Applying Software Defect Prediction Model for reliable product quality

ISSRE 2009

Industry Practice

Lakshminarayanan Devarajan
Alcatel-Lucent

Agenda

- Business Context /Alignment of Goals / Strategic approach
- Problem definition & potential solution
- Empirical Defect Prediction Model framework
- Case study – Applying the technique
- Case study -Process Improvement – Static Analyzer , Scenario based reviews
- Model Strengths
- Results / Limitations / Expectations / Barriers
- Critique of the model
- References & Acronyms
- **Audience**
  - Software engineering groups / Project management group
  - Software Quality group / Process management group
- **Key takeaways**
  - Introduce Empirical Defect Prediction Technique technique & the results
  - Introduce process Improvements – Static Analyzer , Scenario based reviews to find defects early in the cycle

Business context

Telecom software / systems need to have reliable product quality .
The Quality goal is very aggressive

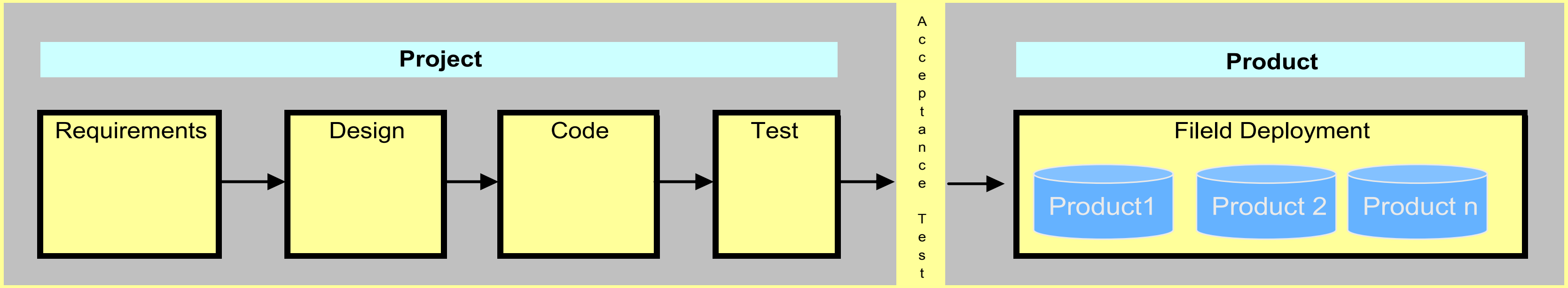
- Customer objectives - Ensure the end user satisfaction to the highest levels
  - Provide a reliable system to the end user
    - Need to meet the Reliability / Availability requirements -99.9xx%
    - Field Defect Density - zero or only very few high severity defects expected

- Organization objectives –Ensure the Customer Satisfaction to the highest levels
    - to be the industry best-in-class TL 9000 [1] Metrics on availabilty
    - Need to be well within the down time requirements on outage frequency / duration limits for all products
    - Need to be well within the Field Defect Density targets for all products

Alignment of Goals

Organisation	
Metric	**Goal**
Outage Downtime	100% within limits for Products 1..n
Field Defect Density	100% within limits for Products 1..n
Customer Satisfaction Index	>4.5 on 5

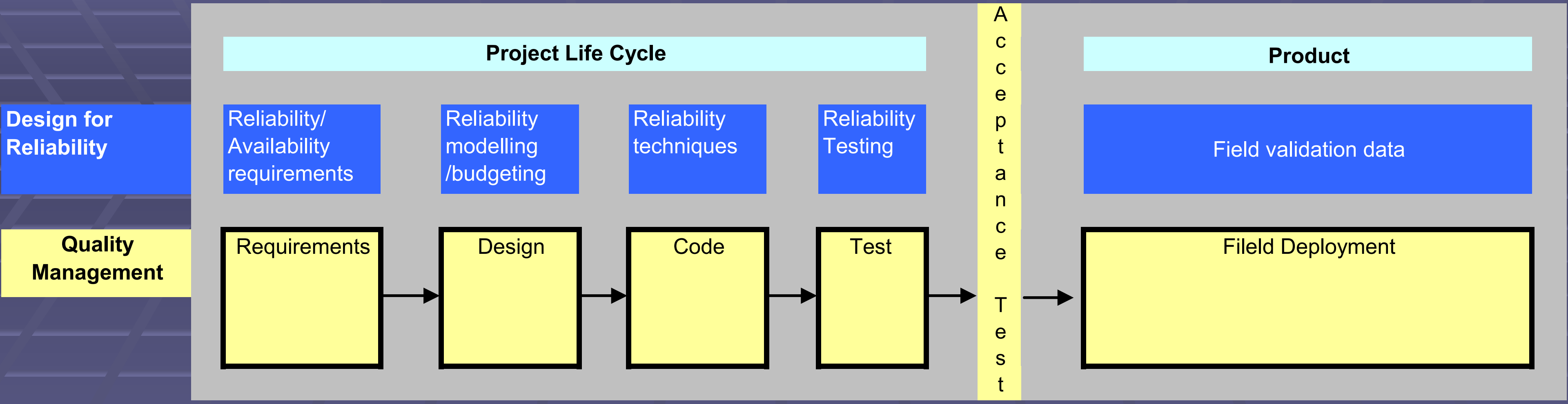
Customer	
Metric	**Goal**
Outage Downtime	< x minutes / Product/year
Field Defect Density	< 0.75 defects / KNCSL



Project	
Metric	**Goal**
Cumulative DRE	>98%

- Alignment
  - The metrics /goals are defined to ensure the Customer / Organization objectives are aligned and met .
  - Field Defect Density - < 0.75 defects / KNCSL .The objective is to minimize the residual defects in the product
  - The Organization metrics are cascaded to the Project metrics
  - Cumulative DRE (Defect Removal Efficiency)goal >98%. DRE goal is set based on the thorough understanding of the defects injected / removed every phase

Strategic approach



Project Life Cycle

Product

Design for Reliability

Reliability/ Availability requirements

Reliability modelling /budgeting

Reliability techniques

Reliability Testing

Field validation data

Quality Management

Requirements

Design

Code

Test

Acceptance Test

Fileld Deployment

- **Outage Downtime - <**x minutes / Product/year
- The Reliability model helps to model , budget & predict reliability The product is architected to meet the Reliability requirements . Need a high Quality stable product base to ensure the reliability
- Quality management is needed to ensure the stable product base is achieved early in the development phase
- Strategic approach -Quality Management & Reliability planning activities are complimentary to each other and assist in building reliable product quality

Problem definition & solution

- To effectively do the Quality Management need a **Software Defect Prediction Model** that can assist in planning , monitoring & control & predict field defect density

- This talk focuses on
  - Quality planning & management using **Empirical Defect Prediction Technique**
  - Case study –
  - Applying the technique
  - Process Improvement – Static Analyser , Scenario based reviews

- Software Defect Prediction Model
  - uses the historical data of the organization
  - uses the In-Process defects(total defects created & removed) to predict the residual defects (defects found by customer)
  - can be applied for new projects

Model Strengths

- A simple model (easy to understand ,create ,use & maintain)
- Wide acceptance of the model within the organization across various departments
- Can start using from the early phases of the development cycle
- Can be used for

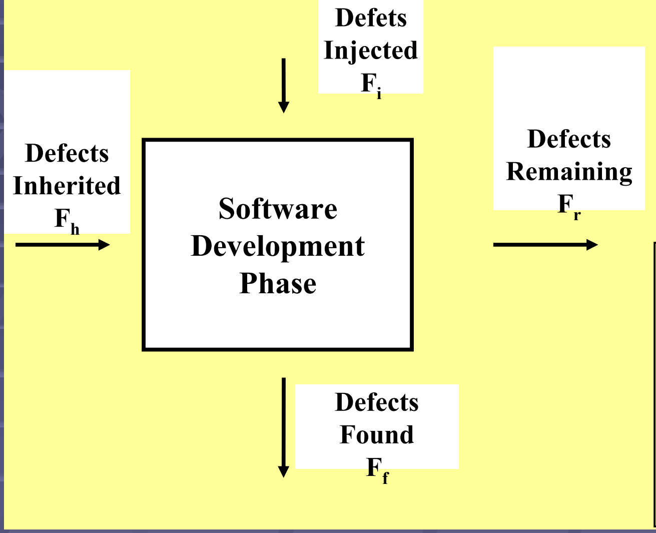
- **Prediction**
  - will the software meet the established quality goals?

- **Quality management**
  - plan & control defect injection and removal activities throughout the development phases

- **Process management / improvement**
  - what Process improvement is needed to meet a given defect density goal ?
  - Plan & monitor process improvements so that process meets customer / business needs.

- **Project management**
  - Is the progress as per Quality plan ?
  - Track progress toward the established goals for delivered software quality
  - are corrective actions needed to meet goals?

Empirical Defect Prediction Model framework


Defets Injected F_i

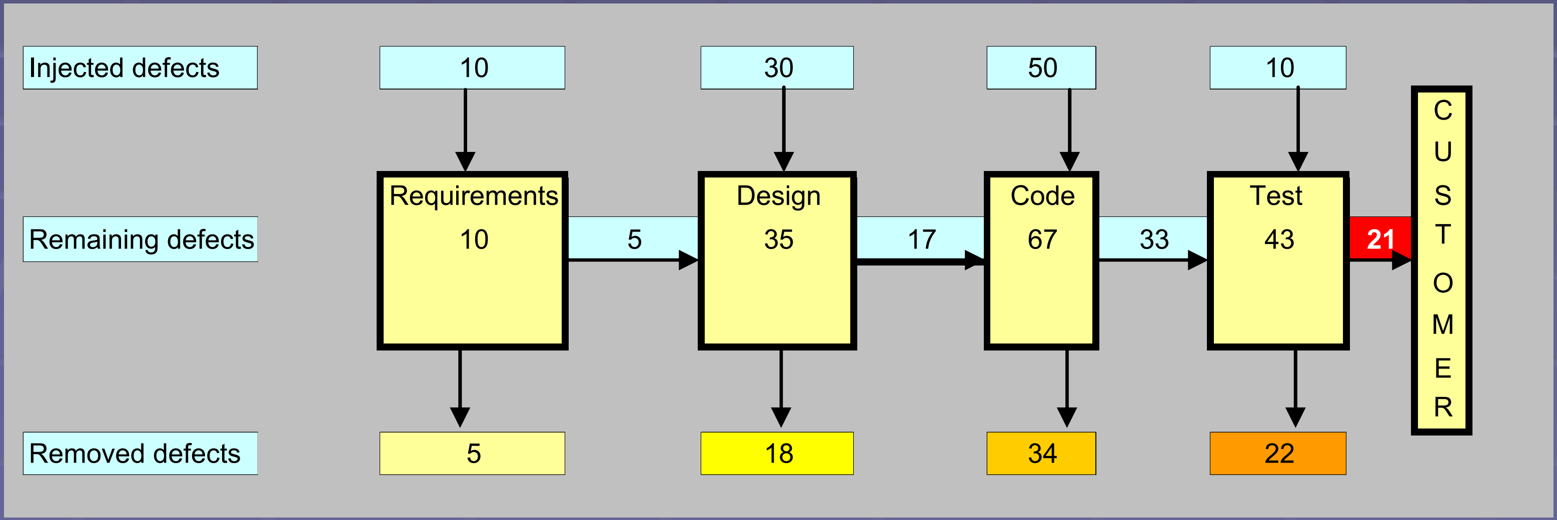
Defects Inherited F_h

Software Development Phase

Defects Remaining F_r

Defects Found F_f

- $D_h + D_i = D_f + D_r$,
- $R = D_f/(D_f + D_r)$, R is the *Defect Removal Efficiency*
- $D_r = F_f(1 - R)/R$


Injected defects

Remaining defects

Removed defects

10

Requirements
10

5

30

Design
35

17

50

Code
67

33

10

Test
43

21

CUSTOMER

5

18

34

22

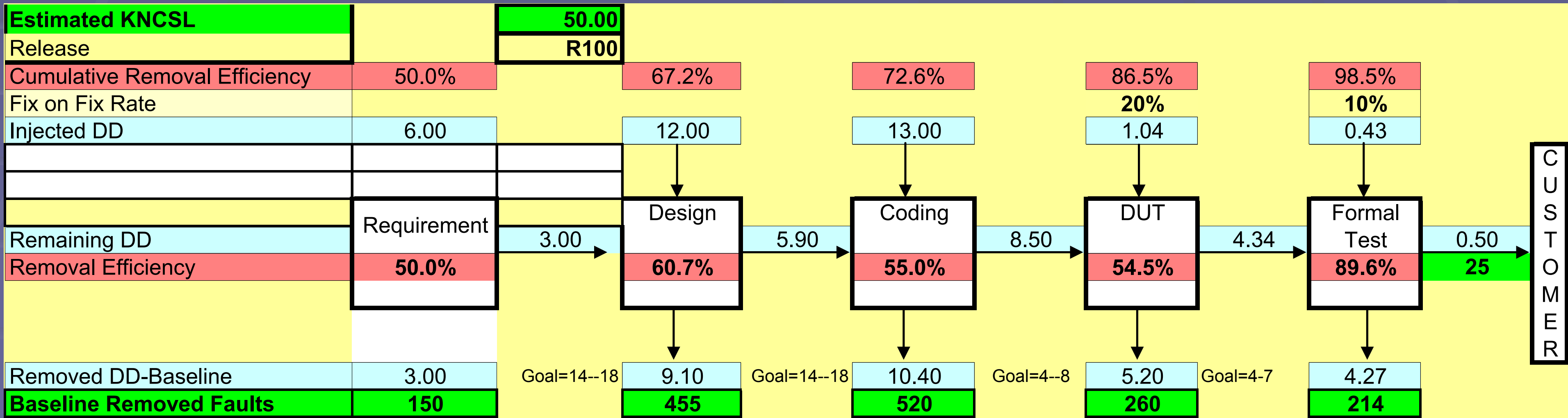
The model above is self explanatory .The example above shows that 21 defects are slipping to the customer (residual defects) . The cumulative defect removal efficiency is 79% .

Case study -Project R100

- Create the Defect profile for a new project

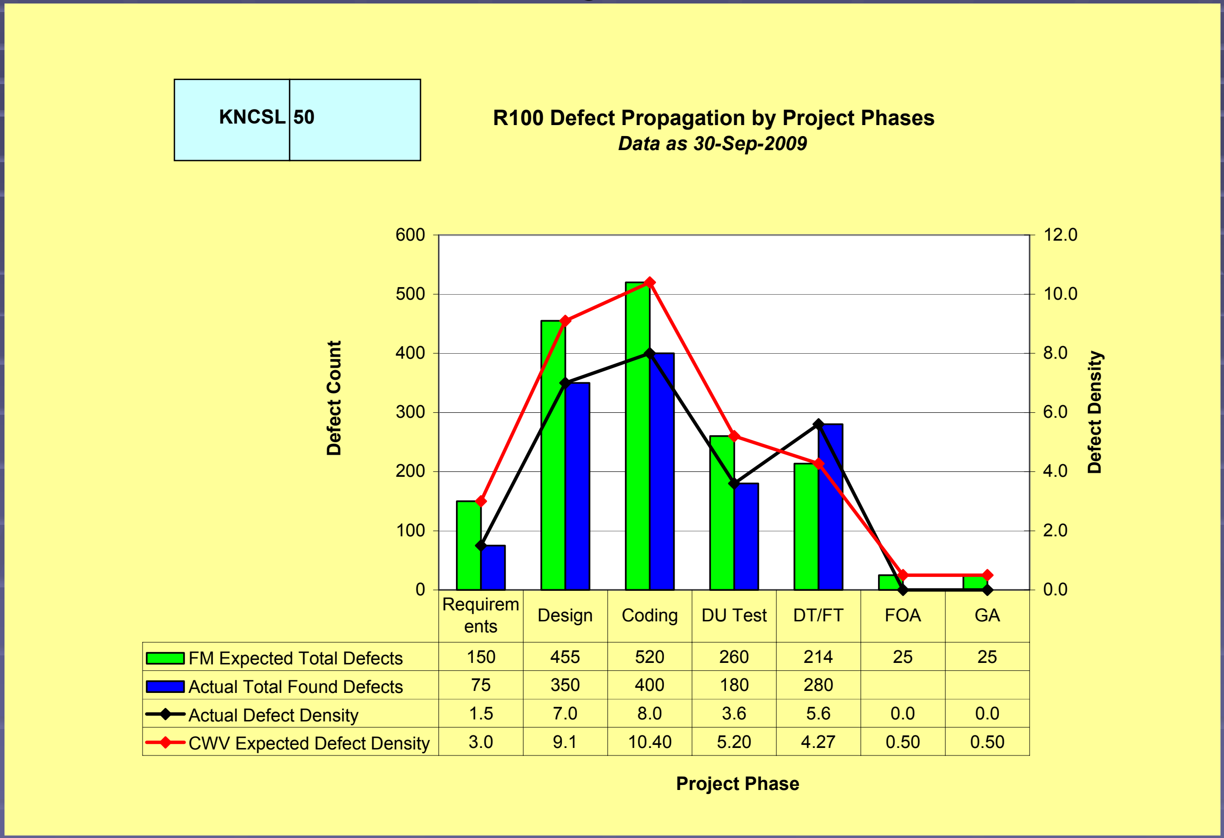
- **Inputs**- size of the new project in KNCSL – For the R100 project size is estimated as 50 KNCSL
    - **The baseline model of the organization** has
    - Defect injection rate for each phase
    - Defect removal rate for each phase

- **Output** – the expected defects injected / removed per phase is arrived by using the baseline model
- The baseline model indicates that 25 defects are slipping to the customer (residual defects) .
- The outgoing defect density is 0.5 defects / KNCSL. The cumulative defect removal efficiency is 98.5% .



Estimated KNCSL		50.00									
Release		R100									
Cumulative Removal Efficiency	50.0%		67.2%		72.6%		86.5%		98.5%		
Fix on Fix Rate							20%		10%		
Injected DD	6.00		12.00		13.00		1.04		0.43		
	Requirement		Design		Coding		DUT		Formal Test		CUSTOMER
Remaining DD		3.00		5.90		8.50		4.34		0.50	
Removal Efficiency	50.0%		60.7%		55.0%		54.5%		89.6%	25	
Removed DD-Baseline	3.00	Goal=14--18	9.10	Goal=14--18	10.40	Goal=4--8	5.20	Goal=4-7	4.27		
Baseline Removed Faults	150		455		520		260		214		

Case Study – Project R100



KNCSL	50

R100 Defect Propagation by Project Phases
Data as 30-Sep-2009

	Requirements	Design	Coding	DU Test	DT/FT	FOA	GA
🟩 FM Expected Total Defects	150	455	520	260	214	25	25
🟦 Actual Total Found Defects	75	350	400	180	280		
◆ Actual Defect Density	1.5	7.0	8.0	3.6	5.6	0.0	0.0
◆ CWV Expected Defect Density	3.0	9.1	10.40	5.20	4.27	0.50	0.50

Phase : Formal testing is complete , the defects have exceeded the expected number of defects.

Analysis :The defects missed in the Requirements /Design /Code / DU test phase is causing the high defect finding .

Corrective action: Defect analysis to understand the root cause & identify counter measures

Plan for process improvements to meet the Organization goal

Defect Analysis -Project R100

- **Requirements Phase -**misunderstood or ambiguous architecture/requirements
  - **Defect Prevention**
    - Form a core team comprising of Systems Engineering ,Architecture development and test) who will be involved in the project .
    - Improved understanding will reduce defects injected
    - Core team creates the Requirements, Architecture , Design & Test artifacts
    - When the development team ramps up the core team will impart the knowledge
  - **Defect removal**
    - Use the Core team for Requirements, Architecture , Design & Test artifacts reviews
- **Design Phase**
  - **Defect Prevention**
    - Feature Interaction matrix to be created & included in the design
  - **Defect Removal**
    - Scenario based Design reviews

Defect Analysis -Project R100

- **Coding Phase - High number of Static Defects -**Memory allocation issues , Null pointer were more than 80 % of the defects
  - **Defect Prevention**
    - Coding standards enhanced . Team trained on the Coding standards
  - **Defect removal**
    - Minimize Coding Phase errors slipping to test
    - Enforce use of static analyzer tools
    - Use scenario based code reviews

Process improvement

The recommendation [2]was to use Static analysis (analyzing source code for good or bad properties) and dynamic analysis (analyzing at run-time for good or bad behavior) for improving the Code quality .

Static analysis tool Flex lint (Compile time analysis) was selected to be used in the R200 project

Static Code Analysis

- Examine Source Code
- Look for (usually bad) properties
  - *Uninitialized* variable usage
  - *NULL* pointer dereferencing
  - *Out-of-bounds* array access
  - portability problems
  - security problems
  - coding style
  - code complexity

"UNO" problems

- *Error type*
- syntactic
- type
- coding style
- corner cases (array bounds)
- algorithm errors

- *Caught by*

compilers (automatic, fast)

static & dynamic analyzers (automatic, slower)

"verifiers" (human+automated,slowest)

A few sample FEM Options used

•FEM-530 is un-initialized symbol

•FEM-644 is possible un-initialized symbol

•FEM-645 is possible un-initialized symbol

•FEM-1541 is member possible not initialized by constructor

•FEM-1744 is member possible not initialized by constructor

•FEM 413 Likely use of null pointer in org operator reference -

•FEM 744 Switch statement does not have default

•FEM 416 out-of-bounds pointers such as "int a[10]; a[10] = 0;"

Static Code Analysis

Deployment plan

- **Objective**
  - Improve the quality of deliverables by removing the static errors.
  - Increase Code Phase defect removal efficiency
- **Scope**
  - All software modules
- **Program milestones**
  - New FEM option list definition & base lining
  - Provide orientation on new FEM option list & their significance to their respective teams
  - Deployment of New FEM option list in Build & development environment
  - Resolve all errors & cleanup code
  - Conduct Spot audits to verify effectiveness of Flexelint usage
  - After the results are ascertained share the results , learning & recommendation for Tools & Process management teams for formally rolling out to the Organization
- **Static analyzer usage process**
  - Flexelint report creation - Developers to use *rncchkLOC* command to create the report
  - Flexelint report analysis & resolution
  - Create the report prior to Code review – Submit report to Moderator for verification
  - Moderator to verify that the output is clean without any errors
  - After rework fixing all the review comments create the report again .
  - Developer to ensure that any new errors introduced are resolved & fixed .

Scenario based Reviews

- Why ? External studies indicate that 35% more defects are found using Scenario based reviews [3]. Pilot programs conducted in the organization indicated up to 20 % more defects are found
- In a typical code review, review will start from a logical starting point like the file containing the top-level function/procedure such as main().
- In a Scenario based review the review sequence is determined by the criticality of the scenario
- Scenario based reviews enhance the **effectiveness** of reviews
  - By providing a clear understanding of **logic** and **interface** solutions implemented by design or code to
  - A method for guiding document reviewers or code inspectors through the actions taken by software in response to one or more "events" (e.g. arrival of a message, occurrence of a hardware error, etc.).
- Scenarios represent the design chosen and permutation and combinations of the design chosen.

Scenario based Reviews Deployment plan

- **Objective**
  - enhance the effectiveness of reviews/reviews
  - Increase Code Phase defect removal efficiency by 20 %
- **Scope**
  - All software modules
- **Program milestones**
  - Provide training on Scenario based Reviews & their significance to their respective teams
  - Deployment of Scenario based Review Process
  - Creation of Templates , availability of experts to hand hold during the initial stages
  - Conduct Spot audits to verify effectiveness of Scenario based Review usage
  - After the results are ascertained share the results , learning & recommendation for Tools & Process management teams for formally rolling out to the Organization
- **Scenario usage process**
  - Scenario Doc is created during Design phase .Used for Design Reviews , Code review , Test plan creation and for future training for maintenance teams

Results

- The model is built with empirical data of similar projects , have been applied to various projects ever since . The model has been in practice for more than 10 years , fine tuned with the lessons learnt .

The model is calibrated with actual defects & field data. Whenever Process improvements are made and the standard process is changed the model is revised based on the actual data

- The range of prediction is +/– 15 % with more than 90 % confidence level
- The projects that have deviated significantly from the model are investigated .
- The possibilities are
    - The data collected from projects may be wrong . If yes, the data validity is ensured first
    - the processes in the Project are extremely good . There may be some best practices that can be shared. The team might be a very experienced team
    - The project may need some training , process improvements
- By using the Defect Prediction Model , reliable product quality can be planned , tracked & improved

Limitations , expectations

- **Limitations**
  - **Size is the primary input .** The model is highly sensitive to size fluctuations . Accurate prediction / measurement of size is critical .
  - Need a accurate estimation process . Need to automate the Size measurement of the product .
  - The process used, the technology and the team composition is similar across projects. Hence the data provides a good fit for prediction . If the projects have lot of differences in the above factors , the accuracy will be affected .
  - The model does not account for changes due to Product complexity , Team composition

- **The model expectations are**
  - The project uses a stable process, (under Statistical Process Control)
  - The In-process data is accurate (data from reviews, reviews, tests)
  - The defects from the field are accurately captured
  - The project uses a standardized lifecycle (same phases)

Critique of the model

- Defect vs. Failure - Is defect free software reliable ? [4]
- There is a debate that removal of defects in the software does not necessarily guarantee , high reliability or absence of failures
- All defects are not equal .There is a class of defects (failure inducing) that impact reliability
- Need to have defect count of these failure inducing defects
  - CR Severity guidelines – Aligned to capturing the reliability field data
  - Sev1 – Service outage, sev2 , priority -1 service impacting , partial outage
  - Data validity , defects scrubbed in CRRB
- Field data on SW outages collected accurately
- Defect injection rate of Sev 1 & Sev2 Prio 1 in addition to all severity is maintained
- With the above data the total defects as well as the failure inducing defects can be clearly maintained The residual data collected from the field is also segregated as defects & failure inducing defects
- Using the Defect Prediction Model , Product quality can be planned , tracked & improved

Barriers

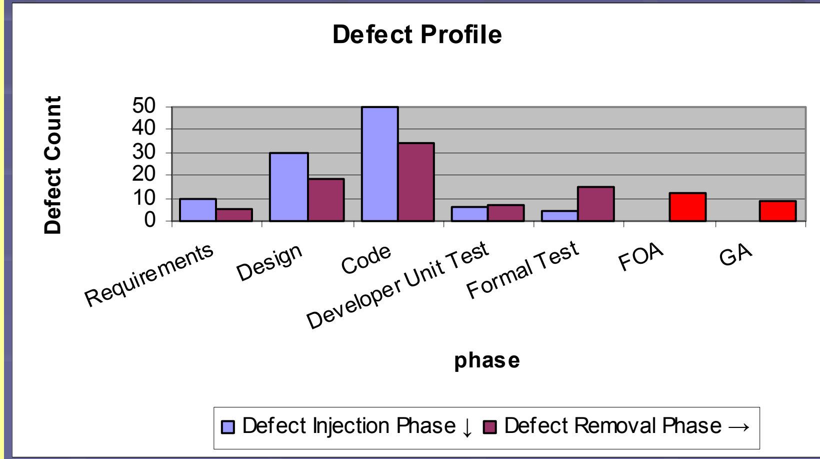
- To ensure the success of any technique / best practice there are Organizational factors that are important and have to be taken care .
- Need a culture that accepts quantitative management
- The defect count needs to be treated as a process goal for guiding quality management .It must not be confused as targets to be met
- The data integrity is very important for maintainging the accuracy of the model . Compliance from all stakeholders is necessary
- Automation of the data capture as much as possible ,helps data integrity

Empirical Defect Prediction Model

How to create the Defect Profile for a completed project

- **Defect Filter Matrix –** The table below is based on the data shown in the example in the previous slide
- **Defect profile –** The chart below shows the process behavior of the organization's defect injection & removal
- **How to create the baseline model for your project**
    - **Defect Filter Matrix -**The number of defects that are injected & removed during the phases in the project have to be captured
    - The Field found data needs to be updated as & when the defects are found by the customer
    - The resultant defect profile indicates the completed project’s defect profile

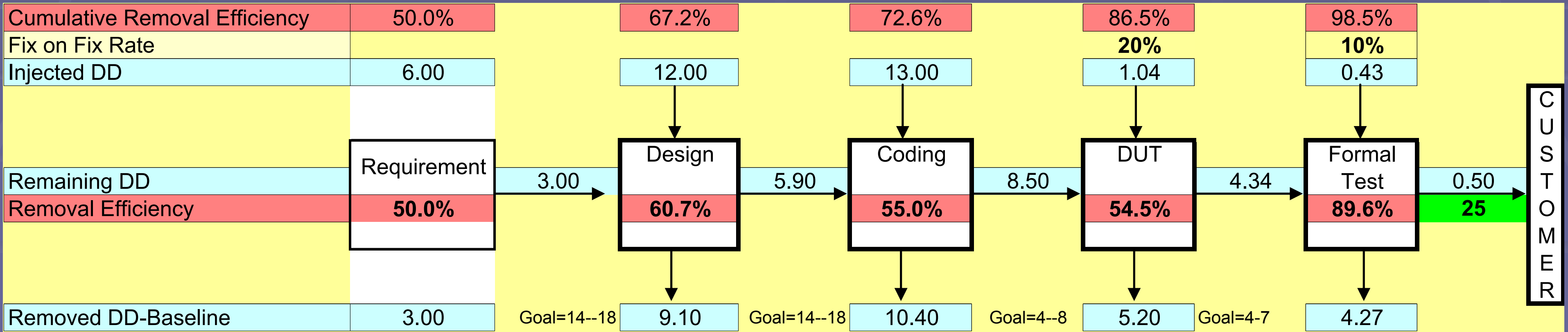
	Defect Removal Phase →							
Defect Injection Phase ↓	**Req Review**	**Design Review (High /Low level)**	**Code Review**	**Developer Unit Test**	**Formal Test**	**FOA**	**GA**	**Defects Injected Total**
Requirements	5	1	1		1	1	1	**10**
Design		17	3	2	5	2	1	**30**
Code			30	3	7	6	4	**50**
Developer Unit Test				2	1	1	2	**6**
Formal Test					1	2	1	**4**
FOA								**0**
GA								**0**
Defects removed total	**5**	**18**	**34**	**7**	**15**	**12**	**9**	**100**



Empirical Defect Prediction Model

How to create the baseline model for your organization

- **How to create the baseline model for your organization**
  - **Defect Filter Matrix –**
  - create the Organization’s defect filter matrix with data from multiple projects
  - Need a size metric to convert the absolute number of defects injected / removed into Defect Injection rate / Defect removal rate . Let us use Lines of code (KNCSL) as the size metric .
  - Capture the size of the projects in KNCSL . Populate the number of defects that are injected & removed during the phases in the projects .Defect density / KNCSL can be computed . Defects / FP , Defects / use case also works fine .
  - The defect Injection rates (DIR) & the defect removal efficiency (DRE) are computed as Defects / KNCSL
  - The DRE (defect removal efficiency) of the various phases & the Cumulative DRE for each phase is arrived at
  - With the historical defect data of completed projects, statistical limits can be ascertained for DIR & DRE . The DIR, the Upper & lower limits specifying the range
  - The resultant baseline model for the organization will look like the below diagram
  - Fix on Fix rate – the defects introduced when fixing a defect / bad fix ,This is computed as a percentage of the Removed DD



Cumulative Removal Efficiency	50.0%		67.2%		72.6%		86.5%		98.5%		
Fix on Fix Rate							**20%**		**10%**		
Injected DD	6.00		12.00		13.00		1.04		0.43		
	Requirement		Design		Coding		DUT		Formal Test		CUSTOMER
Remaining DD		3.00		5.90		8.50		4.34		0.50	
Removal Efficiency	**50.0%**		**60.7%**		**55.0%**		**54.5%**		**89.6%**	**25**	
Removed DD-Baseline	3.00	Goal=14--18	9.10	Goal=14--18	10.40	Goal=4--8	5.20	Goal=4-7	4.27		

References, Acronyms

References

1. TL 9000- QUEST forum . Quality excellence for suppliers of Telecommunications forum
2. New Techniques in Static and Dynamic Analysis - Dr. Howard Trickey, Bell Laboratories presented at SPIN Bangalore – Slide 11 & 12 based on this talk
3. Boehm and Basili, “Software Defect Reduction Top 10 List”, Computer, January 2001.
4. A critique of Software defect prediction models – Norman E Fenton , Martin Neil – slide 16

Acronyms

1. DIR – Defect Injection rate measured as defects/ KNCSL
2. DRE – Defect removal rate measured as defects/ KNCSL
3. DD – Defect density captured as defects/ KNCSL
4. KNCSL –Kilo Non commented source lines (1000 lines of code)
5. FOA – First office application – A customer site where acceptance testing is done
6. GA – General availabilty , when the product is available to the market deployment . This usually follows a successful FOA
7. FEM – Flexelint error message
8. CRRB – Change request review board , defect review board .

Every defect will need a change request to make the software change . The board discusses the defects and assigns to the developers . Comprises of cross functional team for speedy resolution of defects