MTBF, etc. and sensitivity analysis
- Assessment is a “survey” which predicts fielded defect density
 - Fielded defects/Effective Source Lines of Code
 - Predicted defect density is also used to benchmark the project against similar industries and predict “distressed”, “mediocre” and “successful” in terms of project delivery.
- Survey was developed by 20+ years of correlating empirical data from 689 development characteristics to actual fielded defect density from 100+ projects
- Assessment can range from simple to detailed depending on how much time and money your organization is willing to spend
- More detailed means more accuracy which also allows for a more detailed sensitivity analysis
- See attached spreadsheet for a summary of assessment questions
- Some questions pertain to the entire organization (common factors)
- Some questions pertain to the particular configuration item



Microsoft Office
Excel 2007 Workbook

Assessment areas of interest

Assessment area	Description
Avoiding the “Big blob”	“Big Blobs” are the opposite of small incremental deliveries.
Ability to test code from white box perspective	Most organizations cover the requirements. Few cover the lines of code. A common weakness in even the world class organizations.
Ability to define the off nominal behavior or “Shall nots”	Most organizations develop and test requirements for the nominal conditions. Few can do this for the off nominal conditions.
Change management	World wide - more software failures have been caused by this than anything else.
Visualization	A picture says 1000 words. Yet, few organizations use pictures to support requirements or design.
Domain expertise	Contrary to popular belief – number of years with application type (medical, etc) is more important than number of years of language.
Execution	Ability to start the project on time and manage it until completion
Defect tracking	The ability to track defects is oddly associated with having fewer of them
Planning	Ability to see risks in the project before they derail the project
Process	Ability to develop as per standards
Defect reduction	Ability to remove defects before you get to systems testing
Requirements, design, code, test	Techniques for doing these things more effectiveness
Inherent risks	These are things that can’t be changed but can impact defect density

Software reliability assessment survey

Assessment Survey has 469 questions related to

Question category

Application type - Inherent risks of product and release

Project specific practices –

Personnel and domain experience

Project execution and management, Ability to schedule project and manage progress against schedule.

Development techniques related to change control, requirements, design, code, unit test, integration test, software system test, corrective action, process, metrics, tools, risk management.

Software standards and processes for every phase of and development activity

Some questions are associated with world class organizations while others are associated with distressed projects. The absence of either extreme predicts the “average” project.

Basic survey assessment for each configuration item

Question category	# questions	Resources needed	Artifacts required
<p>Project specific practices – Personnel and domain experience Project execution and management, Ability to schedule project and manage progress against schedule. Development techniques related to change control, requirements, design, code, unit test, integration test, software system test, corrective action, process, metrics, tools, risk management.</p>	<p>83 Plus 14 informational questions</p>	<p>At least three people within the software configuration item team are surveyed:</p> <ul style="list-style-type: none"> •Software lead •Typical software engineer •Software test engineer <p>Physical evidence of all yes answers are required</p> <p>Survey takes about 2 hours per person.</p>	<p>Software development plan, schedules, Work samples such as SRS, SDS, STP from either current project or recent past version.</p>

Each configuration item undergoes this assessment

Basic survey assessment for entire product

Question category	# questions	Resources needed	Artifacts required
Application type - Inherent risks of product and release	23	A software manager or director	
Software standards regarding every phase and software activity	14	Any persons familiar with the software standards and processes that apply to all configuration items	Any software standards used by all configuration items

This is called the “Common” assessment in proposal. It is completed only once and used for all configuration items.

Detailed artifact review

Question category	# Questions	Resources needed	Artifacts required
Project specific practices – Software development guidance and standards related to change control, requirements, design, code, unit test, integration test, software system test, corrective action, process, metrics, tools, risk management.	108	Review is performed by Ann Marie Neufelder	Software process documents
Oversights and gaps in the Software requirements, design, code and test plan	155	Review is performed by Ann Marie Neufelder	Example SRS, SDS, STP, code
Mediocre practices	82	Review is performed by Ann Marie Neufelder	All of the above

The detailed artifact review is optional. You can choose to have a detailed review of the software standards and processes. And/Or you can have a detailed review of the configuration items specific development artifacts.

The mediocre practices are those practices that are popular yet not really correlated to reduced defect density. These are identified for purposes of sensitivity analysis.

Common detailed artifact review – 132 questions are answered for entire product

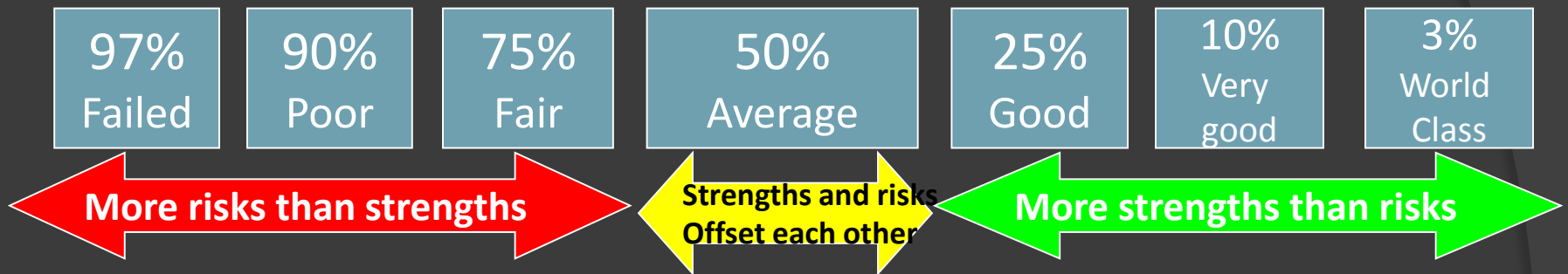
Results of detailed artifact review

- Results of the assessment review include a complete list of items in each artifact that should be present but are not
- These are just a few examples of what is reviewed
 - Do the requirements really trace to the other artifacts such as design, test plans, etc.? Do use cases map to the requirements?
 - State, logic and timing diagrams are used when applicable
 - Interface diagrams and tables are used when applicable
 - There are White box test cases and they map to the design
 - There are Black box test cases and they map to the requirements
 - The design and code have exception handling
 - Identification of the “shall nots” in the requirements and design
 - Written versus unwritten assumptions
 - Do the artifacts have content appropriate for the type of software being developed?
 - Is the level of detail sufficient?
 - Do the artifacts match the interview survey responses?

Regardless of whether you choose the initial assessment or the detailed assessment, the result is one of 7 groups

More fielded defects

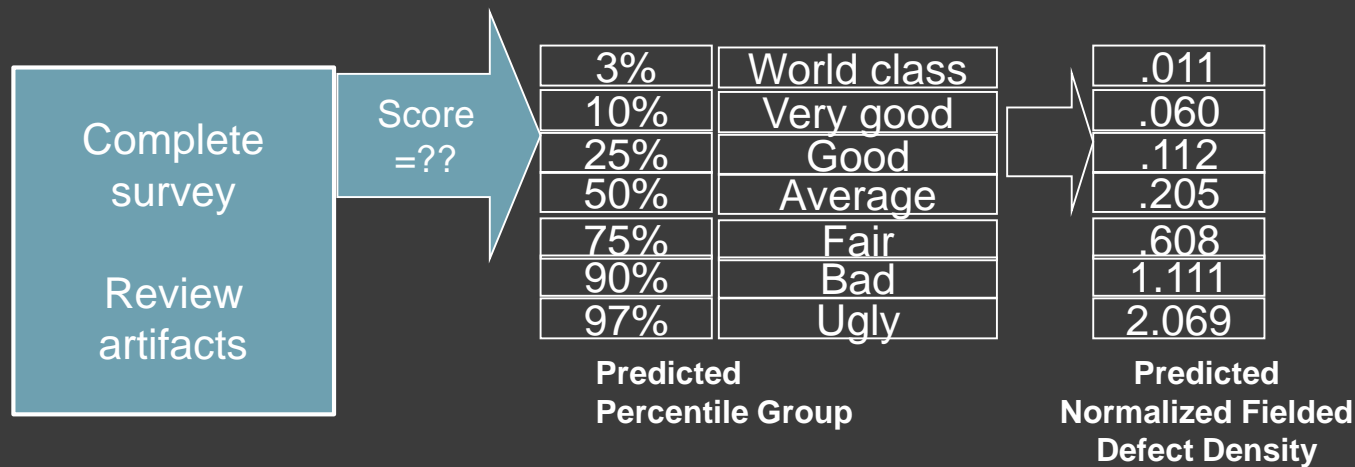
Fewer fielded defects



- Percentile group predictions...

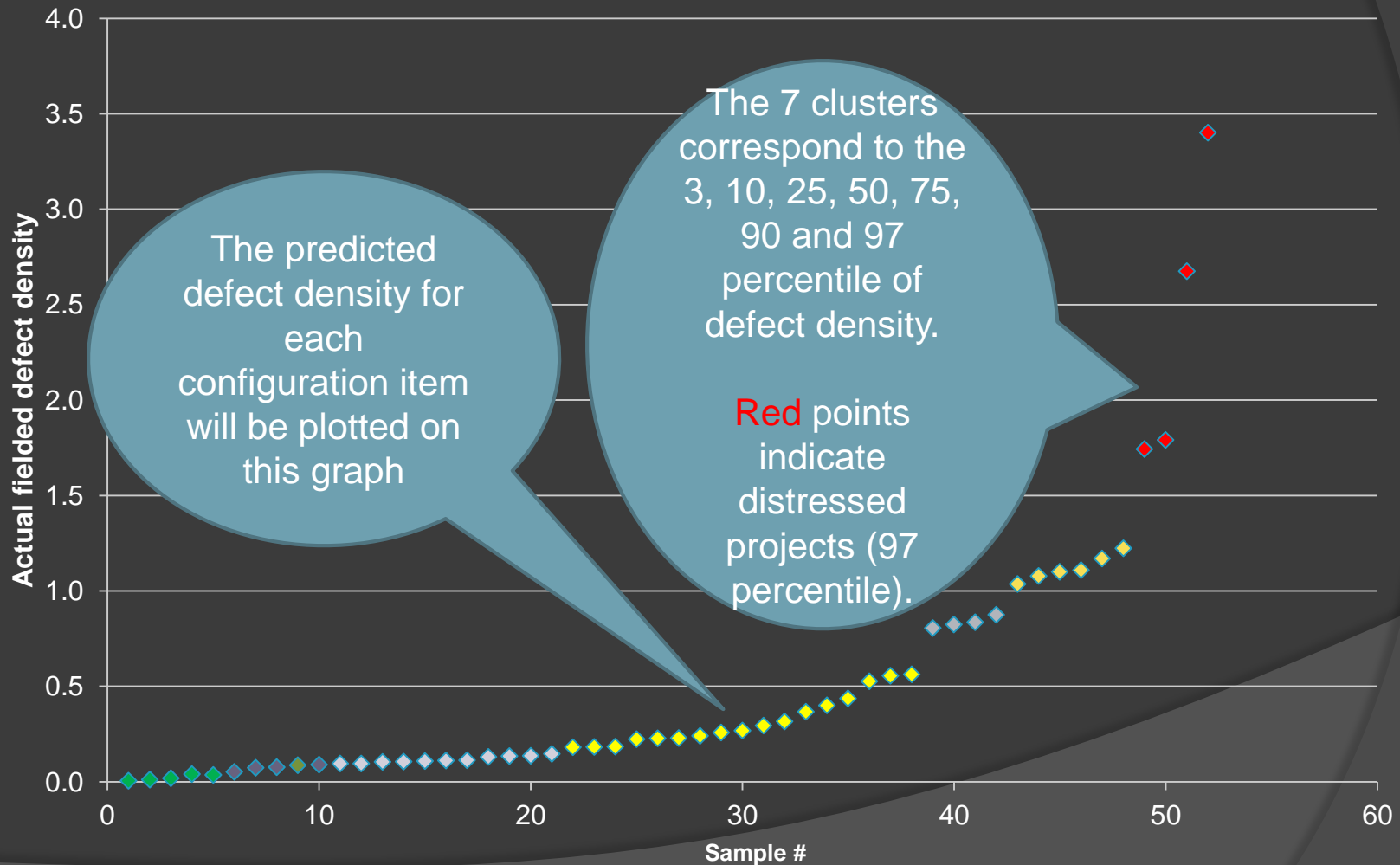
- Pertain to a specific product release
- Based on the number of risks and strengths
- Can only change if or when risks or strengths change
- Some risks/strengths are temporary; others can't be changed at all
- Can transition in the wrong direction on same product if
 - New risks/obstacles added
 - Strengths are abandoned
- World class does not mean defect free.** It simply means better than the defect density ranges in database.

The assessment score predicts defect density



1. The survey is scored.
2. The score is used to determine one of seven percentile groups.
3. That percentile group then predicts defect density.
4. The prerequisite tree is created as shown next by graphing the affirmative and negative responses in each category
5. The defect density can then be multiplied by effective size to yield a prediction of testing and fielded defects.
6. The detailed assessment usually means that prediction is more accurate than with initial assessment.

These are the actual fielded defect densities for the projects in SoftRel Database.



On time delivery statistics for each defect density percentile group

- The projects in the bottom half were 3 times more likely on average to be late than the projects in the top half
- More importantly, when the projects in the bottom half were late, the magnitude of lateness as a percentage of the original schedule is 4 times bigger than the projects in the top half
- *Of all parameters related to software reliability – the ability to stay on schedule is the only parameter that has a nonlinear relationship to defects.*

Group Name	Percentile Group	Late deliveries	
		% Projects that are late	When late, how late as % of schedule
World class	3%	40	12
Very Good	10%	20	25
Good	25%	17	25
Average	50%	34	37
Fair	75%	85	125
Impaired	90%	67	67
Distressed	97%	83	75

A “prerequisite tree” will be generated for each configuration item that is assessed.

All of the survey questions are related to at least one of the items on this tree.

Key practices are cumulative
Practices at the top should not be attempted as long as there are gaps at the bottom.

None of the world class organizations skipped the practices at the bottom or middle

Red – gaps

Yellow – needs refinement

Green – assumed strength
(may be optimistic)

- Formalize white box unit testing with non-peer review
- Measure line or branch coverage
- Define abnormal behavior and shall nots
- Enforce developer white box unit testing
- Employ visualization
- Avoid big blobs - code a little, test a little
- Plan ahead
- Maintain/Increase domain expertise
- Write test plans before code written
- Get all parts of lifecycle in place from requirements to support
- Review and prioritize changes
- Track and record all defects and changes (including field reports)
- Get control of changes and versions
- Get a comprehensive test plan (versus ad hoc testing)
- Independently test every change
- Avoid obsolescence

Group
World Class
Very Good
Good
Average
Fair
Impaired
Failed

Software Reliability Assessment

Benchmarking

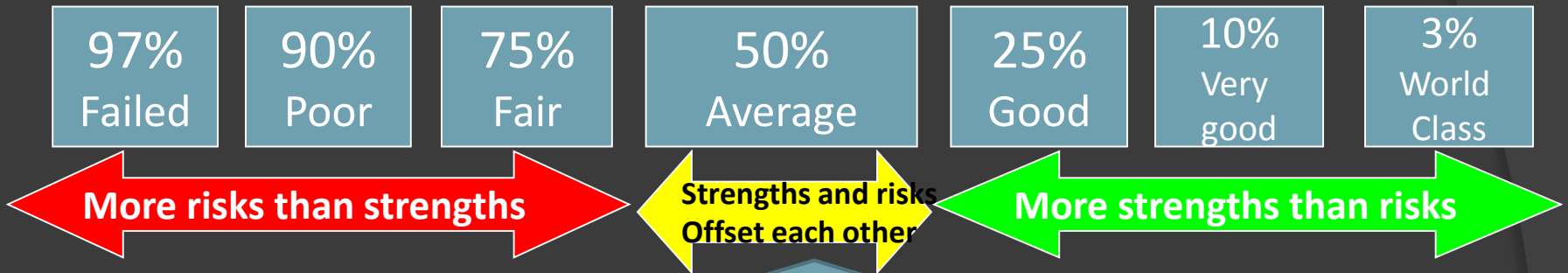
Benchmark against similar applications

- ◎ Once the software reliability assessment is complete, the average defect density of applications similar to yours will be presented
 - Similar industry
 - Similar application type (i.e. firmware, Client/Server software, UI software, etc.)
 - Similar size
 - Similar inherent product and technology risks (i.e. brand new product versus legacy)

Example of benchmarking results

More fielded defects

Fewer fielded defects



Risk based prediction – 23 questions

Initial survey based prediction – 95 questions)

Prerequisite tree sanity check

Detailed survey based prediction 300+ questions

Historical range for projects with similar risks

Historical range for application type

Historical range for your industry

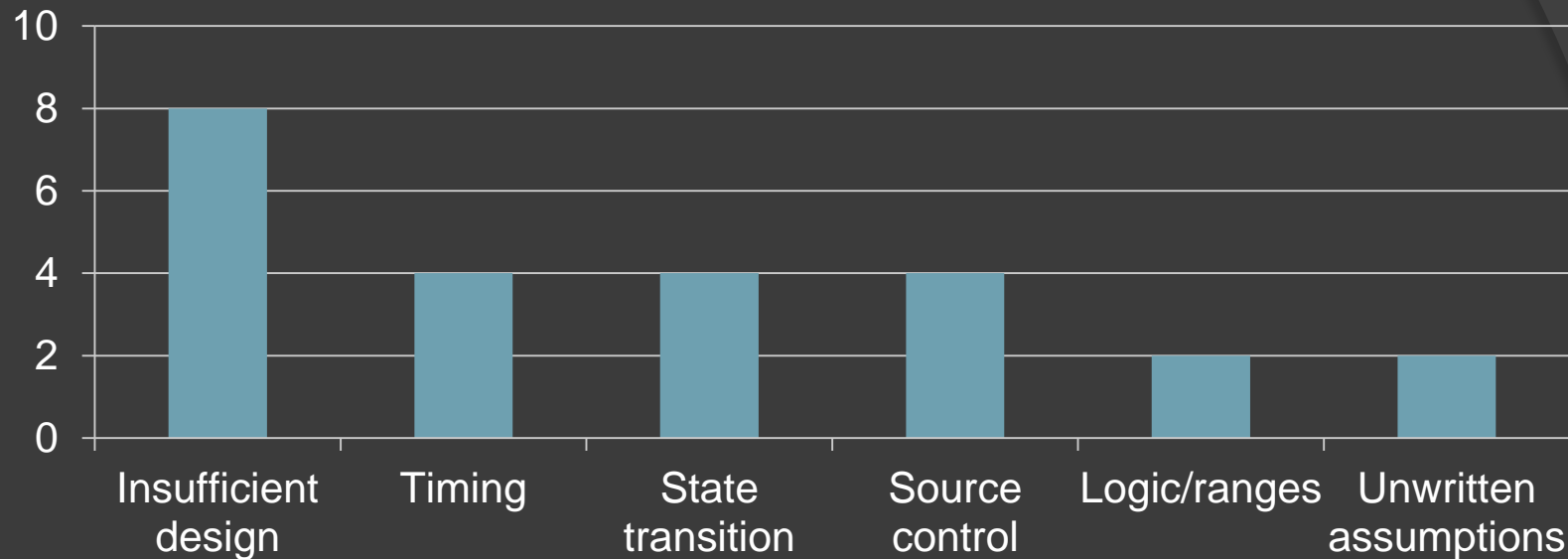
Software Reliability Assessment

Optional defect root cause analysis

Software defect root cause analysis

- Software reliability assessment identified certain gaps and strengths which can lead to a certain “volume” of defects
- A root cause analysis can confirm the “types” of defects
 - Faulty requirements?
 - Faulty design?
 - Faulty implementation?
 - Faulty interfaces?
 - Faulty changes or corrective actions?
 - Faulty source and version control?
- These can and will be unique for each configuration item even if they have the same development processes
- A minimum of 50 past software defects (found during testing or operation) are analyzed and classified by root cause
- A Pareto is generated as shown in the following example
- Often times the most common “type” of defect is also associated with the biggest “gaps” on the prerequisite tree.
- In the event that assessment illustrates multiple gaps in development technique, the RCA can be used to prioritize which gaps are addressed

Example of a root cause analysis



Defects are introduced because of either bad requirements, bad design, bad coding practices or bad change control.

- ⦿ Requirements defect – The “whats” are incorrect.
- ⦿ Design defect – The “whats” are correct but the “hows” are not. Most of the above example root causes are design related.
- ⦿ Coding defect- The “whats” and “hows” are correct but the software engineer did not implement one or more lines of code properly.

Software Reliability Assessment

Sensitivity Analysis

Sensitivity analysis within each configuration item

- These are the most sensitive parameters to reliability in order of impact

Parameter	Impact
Ability to meet schedule (and therefore have no unplanned additions to code during testing and clinical tests)	Exponential
Ability to estimate the reliability growth	Exponential
Effective size (whether or not reused code is really reusable)	Linear – Twice the code means twice the failure rate
Defect density	Linear – Twice the defect density means twice the failure rate
<ul style="list-style-type: none">Ability to successfully execute incremental development and test. Ability to measure progress against schedule on daily or weekly basis.	Most sensitive questions on survey
<ul style="list-style-type: none">Ability to test code from both a black and white box testing perspective	Second most sensitive question on survey

Sensitivity of assessment questions

Assessment area	# Questions	Weighting coefficient
Avoiding Big Blobs	13	16
Change Management	7	15
Coding	15	12
Defect Reduction	60	15
Defect Tracking	50	9
Design	1	10
Domain expertise	6	4
Execution	20	16
Inherent risks	90	21
Personnel	5	15
Planning	1	11
Process	1	23
Requirements	1	7
System testing	2	19
Unit testing	1	7
Visualization	20	3

- Not all survey assessment questions have equal impact on predicted defect density
- *While it's tempting to focus on only the practices with high coefficients, many of the practices with low coefficients are required prerequisites for the high coefficient items*
- Your sensitivity analysis will identify a few practices that have the most sensitivity *for which the prerequisites are already in place.*
- The relative cost of that practice will be compared against the predicted reduction in defect density

Example of sensitivity analysis

- In this example, 4 practices have been identified for improvement.
 - Two are related to avoiding big blobs, one is related to change management and one is related to systems testing.

Category	Coefficient	Prediction - number positive responses	What if scenario - number positive responses
Avoiding Big Blobs	13	7	9
Change Management	7	2	3
Coding	15	4	4
Defect Reduction	60	1	1
Defect Tracking	50	3	3
Design	1	2	2
Domain expertise	6	2	2
Execution	20	4	4
Inherent risks	90	1	1
Personnel	5	4	4
Planning	1	3	3
Process	1	4	4
Requirements	1	8	8
System testing	2	14	15
Unit testing	1	6	6
Visualization	20	1	1

Example of sensitivity analysis

Statistics	What if scenario	Original Prediction
Predicted fielded defect density	0.01404	0.02028
Improvement in defect density	30.79%	
Predicted relative increase in cost of the new practices	26%	

The “what if scenario” is predicted based on the 4 improvements.

It's compared to the original prediction and the result is a 30% reduction in defect density

The relative cost of the 4 improvements is then compared to the cost savings of the 30% reduction.

The work for the cost reduction is not shown here. This is predicted based on each of the improvements as a percentage of total yearly budget.

Sensitivity analysis can also identify practices that aren't valuable from SWR perspective

- ⦿ The sensitivity analysis doesn't necessarily focus on *adding* practices...
- ⦿ 82 popular software development practices have been found to have no correlation to fielded defect density
- ⦿ Replacing mediocre practices with effective practices is often the most economical way to reduce defect density
- ⦿ Example: Code reviews that focus on superficial coding standards have been found to have mediocre correlation to defect density. These code reviews can be replaced with code reviews that focus on common faults instead.
- ⦿ Your sensitivity analysis will include a listing of mediocre practices with regards to SWR.

Sensitivity Analysis Within System

- ⦿ Determine % failures due to software from past systems
- ⦿ Compare actual % failures to predicted % failures
- ⦿ Determine tradeoffs within configuration items and between SW and HW
- ⦿ Look for system design issues that impact overall reliability
 - Software configuration items that are too big (too many unrelated functions in one executable)
 - Software configuration items that support more than one hardware configuration item

Software Reliability Assessment

**Use Assessment to Predict
MTBF, failure rate, reliability,
availability, maintenance
staffing**

Predict effective size in KSLOC

- The bigger the software is, the more potential it has to fail
- Effective means *new*
 - *New or modified code = 100% effective*
 - *Reused but not modified = 10% effective*
 - Terms
 - KSLOC = 1000 executable source lines of code
 - EKSLOC = effective KSLOC
 - Normalized EKSLOC has been converted to one base language since different languages have different densities (See code expansion below)
- Below is an example of what's required for each configuration item

Component	New KSLOC	Modified KSLOC	Reused KSLOC	Language	Language conversion	Normalized EKSLOC	+/-
a	425.00	184.00	214.20	Object Oriented	6	3782.5	10%
b	89.00	25.00	571.80	Object Oriented	6	1027.1	10%
c	113.00	43.00	246.00	Object Oriented	6	1083.6	10%
d	19.00	14.00	50.00	Second Generation	3	114	10%
e	4.00	4.00	17.00	Object Oriented	6	58.2	10%
f	1.00	0.00	3.00	Assembler	1	1.3	10%
Total						6066.70	

Predict defects

- A defect is defined as any defect that *ultimately* requires a change to the source code.
 - The prediction model predicts priority 1-4 (catastrophic through nuisance).
 - This prediction will be adjusted in the next step to filter for certain criticalities of failures
- Below is an example output when defect density is multiplied by effective size
 - The upper and lower bounds are determined by the confidence in the size estimate and defect density estimate
 - Predicted defect density = .060 defects/EKSLOC +/- 2.9%
 - Predicted EKSLOC = 6066.7 +/- 10%

	Nominal	Upper bound	Lower bound
Total Defects Inherent in the software (may take years to uncover)	364	410	317

Predict defect profile

- An industry standard exponential formula is used to extrapolate when the defects will occur once the software is fielded
- Defects cannot be fractional, however, the significant digits are maintained for other calculations
- This is an example of a defect profile

Months of reliability growth	Total defects predicted this month
1	42.722
2	37.702
3	33.272
4	29.363
5	25.912
6	22.868
7	20.181
8	17.809
9	15.717
10	13.870
11	12.240
12	10.802
13	9.533
14	8.413
15	7.424
16	6.552
17	5.782
18	5.102
19	4.503
20	3.974
21	3.507
22	3.095
23	2.731
24	2.410

Predict failure rate

- The failure rate is predicted by dividing the predicted defect profile by the predicted duty cycle of the software

Months of reliability growth	Total defects predicted this month	Predicted duty cycle in hours	Predicted failure rate	Predicted MTTF in hours
1	42.722	730	0.058523	17.08722
2	37.702	730	0.051647	19.36237
3	33.272	730	0.045578	21.94037
4	29.363	730	0.040223	24.86122
5	25.912	730	0.035496	28.17228
6	22.868	730	0.031326	31.92234
7	20.181	730	0.027645	36.17264
8	17.809	730	0.024396	40.99051
9	15.717	730	0.02153	46.44652
10	13.870	730	0.019	52.63158
11	12.240	730	0.016767	59.64052
12	10.802	730	0.014797	67.58008
13	9.533	730	0.013059	76.5761
14	8.413	730	0.011525	86.77047
15	7.424	730	0.01017	98.32974
16	6.552	730	0.008975	111.4164
17	5.782	730	0.007921	126.2539
18	5.102	730	0.006989	143.0811
19	4.503	730	0.006168	162.1141
20	3.974	730	0.005444	183.694
21	3.507	730	0.004804	208.1551
22	3.095	730	0.00424	235.8643
23	2.731	730	0.003741	267.3014
24	2.410	730	0.003301	302.9046

Summary

- ◎ Software reliability assessment can be used for
 - Predicting any reliability figure of merit
 - Combining software reliability predictions into the system level model
 - Determining the most effective way to meet a reliability objective
 - Determine factors that are less effective
 - Compare your reliability figure of merit with those from similar industries, application type and risks