Software Engineering Metrics

Linda M. Laird

"In Praise of Defects"

Stevens Institute of Technology

linda_m_laird@msn.com

908 232 5954

Some Quotations

- "*If debugging is the process of removing bugs, then programming must be the process of putting them in.*" Unknown
- *"No software system of any realistic size is ever completely debugged --- that is, error free*." Edward Yourdon and Larry Constantine[i]
- "…*defects do follow a Rayleigh pattern, the same as effort, cost, and code constructed. This curve can be projected before the main build begins; it can be finely tuned. It is the key ingredient for the statistical control of software reliability.*" Lawrence H. Putnam and Ware Myers

[i] Yourdon and Constantine, Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design, Prentice-Hall, Englewood Cliffs, NJ, 1979

Some Questions:

- Your system has no reported field defects. It was released six months ago. Is it time to break out the champagne?
- You are building a commercial application. Your CEO, who is very dedicated to quality, directs you to find and fix 99% of the bugs before you ship. What is your response?

Some Answers

- Zero Defects?
  - HP had a system with no reported field defects. Initially, it was thought to be an example of "zero defects." They later discovered that was because no one was using it.[i] Be very careful to understand the usage of your system if the number of reported field defects is significantly lower than expected. It may be due to low usage.
- Fix 99% of the Bugs?
  - Studies repeatedly show that finding and fixing 99% of bugs is near impossible, and very costly. Our suggestion is discuss the defect benchmark data with your CEO, and to suggest that a reliability or availability target may be more appropriate.

Ref: From Kan

In Praise of Defects

- Are defects good or bad?

The Silver Lining of Defects

- Defects are real, observable manifestations and indications of the software development progress and process
  - From a schedule viewpoint – progress
  - From a quality viewpoint – early indication
  - From a process engineering – indicate effectiveness of different parts of the process and targets for improvement
- You can see them, count them, predict them, trend them
- To those of us who pay attention – they give a wealth of understanding and insight into a project and the software process

Defects Table of Contents

- Faults and Failures
- Defect Dynamics and Behaviors
- Defect Projection Techniques and Models
  - Dynamic Models
  - Static Models
    - Defect Removal Efficiency
    - Coqualmo
- Defect Benchmarks

Failures

Faults

Defects

?

Errors

Bugs

Faults vs. Failures

If the code that contains the faults is never executed in operation…….

Then

the system never fails. And the Mean Time Between Failures (MTBF) -> Infinity

Conversely….

If the there is only 1 fault in the entire system, and it is executed every time, then the system has no reliability, and the MTBF -> 0.

Failures only occur in operation.

Faults are defects in the system that may or may never be seen in operation.

Defect Dynamics and Behaviors

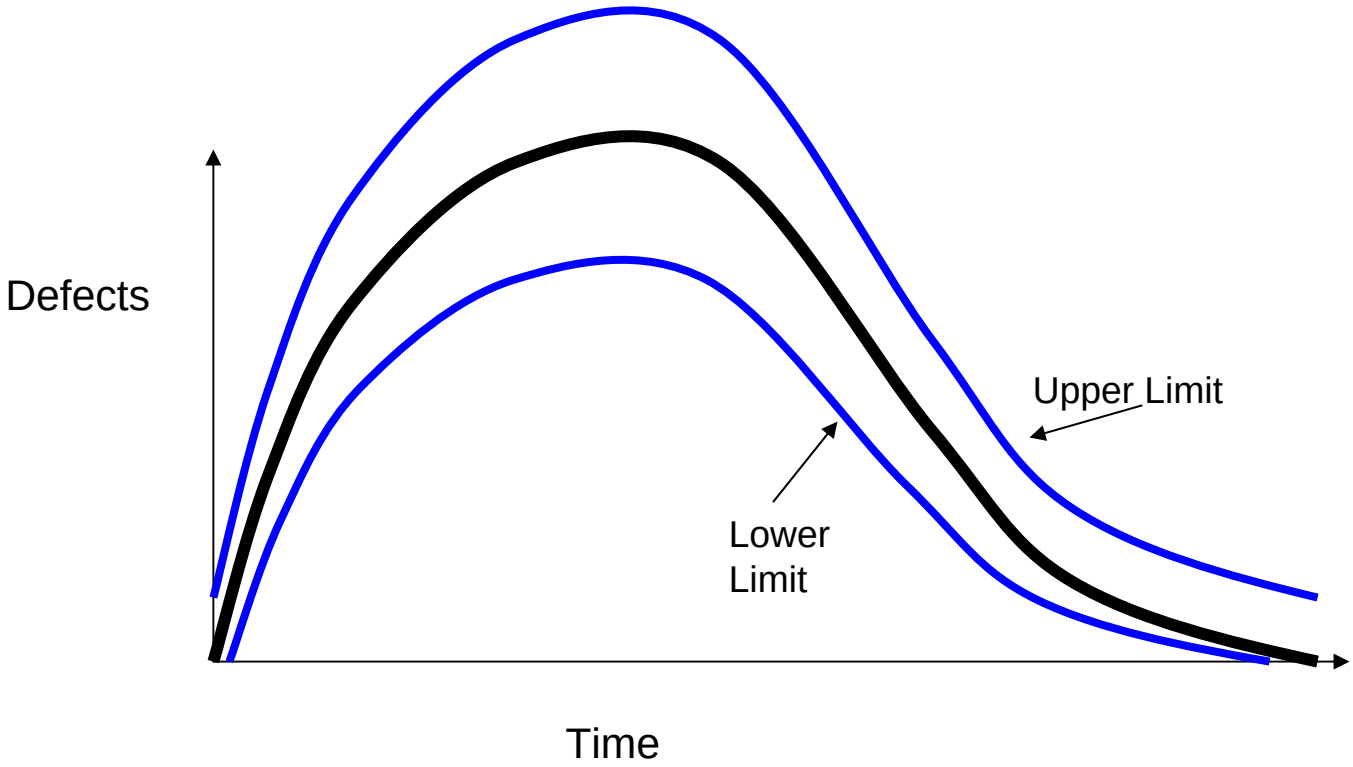
Defect Dynamics and Behaviors

- Defects have certain dynamics, behaviors, and patterns which are important to understand in order to understand the dynamics of software development

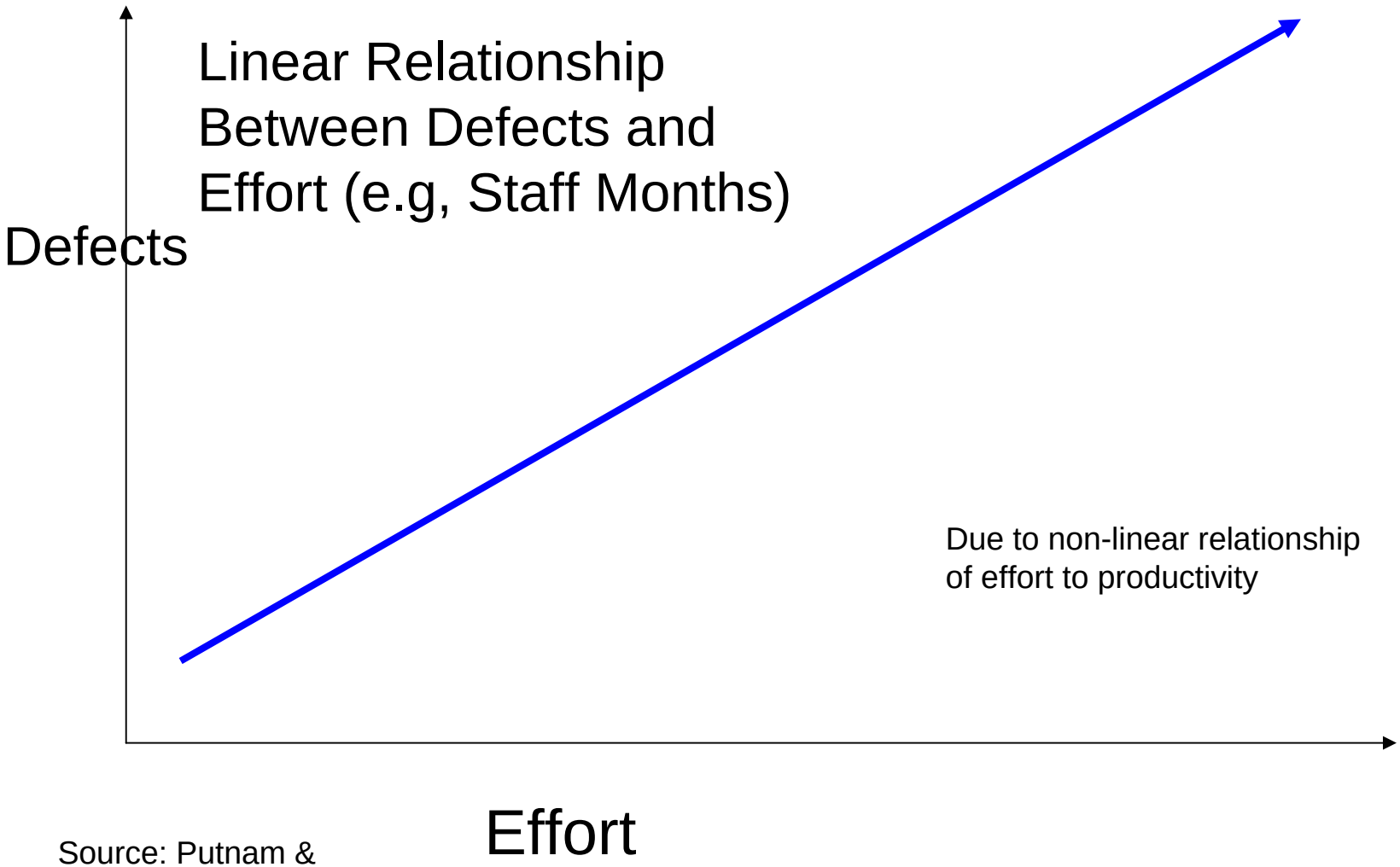
- This is information that should be part of your "intuition" about software development – your knowledge that this is how software development typically behaves.

Projected Software Defects

In general, follow a Rayleigh Distribution Curve…can predict, based upon project size and past defect densities, the curve, along with the Upper and Lower Control Bounds

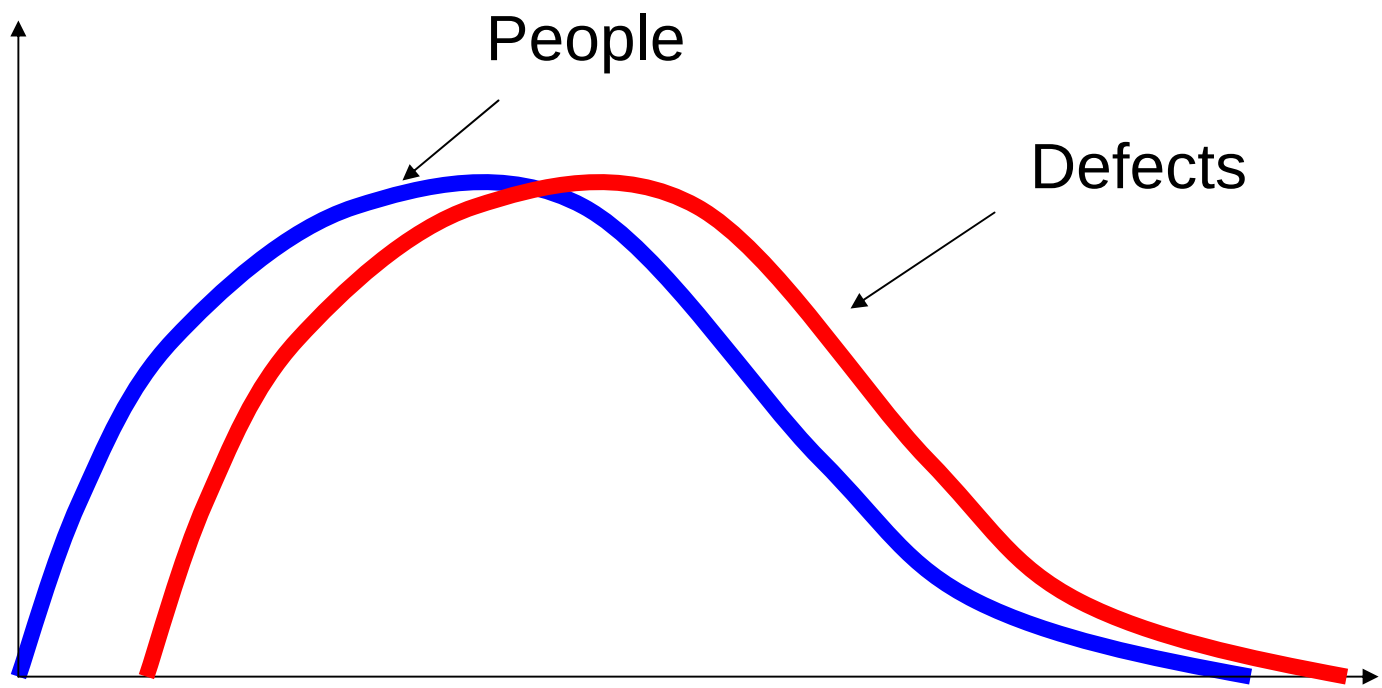


Defects vs. Effort



Source: Putnam & Myers

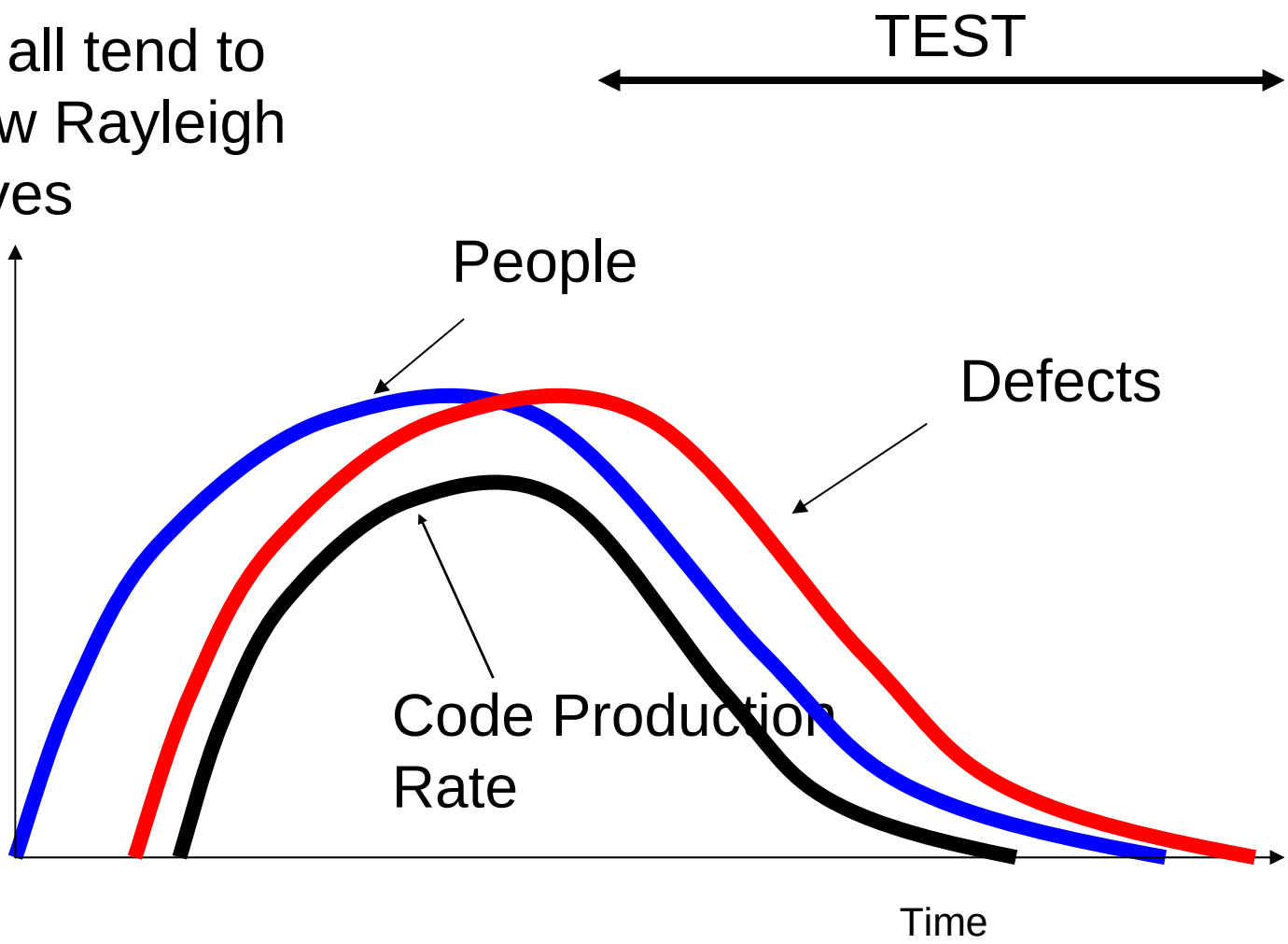
Defects Detected tends to be similar to Staffing Curves – time lag for error dectection



Source: Industrial Strength Software, Putnam & Myers, IEEE, 1997

Which is related to Code Production Rate

And all tend to follow Rayleigh Curves



Note: Period during test is similar to exponential curve

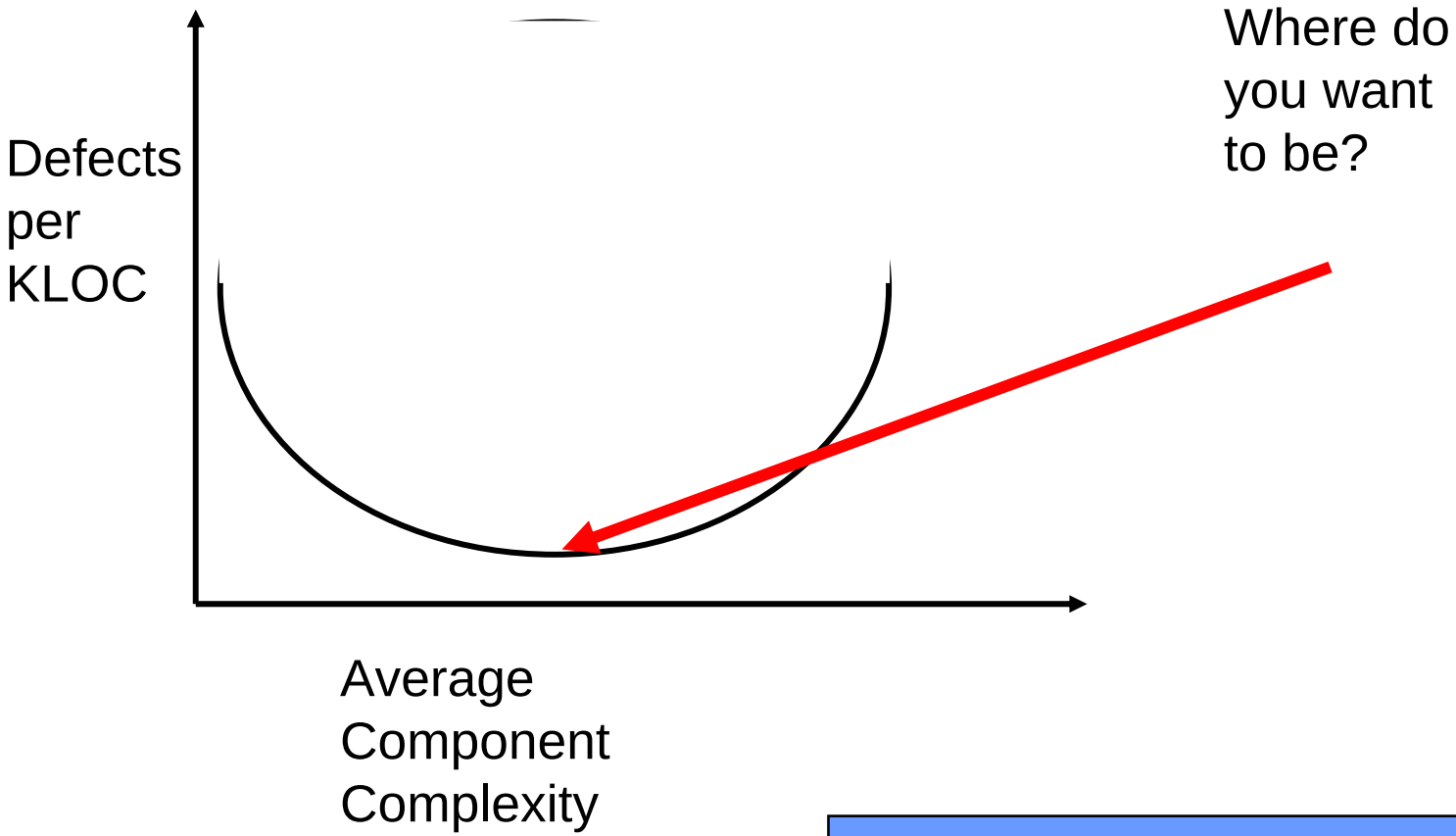
Source: Putnam & Myers

What conclusions can YOU draw from this?

- My view: People tend to make errors (which lead to defects) at a relatively constant rate…..
  - E.G., the more people and the more effort….the more defects (unless, of course, they are working to prevent or remove defects)

Defect Density vs. Module Complexity

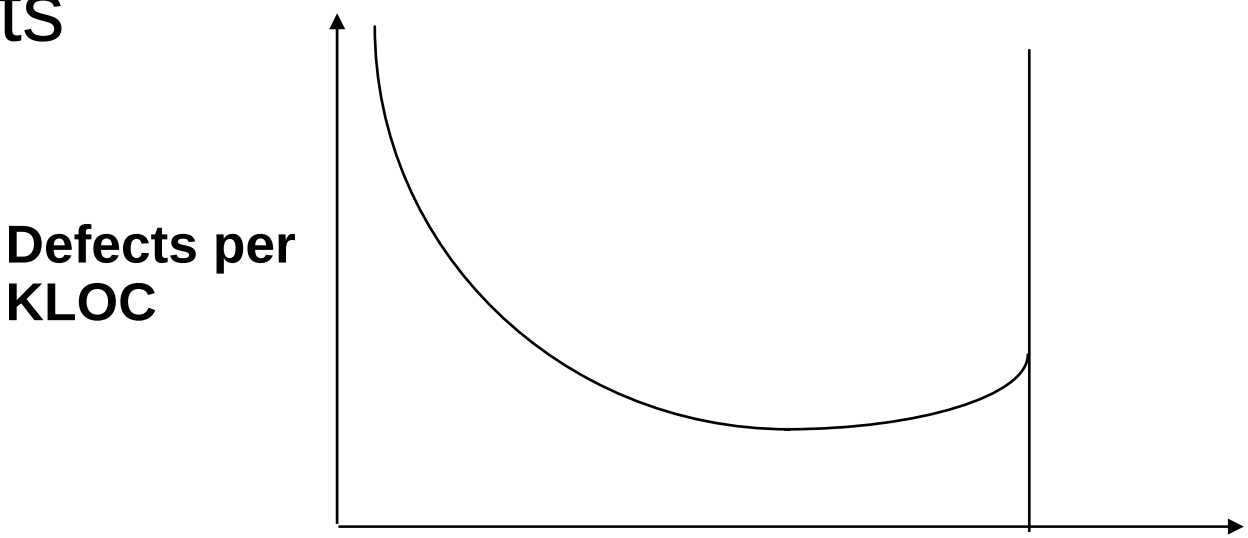
- The bathtub chart



Ref: Hatton, Les, “Why don’t we learn from our mistakes?”, 1997

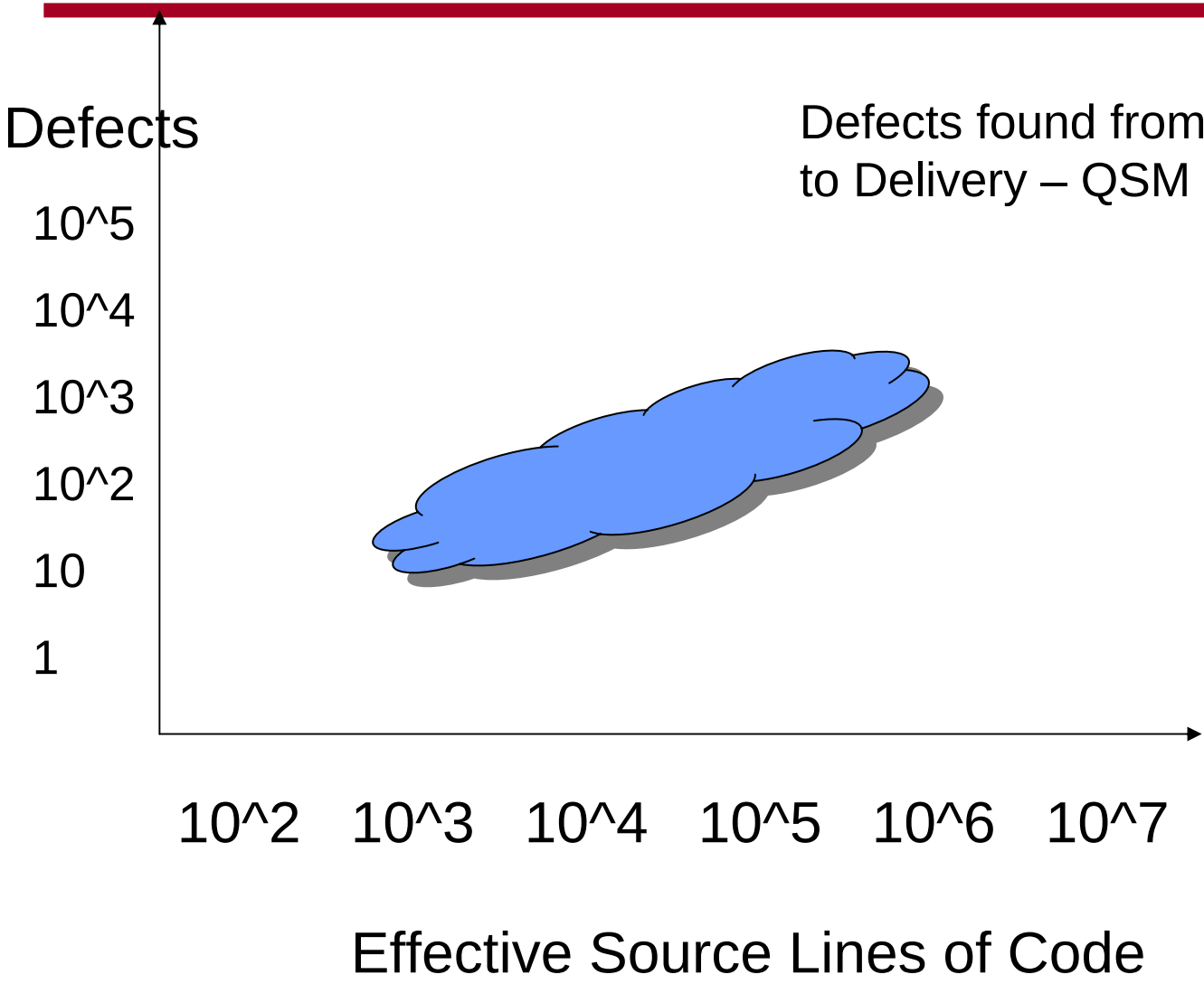
Implications of the Bathtub Curve

- Concentrate defect detection on modules that have either very high or very low complexity
- Truncate complexity to force fewer defects



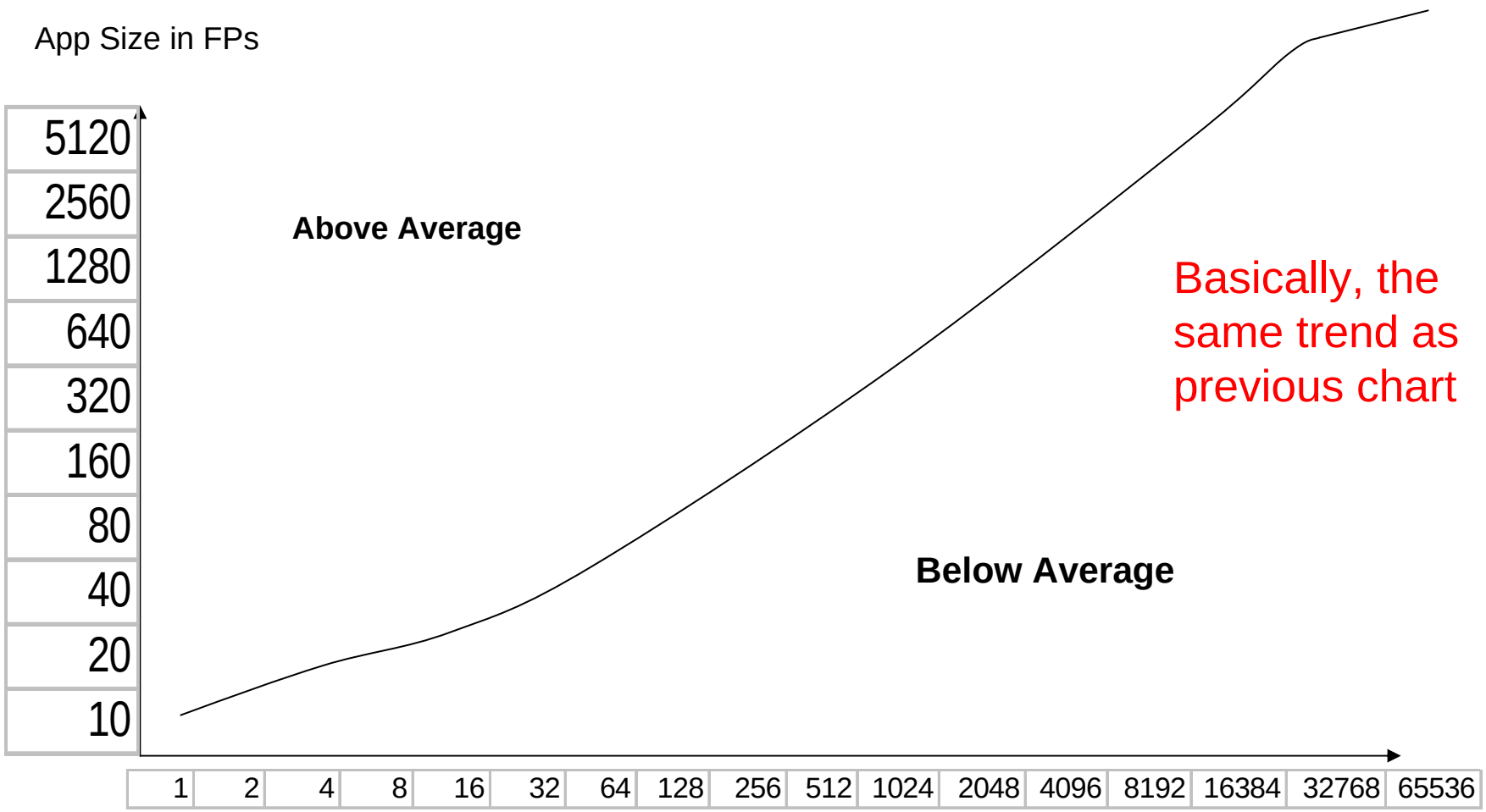
Ref: Hatton, Les, “Why don’t we learn from our mistakes?”, 1997

Defect Density vs. System Size



Ref: Putnam and Myers, Industrial Strength Software, 1997

US Industry Ave. for Potential Software Defects



Source: Jones, Applied Software Measurement, 1991, Mcgraw-Hill, p 165

- So…you've been testing like mad, and you think you've turned the corner…because the number of new defects is slowing down….
  - Can you predict how many defects remain?
  - Can you predict how long it will take to find 95% of them?

- Defect Projection Techniques and Models

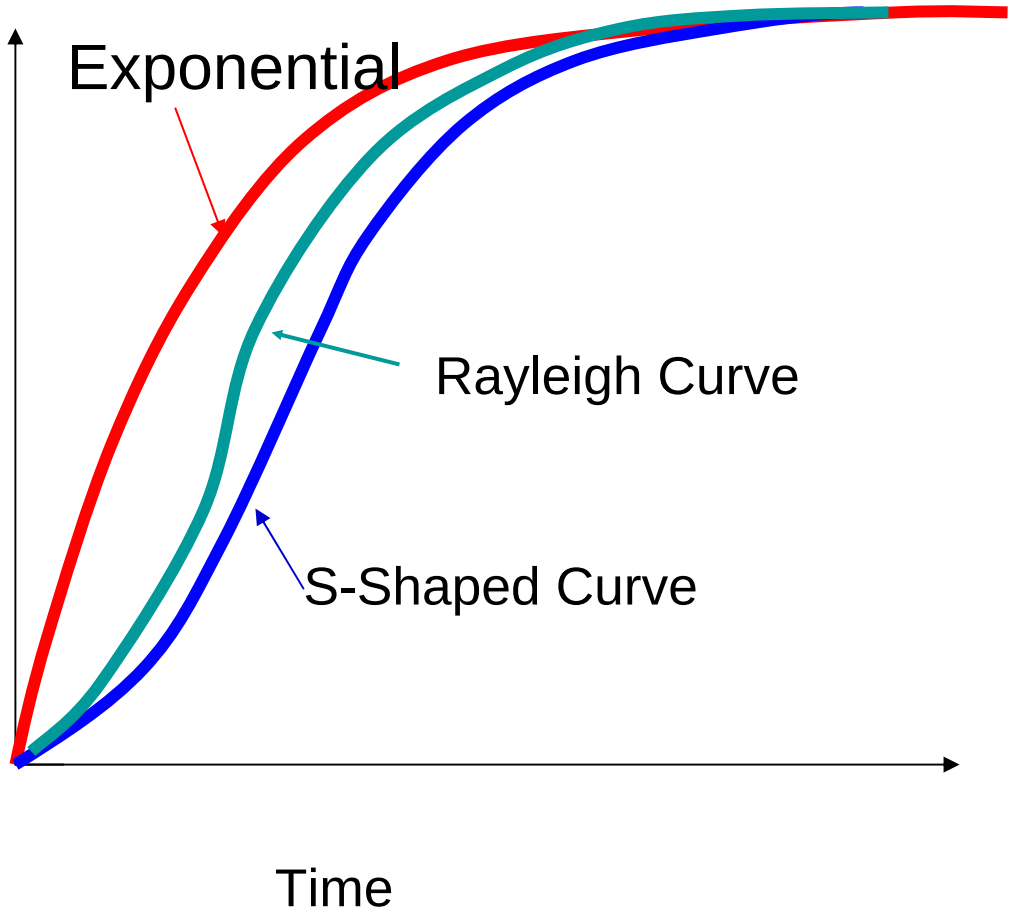
Defect Projection Techniques and Models

- The objective of defect prediction and estimation techniques and models is to
  - project the total number of defects
  - their distribution over time.
- Types of techniques - vary considerably
  - Dynamic models are based upon the statistical distribution of faults found.
  - Static models use attributes of the project and the process to estimate the number of faults.
- Dynamic models require defect data, and are used once the tracking of defects starts.
- Static models can be used extremely early in the development process, even before the tracking of defects begins.

Dynamic Defect Models

- Based on predicting, via calculations or tools, the distributions of faults, given some fault data.
  - Primary concept: the defects do follow distribution patterns, and given some data points, you can generate the arrival distribution equations.
- There are many different defect distribution equations
  - Primary ones are Rayleigh, Exponential, and S-Curves.
    - Rayleigh distributions are for modeling the entire development lifecycle
    - Exponential and S-Curves are applicable for the testing/deployment processes.

Dynamic Distribution Curves



Note-there are infinite Rayleigh, S, and Exponential curves. These just give you a look at the shapes.

Defect Distribution functions

- There are 2 primary functions in defect distribution
  - f(t) = Probability distribution function (PDF) of arrival rates of defects
  - F(t) = Cumulative distribution function of total number of defects to arrive by time t.
- They are related by:
  - $F(t) = \int_0^t f(t)dt$
- And f(t) describes the distribution of the defect arrivals– it can be a uniform distribution, normal, or other distributions

Rayleigh Model

- Defect Arrival Rate (PDF) – the number of defects to arrive at time t = $f(t) = K*(2t/c^2)*e^{-(t/c)2}$
- Cumulative Defects (CDF) -- the total number of defects to arrive by time t =

$$F(t) = K*(1-e^{-(t/c)^2})$$

- Where the c parameter is a function of the time t_{max} that the curve reaches its peak
- c = t_{max} * sqrt (2)
- K=total number of injected defects

Predicting defects using data points and the Rayleigh Distribution

- This means you can – assuming a Rayleigh Distribution….
  - Mathematically determine the curve and the equation, as long as you've hit the maximum.
- Note that the percent of defects that have appeared by t_m
  - = $100*F(t_m)/K = 100*(1 – \exp(-.5)) \sim = 40\%$.
  - Therefore, ~40% of the defects should appear by time t_m.

Using the Rayleigh Distribution

- Given n data points, plot them
- Determine t_{max} (the time t at which f(t) is max)
- Then since you have the formulae

$$F(t) = K * (1 - e^{-(t/c)^2})$$

$$f(t) = K * (2t/c^2) * e^{-(t/c)2}$$

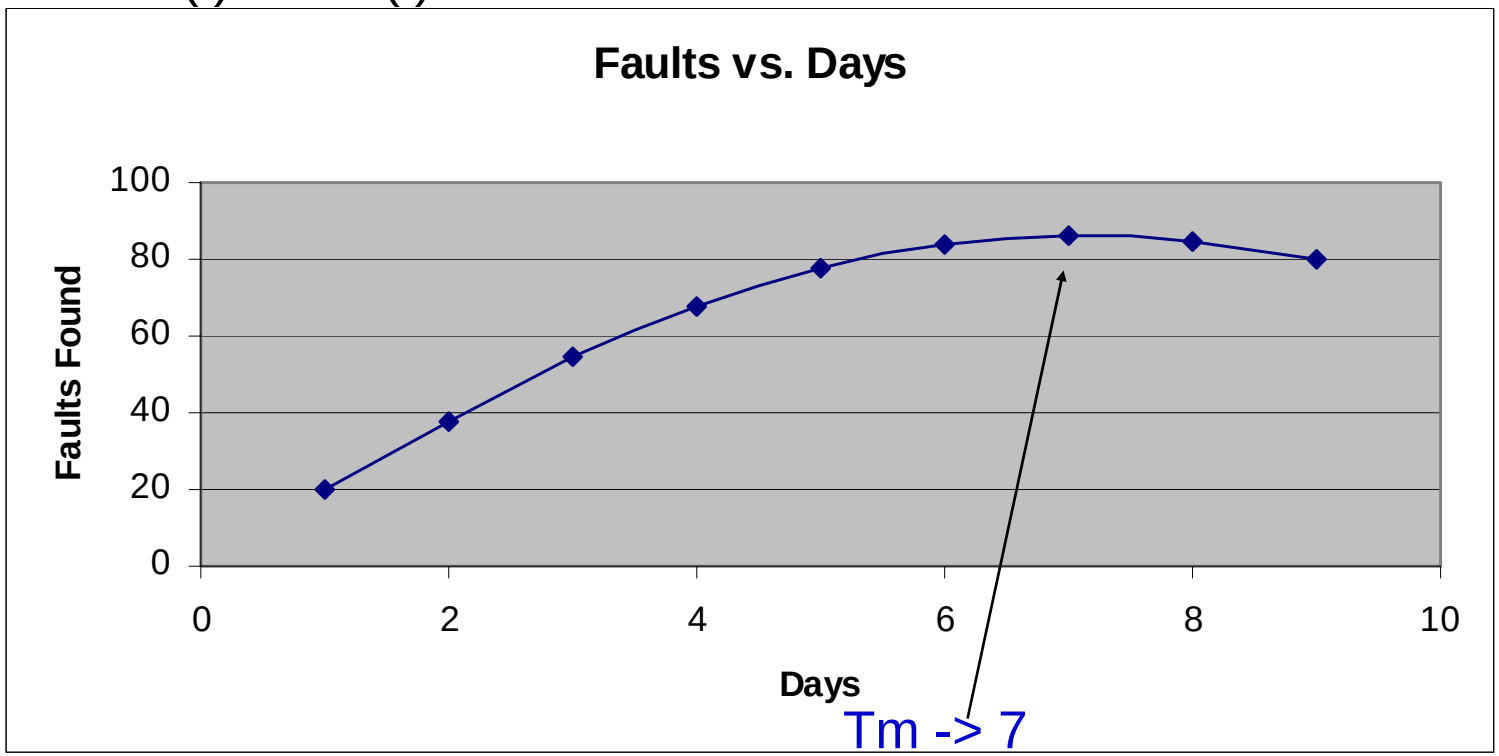
- Where F(t) is the cumulative arrival rate,f(t) is the arrival rate for defects, and K is the total number of defects…
- And you can then use these to predict the later arrival of defects.

Extremely Simple Method – the 40% rule

- Given that you have
  - defect arrival data, and
  - the curve has achieved its maximum at time t_m (e.g., the inflection point)
- You can calculate f(t), assuming the Rayleigh distribution
- The simplest method: use the pattern that ~40% of the total defects have appeared by t_m.
  - Crude calculation, but a place to start.

Extremely Easy Method Example

Given the data shown below (429 defects by Tm) – determine f(t) and F(t)



Therefore, expected total number of faults = 429*(100/40) = 1072 or ~1075

Then you can determine f(t), since you know K and T_m

Example continued

- Since: $F(t) = K * (1 - e^{-(t/c)^2})$

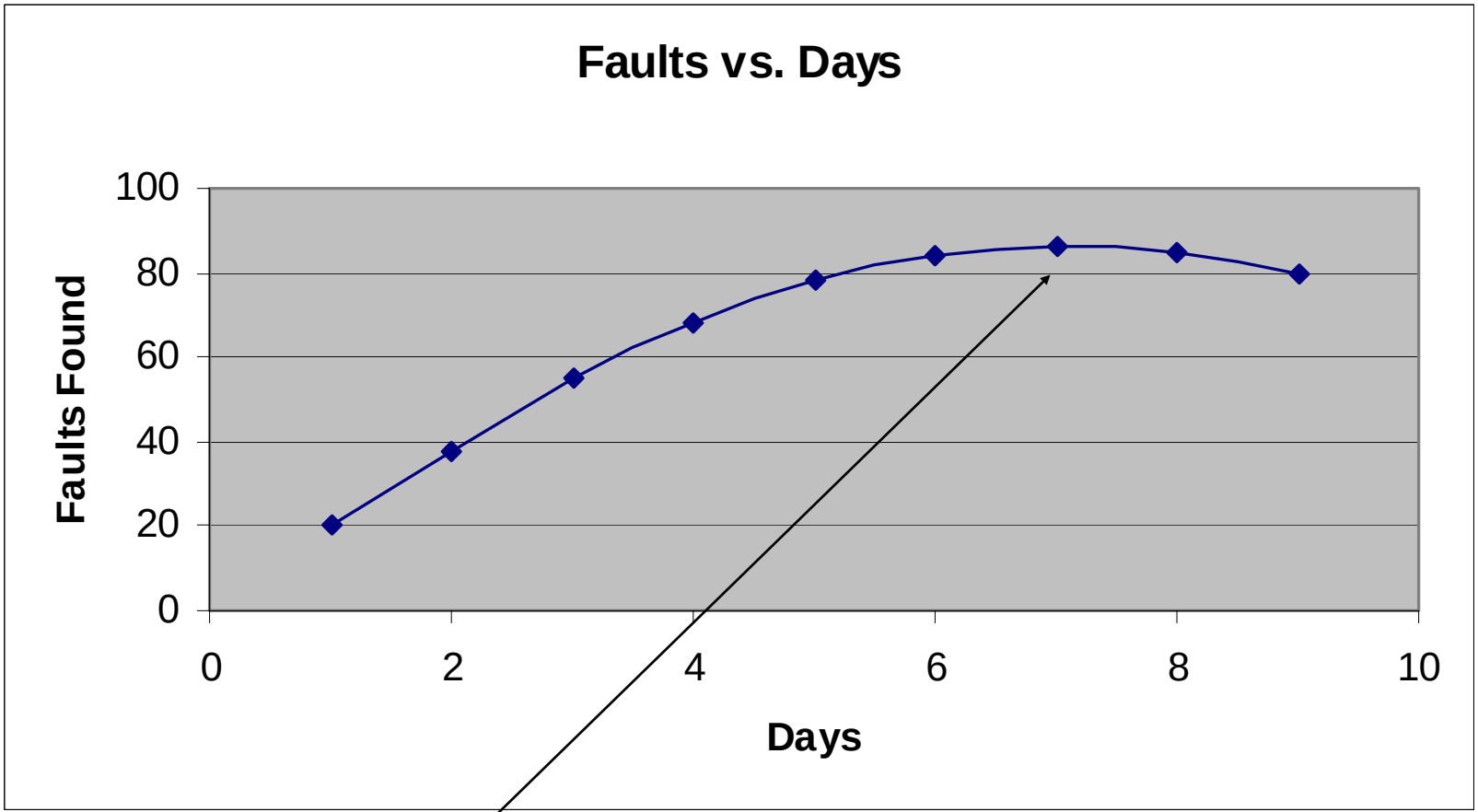
$$f(t) = K * (2t/c^2) * e^{-(t/c)2}$$

- Then substituting in:
  - f(t) = 1075 * t /49 * e$^{-\ t\text{^}2\ /98}$
    - = 21.93 * t * e$^{-\ .01t\text{^}2}$
  - F(t) = 1075 *[1 - e$^{-\ .01t\ \text{^}2}$]
- Then you could plot this out on the same graph and see how well it matches the data.

Predicting Arrival Distributions: Method 2

- You can solve for f(t) by using t_m and one or more data points to solve for K and f(t).
- The simplest way is to take just one data point … and at t=1

Example: 594 Faults found by day 9



What is the arrival function f(t)? Need tmax and K to determine f(t).

Tmax -> 7

$$f(t) = K * \left[(1/7)^2 t e^{-(1/2*7^2)t^2)} \right]$$

Then what would you do?

- Solve for K (use t = 1, defects = 20) => K=20*49/e(-1/98)= ~990
- You now have an equation:

$$f(t) = (990 / 49)te^{-(1/98)t^2}$$

$$= 20.2te^{-.01t^2}$$

- Then plot out the equation and use it to predict arrival rates (and also see how well it matches to data!)

Statistically Correct?

- At least three points are needed to estimate these parameters using non-linear regression analysis
  - further statistical tests (such as Standard Error of Estimate and Proportion of Variance are necessary) for these parameters to be considered statistically valid.
- In the case where high precision is important, use a statistical package or one of the reliability tools to calculate the line of best fit.

Statistically Close?

- As intermediate step, you can calculate K for all known values of t and f(t), and then take the mean and standard deviation.
- That is, $K= (49*f(t) * e^{-.01*(t^2)})/t$. These calculations are easily performed with a spreadsheet.
- Note:
  - these calculations are sensitive to t_m
  - although they are not completely valid in a statistical sense, they give the statistically-challenged practitioner a relatively easy method for calculating the distribution functions, with some initial control limits. Considering the precision of the typical data, this seems reasonable

What do these charts mean?

- If the defect data is a reasonable fit for a Rayleigh curves…then you can use it to predict outstanding defects, and defect arrival rates

Method 3: Predicting the arrival rates given total number of defects and schedule

- Given total number of defects expected over the life of a project
  - You can use the Rayleigh curves to predict the number of defects expected during any time period
  - You can also compare projected defects with actual defects found to determine project performance.
- Let T_d is the total duration of the project, such that 95% of all defects have been found.[1] K is the total number of faults expected for the lifetime of the delivery.
- Then the defects for each time period t is:
  - $f(t) = (6\ K/\ (T_d^2))*t*e\ ^{(-3(t/\ T_d)^{2})}$

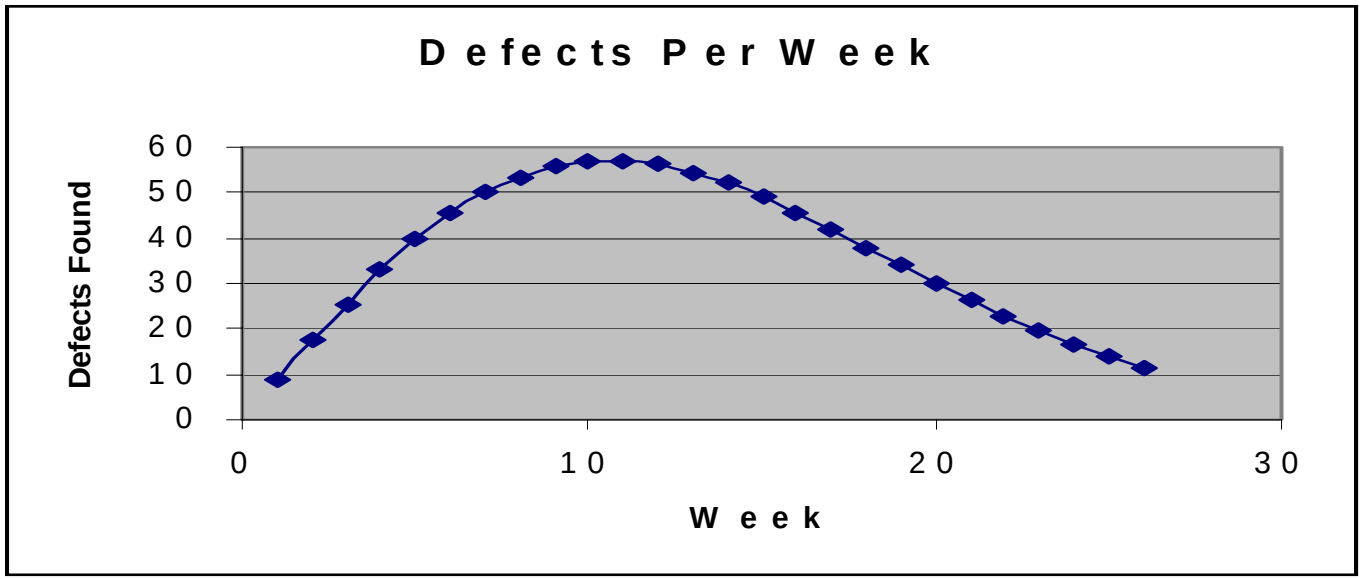
[1] The 95% number here is used as a reasonable target for delivery. The equations are derived from $F(T_d)/K = .95$ Another percentage would result in slightly different constants in the equations.

Example: Method 3

- Assume that:
  - similar projects have had a defect density of 10.53 defects per 1KLOC
  - the predicted size is 100KLOC.
  - you expect to have 5% fewer defects (process improvements!).
  - therefore, you project that the total number of defects for this project will = 10.53*100*.95 = 1000.
- Then
  - Since $f(t) = (6\ K/\ (T_d^2))*t*e\ ^(-3(t/\ T_d\ )^2)$
  - And given T_d=26 weeks
  - Then
    - f(t) = 6000/676 * t * e ^ (-3/676* (t^2))
    - f(t) = 8.76 * t * e ^ (-.00444*t^2)

Example Continued - Graphically

- The expected distribution of faults would be:



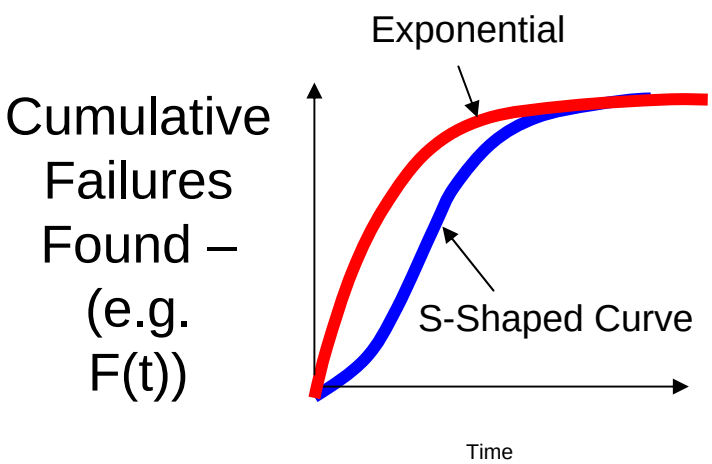
- Based upon the distributions in your data from prior projects, you can add in control limits. You may want to use 2SD which will give you a 95% confidence range of performance.
- If you do not have data from prior projects, you can still project overall defect densities in a number of ways, as discussed later in this lecture, and use the same technique to project arrival rates.

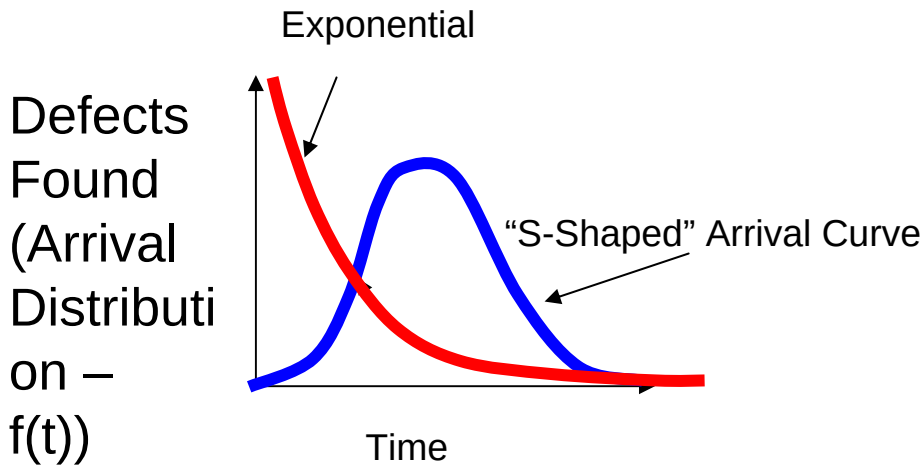
Tools

- The Rayleigh model is used in a number of tools. Some of the more notable ones are:
  - SPSS (Regression Module)
  - SAS
  - SLIM (by Quantitative Software Management)
  - STEER (by IBM).

Exponential and S-Curves Arrival Distribution Models

- Once testing begins, exponential and s-curve functions are the primary modeling functions for defect arrivals.





Using exponential curves & S Curves

- Given a set of data points, you want to know K. The techniques are similar to those for Rayleigh curves. You can:
- Use Reliability/Statistical Analysis Tools
- Solve for a few points
- (Eyeball in your own K…but don't tell anyone I said that).

Empirical Data for the Rayleigh Models - 1

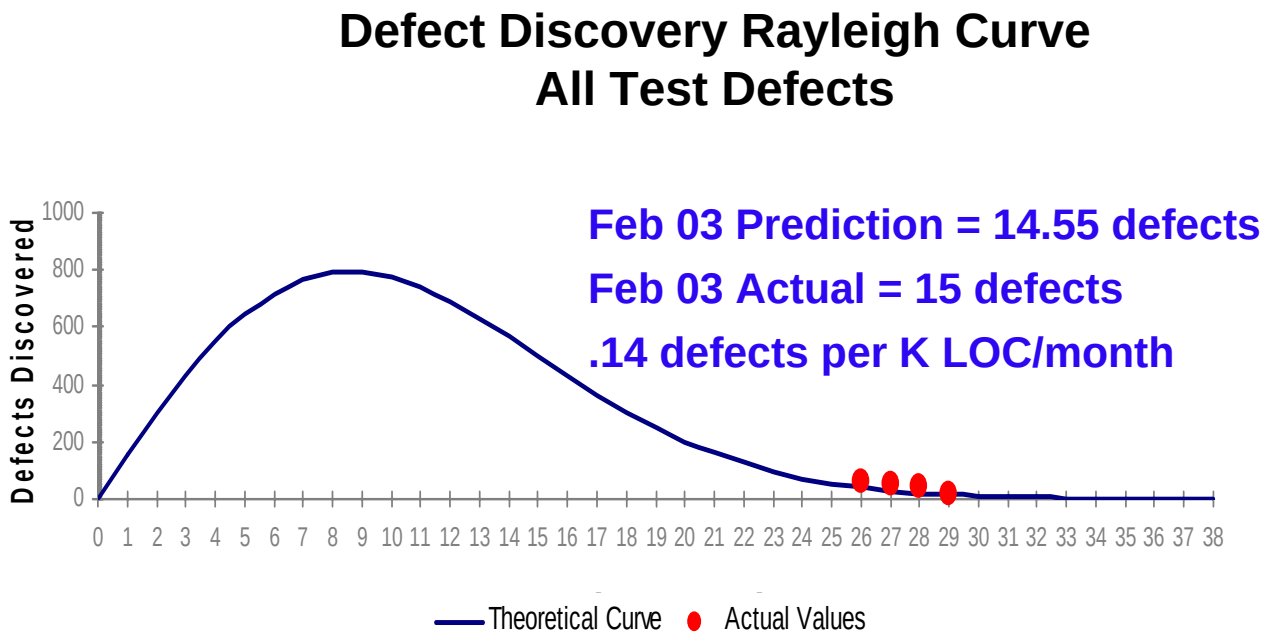
- Putnam and Myers (1992) - total defects projected using Rayleigh curves within 5% to 10%. (!!)
- Biswas and Thangarajan (2000) from Tata Elxsi Ltd reported on using a Rayleigh model successfully on over 20 projects
- IBM Federal Systems in Gaithersburg, MD using their STEER software tool estimated latent defects for 8 projects and compared the estimate with actual data collected for the first year in the field. The results were "very close."
- With small projects, there are fewer data points, and therefore, less confidence.
- Some suggest Weibull curves with m=1.8 as the best fit, but the manual calculations become more difficult

Ref: Thangarajan & Biswas, "Mathematical Model for Defect Prediction across Software Development Lifecycle"

Ref: Hollenback, "Quantitatively Measured Process Improvements at Northrop Grumman IT," 2001

Rayleigh Results -2

- Below are charts from Northop Grumman in 2001, which show their successful use of the Rayleigh Model to predict test defects from defects found earlier in the development cycle.



Recommendations for Dynamic Models

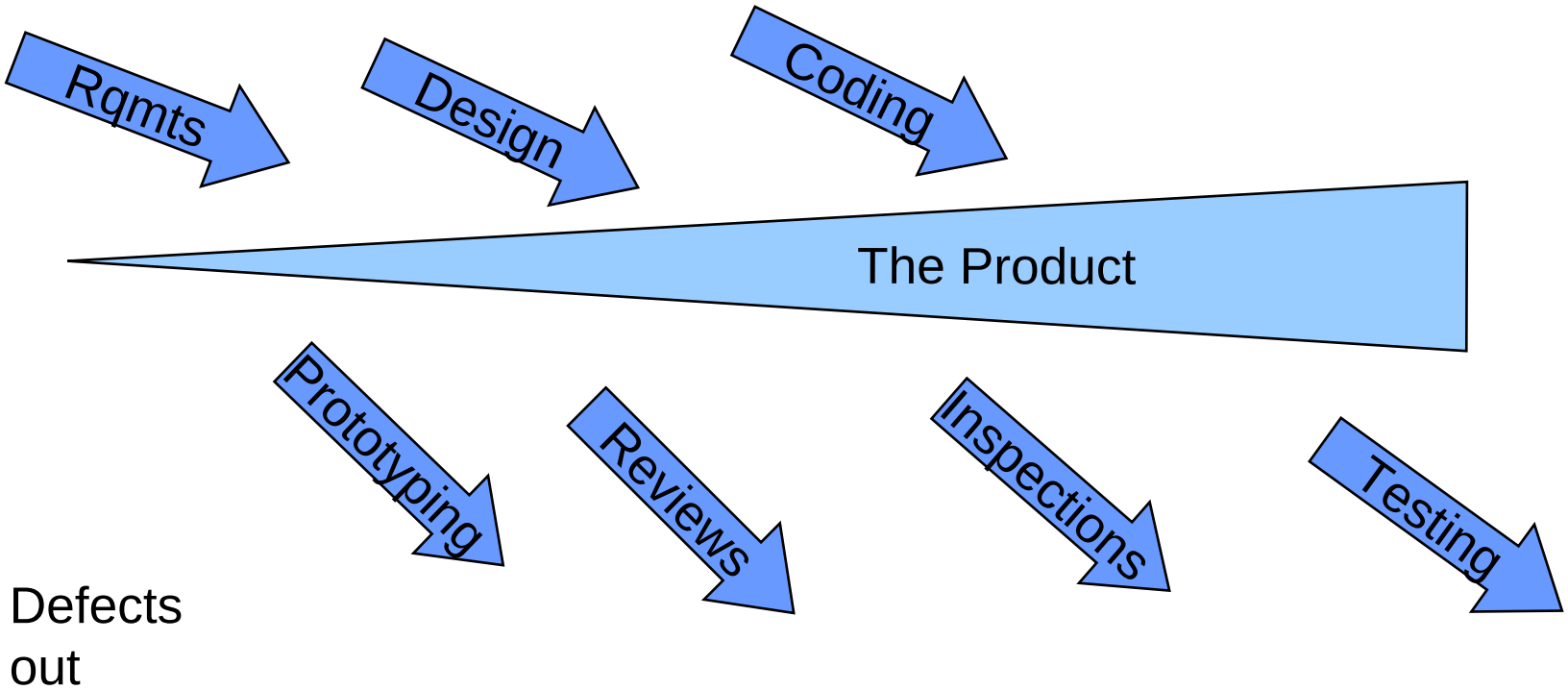
- Over the past 15 years, and with the move to higher SEI levels, much more defect data has been recorded than ever before.
- Even so, organizations will vary from the standard patterns. Defect arrival rates tend to follow the Rayleigh curves, and the equations can be finely tuned to match an individual environment.
- The first and primary recommendation is to start tracking and using your defect data if you are not doing it.
- The second recommendation is to use as many models as possible, compare them with each other, track the results, and see what works best for you.

Static Defect Models

- Based upon software development process and past history – not defect data for this project
  - Can use extremely early in the dev process
  - Useful for evaluating and predicting the impact of process changes on quality

Static Defect Models

Fundamental Defect Introduction and Removal Model – Boehm and Jones



Defect Model Continued

- So, based on this model, the goal in software development, for delivering the fewest defects, is to:
  - Minimize the number of defects that go in
  - Maximize the number of defects that are removed

- Defect Removal Efficiency
  - A key metric for measuring and benchmarking the defects removed from a product

Defect Removal Efficiency

- Both a project and process metric – can measure effectiveness of Quality activities or the quality of a project
- DRE = E/(E+D)
  - Where E is the number of errors found before delivery to the end user, and D is the number of errors found after delivery.
  - The Goal is to have DRE approach 1
- Or $DRE_i = E_i / (E_i + E_{i+1})$
  - Where E_i is the number of errors found in a software engineering activity i, and E_{i+1} is the number of errors that were traceable to errors that were not discovered in software engineering activity I.
  - The goal is to have this approach 1 as well…e.g., errors are filtered out before they reach the next activity

Source: Software Engineering, Pressman, 200, McGraw-Hill

DRE Examples:

- You found 10 defects after you delivered your product. There were 990 defects found and removed before delivery. Your DRE = 990/1000 = 99%.
- You found 10 requirements defects when you were in coding and testing. Luckily, you found none after that, and you hope you got them all. During your requirements process, you found 40 requirements defects. What is your DRE for the requirements process? 40/50 = 80%.

Using the DRE as a Static Model

- Projects that use the same team and the same development processes can reasonably expect that the DRE from one project to the next are similar.
- For example, if on the previous project, you removed 80% of the possible requirements defects using inspections, then you can expect to remove ~80% on the next project.
- Or if you know that your historical data shows that you typically remove 90% before shipment, and for this project, you've used the same process, met the same kind of release criteria, and have found 400 defects so far, then there probably are ~50 defects that you will find after you release.

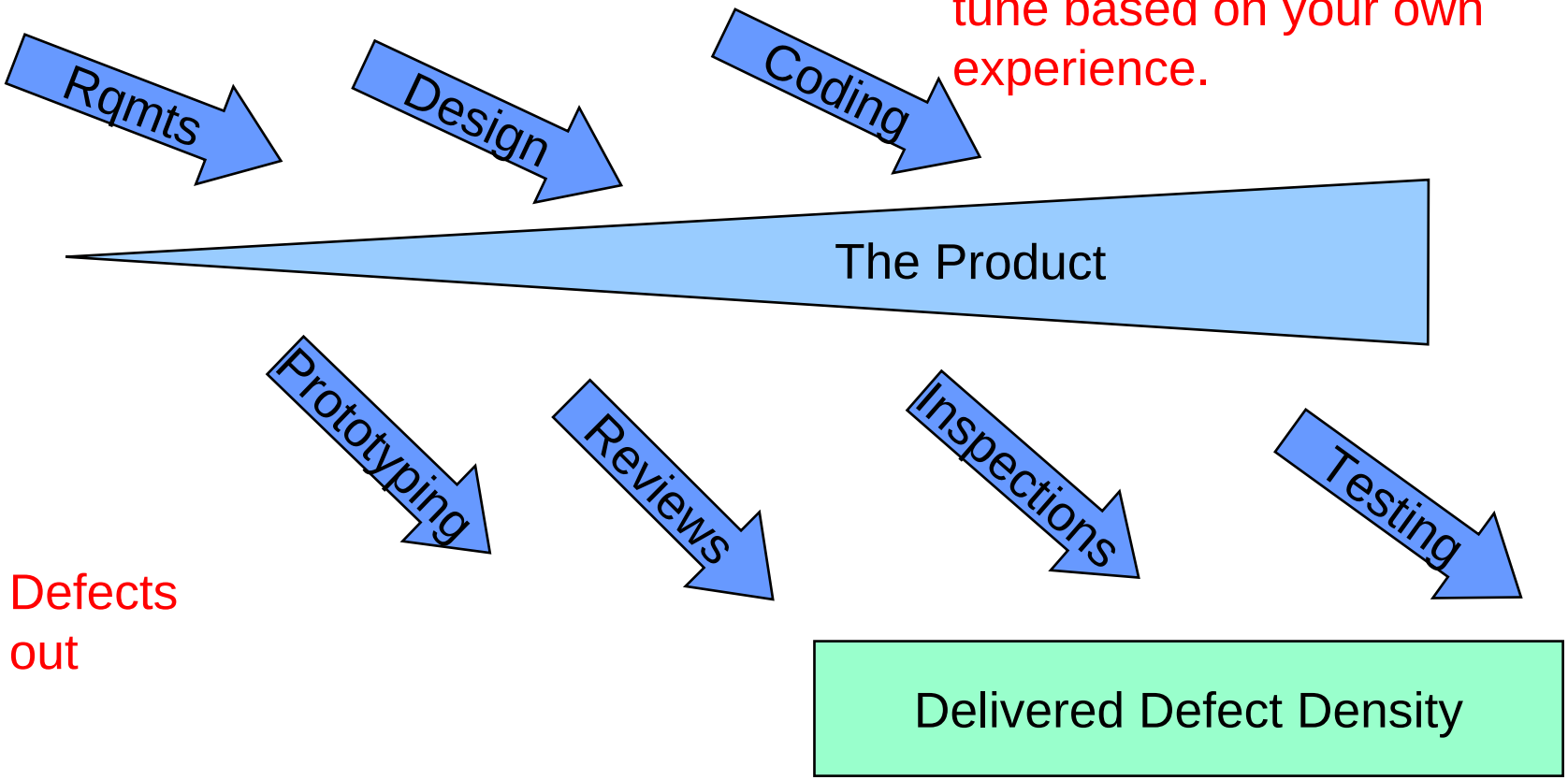
Static Defect Model Tools

- Variety of static defect model tools available, some commercially, some freeware
- Tools allow tuning for your environment
- Will look at one tool, COQUALMO, which was a research project and an extension to COCOMO-II.

Coqualmo is a model which predicts…Delivered Defect Density (per KLOC or per FP)

Defects In

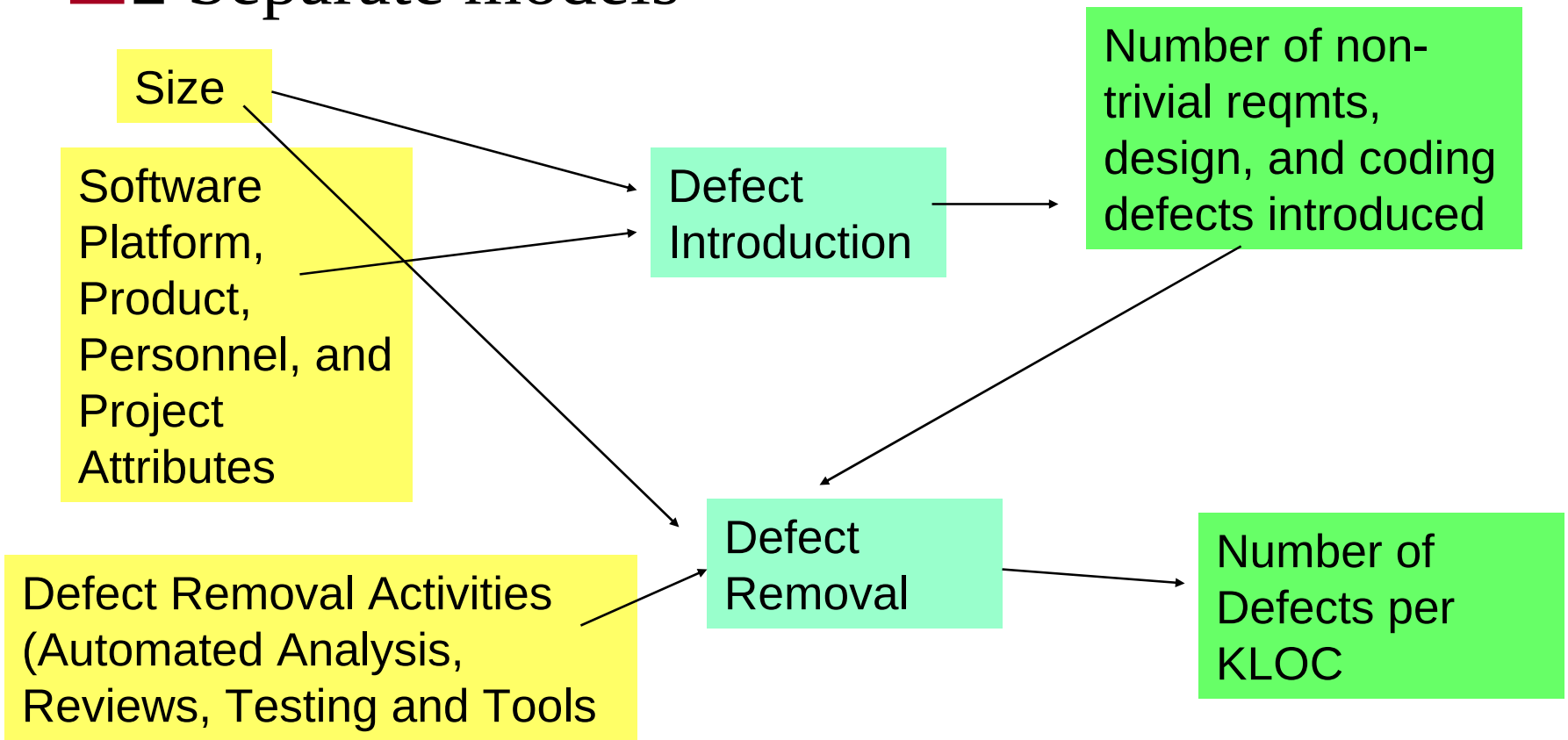
Based upon a variety of factors…And which you can tune based on your own experience.



Defects out

Coqualmo Models

- 2 Separate models



Source: COCOMO II

COQUALMO – More detail

- Quantitative model for defect introduction and removal
  - Chulani and Boehm at USC – 1999
  - Consistent with COCOMO model by Boehm
  - Current data is from the COCOMO clients and "Expert Opinion" – can tune it to your environment

- Defects Introduced (DI) = $\sum_{j=1}^{3} A_j * (Size)^{B_j} * QAF_j$

  Where A is the multiplicative constant (for rqmts, design, coding)

  B is initially set to 1 and accounts for economies of scale

  QAF is the quality factor that is taking into account 21 defect introduction factors (Platform, Product, Personnel, and Project)

DI equation – in English

- What does that equation say?
  - That the number of defects introduced is the sum of the number of defects introduced in each requirements, design, and coding
  - The number of defects introduced in a given phase = A * (size) ^{B} * QAF where
    - A is based upon which phase (typically 10, 20, and 30)
    - B is based upon size
    - QAF is based upon the quality of the process, platform, etc.

Defects Introduced

- Nominal values, per KSLOC are:
  - DI(requirements) = 10;
  - DI (design) = 20
  - DI (coding) = 30
  - DI(total) = 60
- E.G., for for every 1K lines of code, the model predicts that, assuming a “nominal situation” there would typically be 60 defects injected into the code, 10 of which were requirements defects, 20 were design, 30 which were coding
- Process Maturity factor had highest impact on defect introduction…with everything else held constant, it varies result by a factor of 2.5…which says that if you have a very good process, you significantly reduce the number of defects introduced

Defect Removal

- Initial values determined by experts using the 2-Delphi technique
- Looked at three different removal techniques: Automated Analysis, People Reviews, Execution Testing and Tools
- Rated %DRE for removing defects for 6 levels of effectiveness of technique for each phase (rqmts, design, coding)
- Computed residual defects as
  - If all techniques “Very Low Effectiveness” → = 60 defects per KSLOC
  - If all techniques “Extra High Effectiveness” → = 1.57 defects per KSLOC
  - If all techniques “Nominal” → = 14.3 defects per KSLOC

Summary on COQUALMO model

- Mathematical model which takes as input
  - Your view of your defect injection drivers
  - Your view of your defect removal drivers
- Gives you a projection of # of defects remaining in your system
- Can be used to estimate impact of “driver changes” on defect density
  - “what if” analysis
  - “improvement investment” analysis

- The COQUALMO model is an excellent model which your organization can use to engineer and the tune is defect prevention, injection, and removal processes.

Defect Benchmark Data

- Data is surprisingly sparse.
- Many companies hold it very proprietary.
- Easy to misuse the data,
  - punishing projects that somehow are not meeting certain benchmarks
  - misleading results caused by differences in counting techniques, such as in size, usage, and severity.
- Don Reifer[i] has taken a lead in publishing software productivity, cost, and quality data, in hopes of encouraging others to do so.

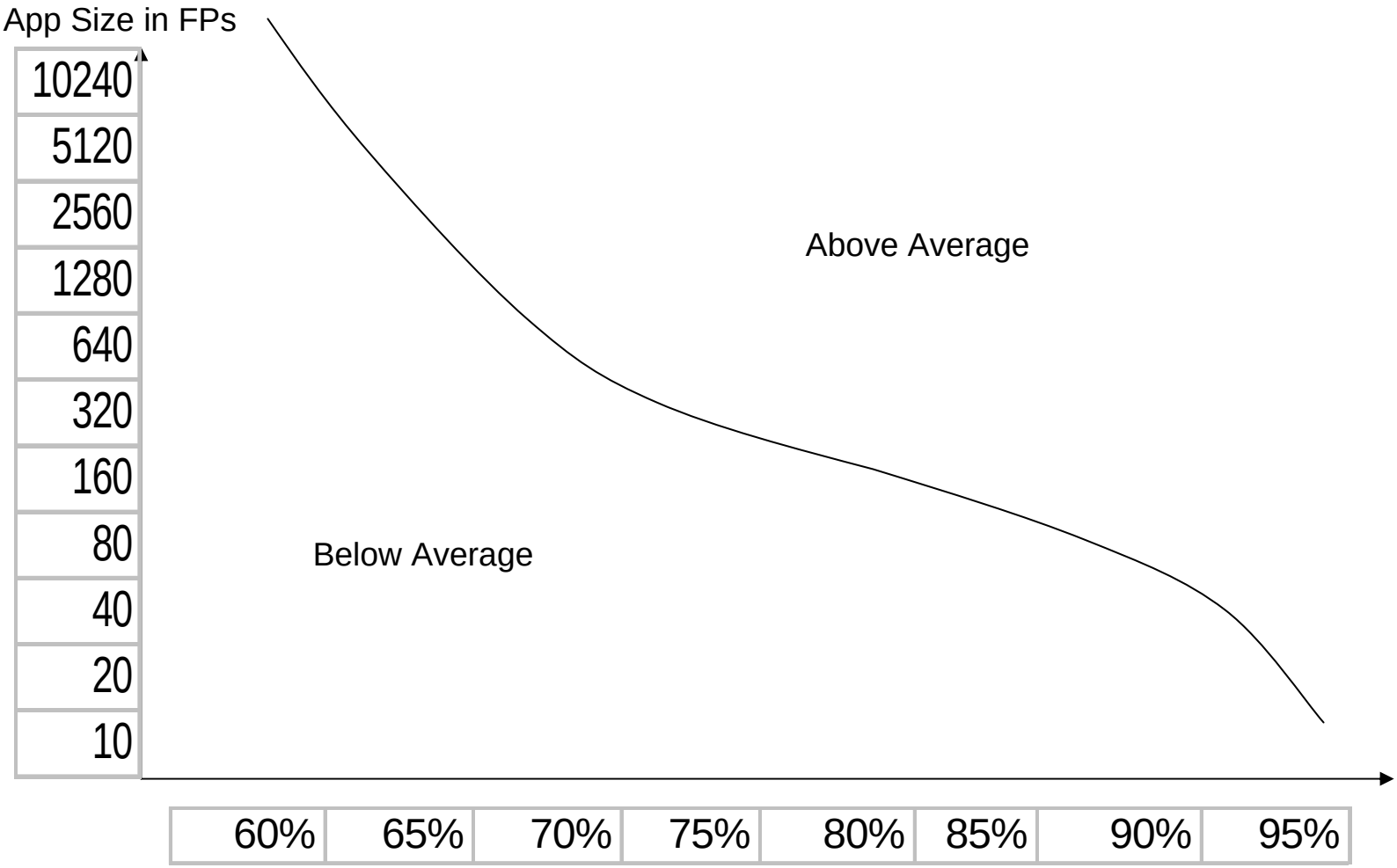
[i] Reifer, Don, The DoD Software Tech News, "Software Cost, Quality, & Productivity Benchmarks", July 2004

Application Domain	Projects	Error Range (per KESLOC)	Error Rate	Notes
Automation	55	2 to 8	5	Factory automation
Banking	30	3 to 10	6	Loan processing, ATM
Command & Control	45	0.5 to 5	1	Command centers
Data Processing	35	2 to 14	8	DB-intensive systems
Environment/ Tools	75	5 to 12	8	CASE, compilers, etc.
Military -All	125	0.2 to 3	< 1.0	See subcategories
Airborne	40	0.2 to 1.3	0.5	Embedded sensors
Ground	52	0.5 to 4	0.8	Combat center
Missile	15	0.3 to 1.5	0.5	GNC system
Space	18	0.2 to 0.8	0.4	Attitude control system
Scientific	35	0.9 to 5	2	Seismic processing
Telecom	50	3 to 12	6	Digital switches
Test	35	3 to 15	7	Test equipment, devices
Trainers/ Simulations	25	2 to 11	6	Virtual reality simulator
Web Business	65	4 to 18	11	Client/server sites
Other	25	2 to 15	7	All others

Domain Data Comments

- Defect rates in military systems are much smaller due to the safety requirements
- Defect rates after delivery tend to be cyclical with each version released. They initially are high, and then stabilize around 1 to 2 defects per KLOC in systems with longer lifecycles (> 5 years). Web Business systems tend to have shorter lifecycles (<=2 years) and may never hit the stabilization point.

Cumulative Defect Removal (Reviews, Inspection,Testing)



Source: Jones, 1991

SEI Defect Removal – General Dynamics Corp.

SEI Level	Delivered Defect per KLOC	% Defects Removed Before Ship
2	3.2	25.5
3	0.9	41.5
4	0.22	62.3
5	0.19	87.3

Diaz & King, 2002 (in Kan)

Latent Defects

- Two different studies on millions of lines of debugged and released C and Fortran code by Hatton and Roberts using only static code analyzers found ~6 defects per KLOC. These were faults such as uninitialized variables. Obviously, there are more latent defects in the code than the static analyzers found.

Ref Hatton & Roberts, from Fenton and Pflegger

Defects Table of Contents

- Faults and Failures
- Defect Dynamics and Behaviors
- Defect Projection Techniques and Models
  - Dynamic Models
  - Static Models
    - Defect Removal Efficiency
    - Coqualmo
- Defect Benchmarks