Statistical SQA

Introduction

- Statistical quality assurance reflects a growing trend throughout industry to become more quantitative about quality.
- For software, statistical quality assurance implies the following steps:
    1. Information about software defects is collected and categorized.
    2. An attempt is made to trace each defect to its underlying cause (e.g., non-conformance to specifications, design error, violation of standards, poor communication with the customer).
    3. Using the Pareto principle (80 percent of the defects can be traced to 20 percent of all possible causes), isolate the 20 percent (the "vital few").
    4. Once the vital few causes have been identified, move to correct the problems that have caused the defects.

Statistical Quality Assurance

- This relatively simple concept represents an important step towards the creation of an adaptive software engineering process.
  - Changes are made to improve those elements of the process that introduce error.
- Assume that a software engineering organization collects information on defects for a period of one year.
  - Some of the defects are uncovered as software is being developed.
  - Others are encountered after the software has been released to its end-users.

Statistical Quality Assurance

All the errors can be tracked to one (or more) of the following causes:

- incomplete or erroneous specifications (IBS)
- misinterpretation of customer communication (MCC)
- intentional deviation from specifications (IDS)
- violation of programming standards (VPS)
- error in data representation (EDR)
- inconsistent component interface (ICI)
- error in design logic (EDL)
- incomplete or erroneous testing (IET)
- inaccurate or incomplete documentation (IID)
- error in programming language translation of design (PLT)
- ambiguous or inconsistent human/computer interface (HCI)
- miscellaneous (MIS)

Table 1: DATA COLLECTION FOR STATISTICAL SQA

	Total		Serious		Moderate		Minor	
Error	No.	%	No.	%	No.	%	No.	%
IES	205	22%	34	27%	68	18%	103	24%
MCC	156	17%	12	9%	68	18%	76	17%
IDS	48	5%	1	1%	24	6%	23	5%
VPS	25	3%	0	0%	15	4%	10	2%
EDR	130	14%	26	20%	68	18%	36	8%
ICI	58	6%	9	7%	18	5%	31	7%
EDL	45	5%	14	11%	12	3%	19	4%
IET	95	10%	12	9%	35	9%	48	11%
IID	36	4%	2	2%	20	5%	14	3%
PLT	60	6%	15	12%	19	5%	26	6%
HCI	28	3%	3	2%	17	4%	8	2%
MIS	56	6%	0	0%	15	4%	41	9%
Totals	942	100%	128	100%	379	100%	435	100%

Statistical Quality Assurance

- To apply statistical SQA, Table 1 is built.
- The table indicates that IES, MCC, and EDR are the vital few causes that account for 53 percent of all errors.
  - It should be noted, however, that IES, EDR, PLT, and EDL would be selected as the vital few causes if only serious errors are considered.
- Once the vital few causes are determined, the software engineering organization can begin corrective action.
  - For example, to correct MCC, the software developer might implement facilitated application specification techniques to improve the quality of customer communication and specifications.
  - To improve EDR, the developer might acquire CASE tools for data modeling and perform more stringent data design reviews.
  - The corrective action focuses primarily on the vital few.
  - As the vital few causes are corrected, new candidates pop to the top of the stack.

Statistical Quality Assurance

- Statistical quality assurance techniques for software have been shown to provide substantial quality improvement [ART97].
- In some cases, software organizations have achieved a 50 percent reduction per year in defects after applying these techniques.
- In conjunction with the collection of defect information, software developers can calculate an error index (EI) for each major step in the software process [IEE94].

Statistical Quality Assurance – Error Index

- After analysis, design, coding, testing, and release, the following data are gathered:

  E_i = The total number of errors uncovered during the ith step in the software engineering process

  S_i = the number of serious errors

  M_i = the number of moderate errors

  T_i = the number of minor errors

  PS = size of the product (LOG, design statements, pages of documentation) at the ith step

  W_s, W_m, W_t = weighting factors for serious, moderate, and trivial errors,

  where recommended values are $W_s = 10$, $W_m = 3$, $W_t = 1$.

- At each step in the software process, a phase index, PI_i, is computed:

  $$\mathbf{PI_i = W_s(S_i / E_i) + W_m(M_i / E_i) + W_t (T_i / E_i)}$$

- The error index is computed by calculating the cumulative effect on each PIi, weighting errors encountered later in the software engineering process more heavily than those encountered earlier:

$$\mathbf{EI = \Sigma(i \times PI_i)/PS = (PI_1 + 2PI_2 + 3PI_3 + \ldots\ iPI_i)/PS}$$

- The weighting factors for each phase should become larger as development progresses. This rewards an organization that finds errors early.
- The error index can be used in conjunction with information collected in Table 1 to develop an overall indication of improvement in software quality.

The application of the statistical SQA and the Pareto principle can be summarized as:

Spend your time focusing on things that really matter, but first be sure that you understand what really matters!