Industrial Strength Software Measurement

Audris Mockus, David Weiss
{audris,weiss}@avaya.com

Topics

- Why measure?
  - On industrial scale?
  - On project scale?
  - On individual scale?
  - On country scale?
- The GQM model for measurement?
    - Goals, Questions, Measures
    - Evolution of goals
      - The cost, quality, time to market rotation
  - Characteristics of industrial measurement
- Some of our goals
- Available data
- Some examples
  - Interval Quality
  - Registration Refactoring
  - Introduction of Test Coverage Tools

Measurement Approach: GQM

- Identify goals of software development process
  - Example: Produce more new features, fewer defects with fewer, more distributed, resources.
- Propose questions whose answers establish progress towards goals
  - Example: What is the ratio of new features to bug fixes by product? By site?
- Define measures that can be used to answer questions and that can be practically obtained for the software project
  - Example: Ratio of new feature MRs to bug fix MRs by product and site, normalized.
- Validate measures internally and externally
  - Example: remove tool generated artifacts and ensure the measure represents the phenomena it is intended to measure
- Establish infrastructure for data collection and analysis
  - Dashboards
  - Automated data collection and analysis

Software Changes: A Fabric of Measurement

- MR = Modification Request
  - For every change
    - Why was it made?
    - Who requested it?
    - Who made the change?
    - What was changed?
    - When was it changed?

      ….
- States of an MR
  - Created (Developer, Tester, Support)
  - Assigned (MR Review Board)
  - Submitted (Developer)
  - Verified (Tester)
  - Completed (MR Review Board)
  - Accepted

Background

- **Software is created incrementally, via changes recorded by a VCS**
- **A delta is an addition and deletion of lines in a file**

before:

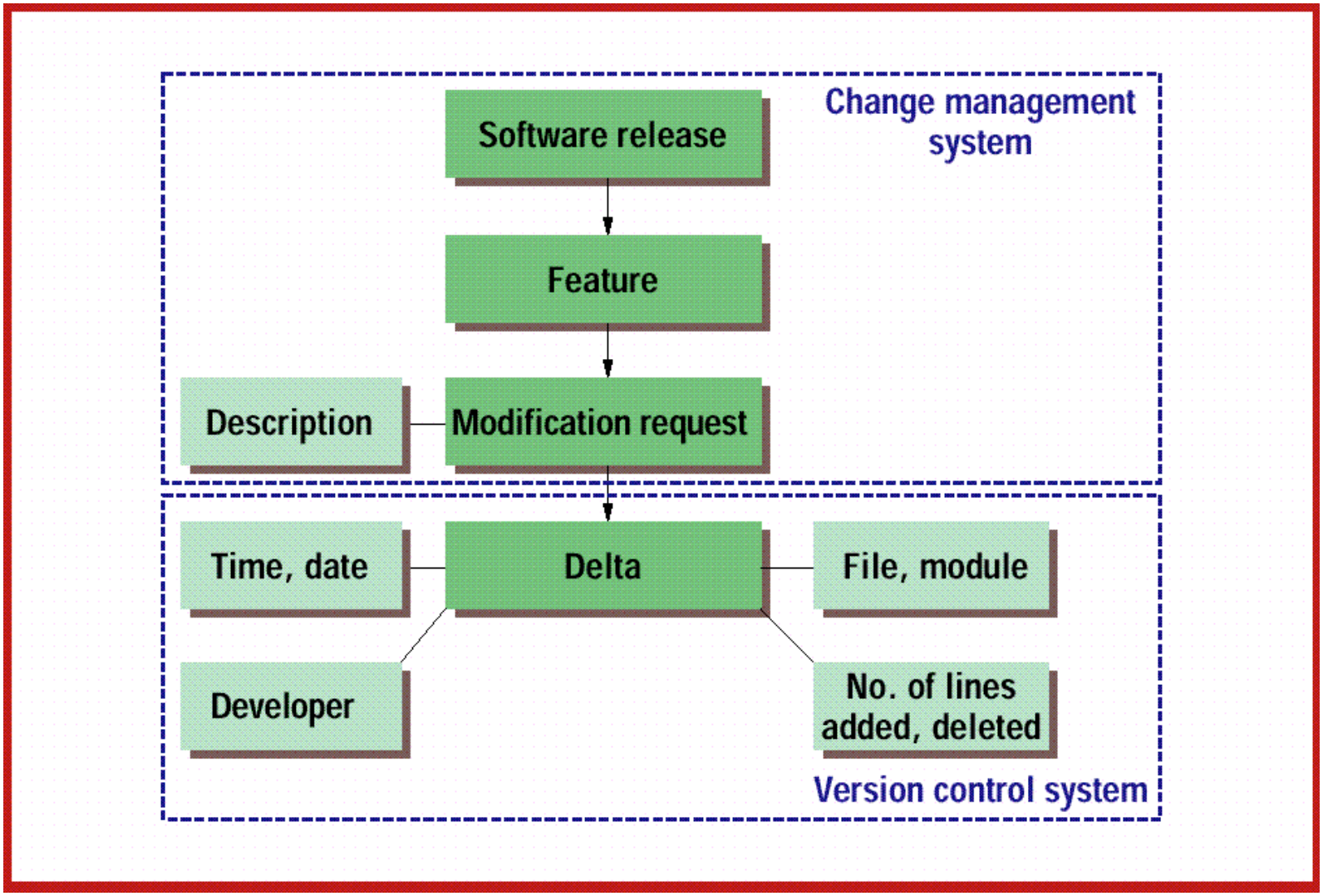
```
int i=N;
while (i)
    printf (“%d\n”,i--);
```

after:

```
// print N integers
    int i=N;
while (N > 0 && i > 0)
    printf (“%d\n”,i--);
```

  - **one line deleted**
  - **two lines added**
  - **two lines unchanged**

Change Hierarchy



Characteristics of Industrial Measurement

- Meaningful
  - Show progress towards meeting goals
    - Trends, snapshots, figures of merit
- Nonintrusive
  - Don't add to developers' burden
  - Use data (already) collected for development purposes
- Automatable
  - Handle large amounts of data over long periods of time
    - 10s of thousands of records over decades
  - Automatically produce dashboards (website)
- Customizable
  - Each project can customize for its own version of goals
- Feasible
  - Data can be collected in an automated way
  - Verification possible

Some Key Feasible Measures

- Diffusion (# of subsystems, modules, files, developers)
- Size (# of lines added, deleted, and in the touched files)
- Diffusion & Size (# of deltas, MRs)
- Lead time (interval from start to completion)
- Purpose (Fix/New)
- Identity and experience (# of delta done in the past/recently/on a relevant part of the product) of creators

Some Benefits of Change Measures

- **Availability and cost**
  - **obtainable for all projects using CM**
  - **nonintrusive – use existing data**
- **Detail and coverage**
  - **fine grained – information at MR/delta level**
  - **complete – all parts of software are recorded**
  - **massive – larger than surveys/project measures**
- **Stability and bias**
  - **uniform – slowly change over time**
  - **unbiased – no observer effect**

Some Drawbacks of Change measures

- Require validation and careful interpretation
  - Data recorded for other purposes
  - Often need nontrivial datamining techniques
  - Different project support systems contain different attributes
  - Different projects may use the same system in different ways

Some Current Avaya Goals (1)

- **Significantly improve predictability**
  - **Is predictability improving?**
  - **What fraction of projects are on time?**
  - **What are the factors associated with late projects?**
- **Significantly improve quality**
  - **Is quality improving?**
  - **What is the customers' perceptions of software quality?**
  - **What is the in-process quality?**
- **Rapidly produce new products (days and weeks instead of months and years)**
  - **Use a modular, family architecture**
  - **Take advantage of commonalities to compose and generate rather than hand code**
  - **Make production predictable**
  - **Continually predict, trial, and leverage expected future needs**
  - **Develop infrastructure for composing products from modules**

Some Current Avaya Goals (2)

- Keep production within limits of resources, which are becoming more distributed
  - How distributed are resources? What's the trend?
  - Are there differences in productivity, quality among sites?
- Make globally distributed development (independent component development at different sites) an advantage
  - Are there differences in productivity, quality among sites?
- Introduce new software development processes
  - Agile development using automated test tools
- Ensure minimum of 60% test coverage for all new code

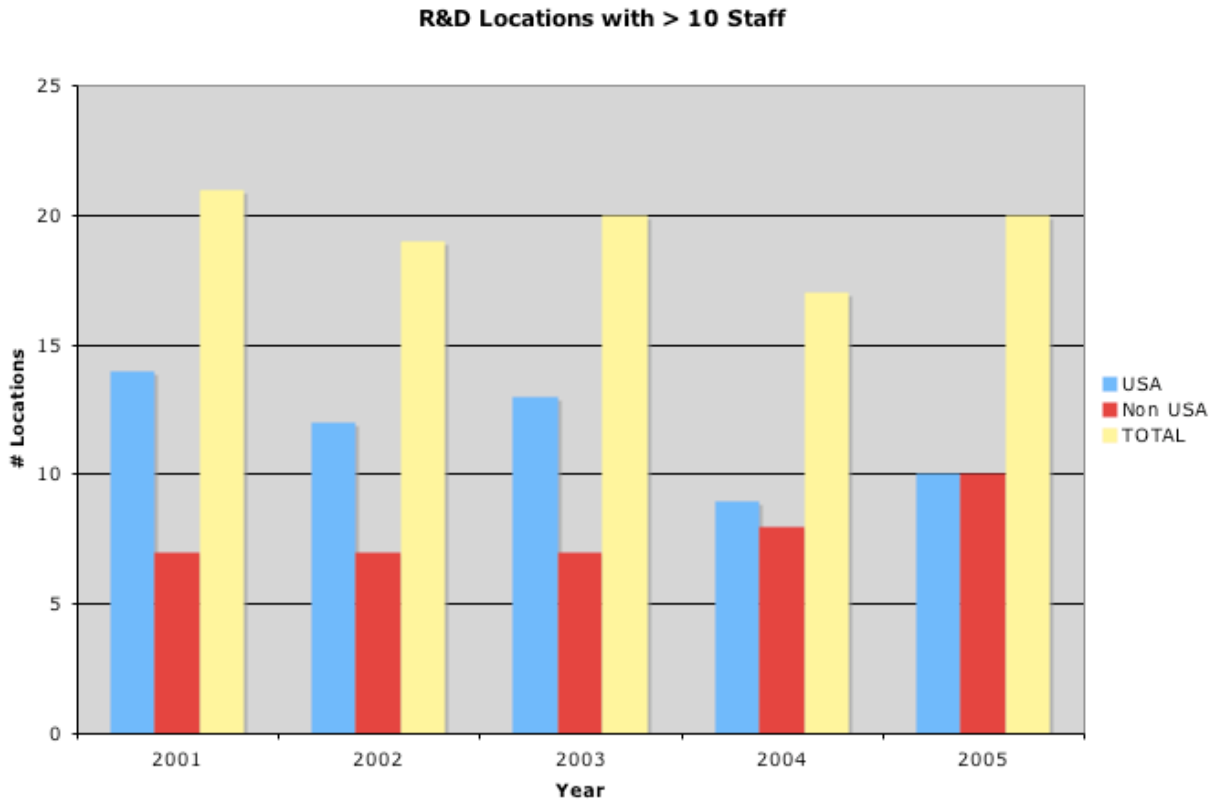
(A Few) Proposed Questions

1. What is the time and effort to create a new product? How predictable is product creation (time, effort, resources)?
   - Snapshot and trends
- What is the ratio of new modules to reused modules in a product?
  - Snapshot and trends
- For each (new?) product, which modules are new, which are reused unchanged, and which are reused wit adaptation or configuration?
- What is the ratio of new features to bug fixes by product? By site?
  1. Snapshot and trends
- What is the time and effort to create a new version of a new module? By site? How predictable is module creation (time, effort, resources)?
  - Snapshot and trends
- Is the architecture modular? Are interfaces suitable for use in many products and well-defined?
- Does the architecture match the organization (one site per module)?
- Is iterative development possible?
- What is the quality of products? What is the quality of modules?
  - Snapshot and trends

Plan

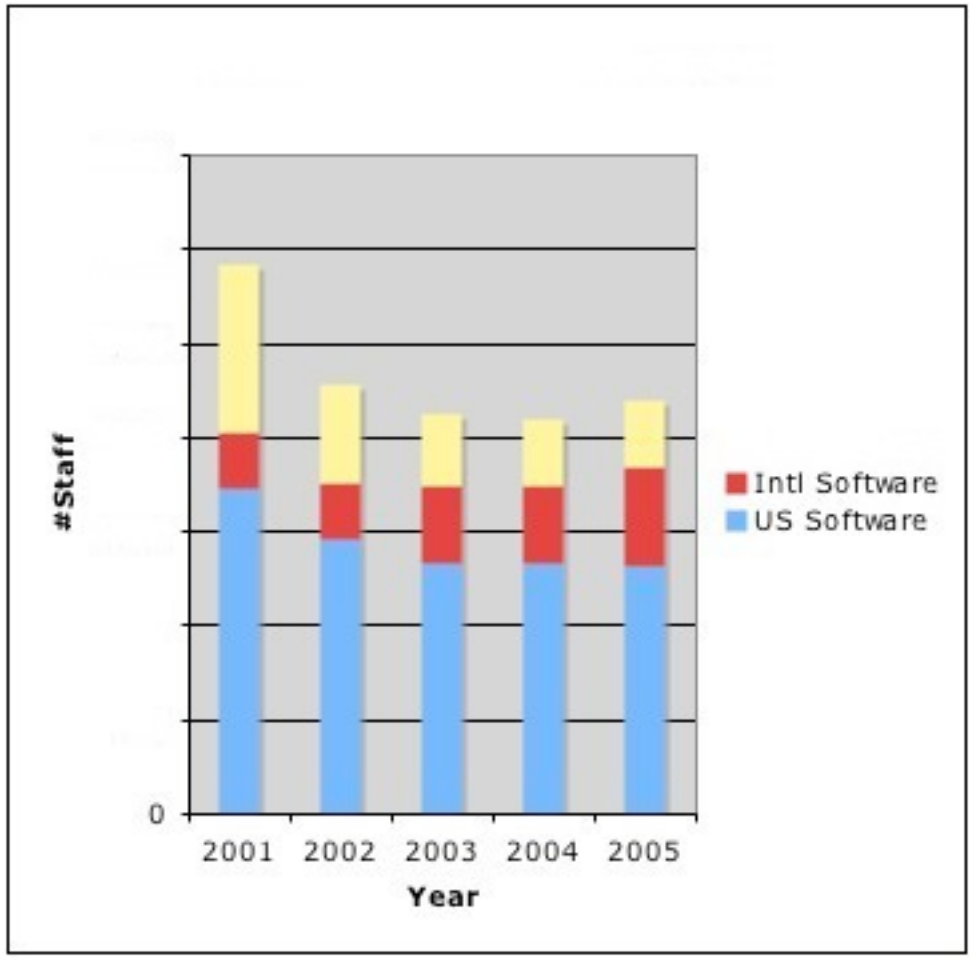
- Iterate on goals, questions
- Define data collection needs and resources
  - Who is responsible for assuring (accurate) data are collected?
- Trial data collection and analysis
- Iterate, revise, scale-up: create dashboards

Distribution of Software Development

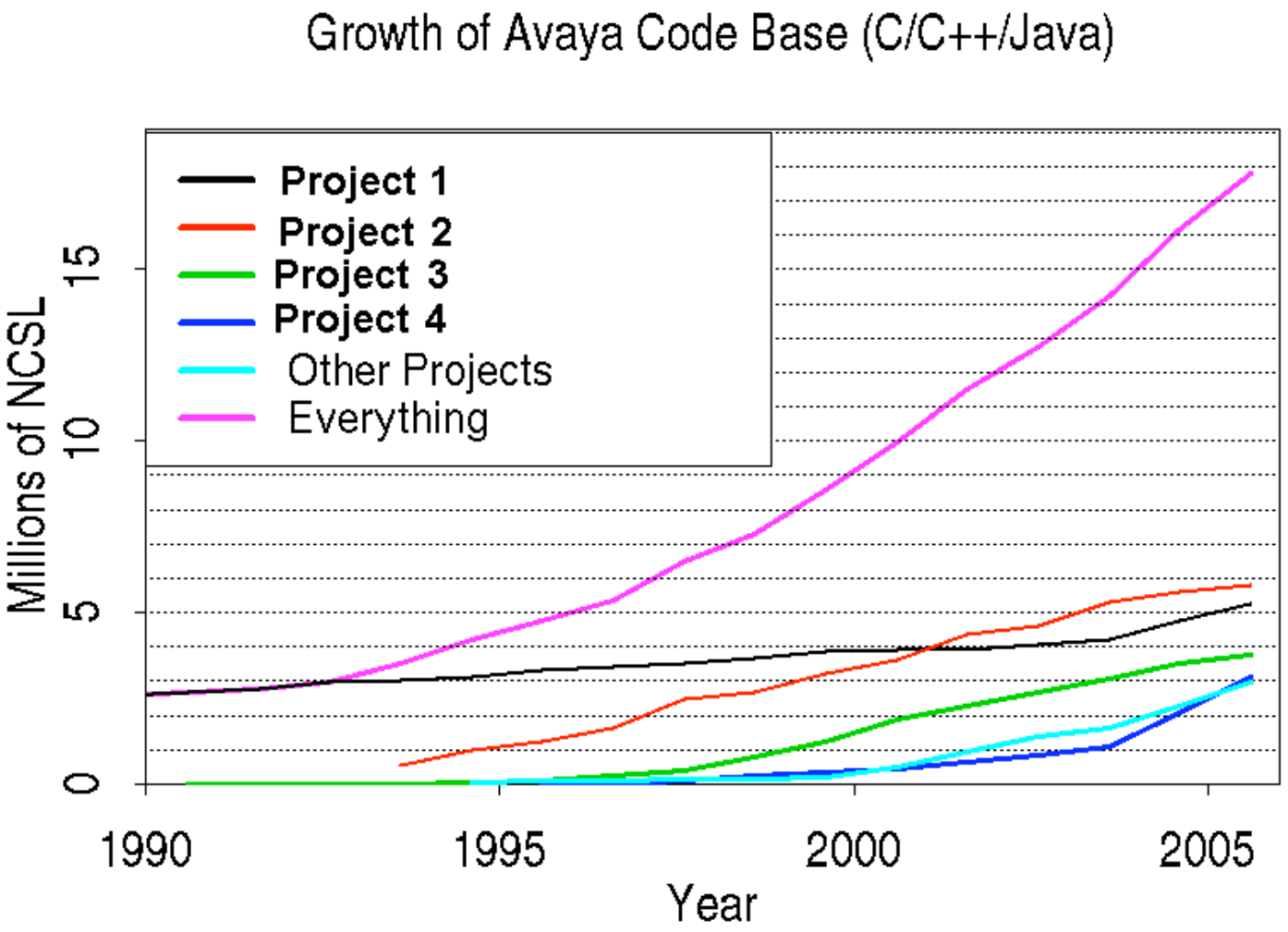


The number of International R&D locations has increased while the number of US locations has decreased between 2001 and 2005.

Resources for Software Development

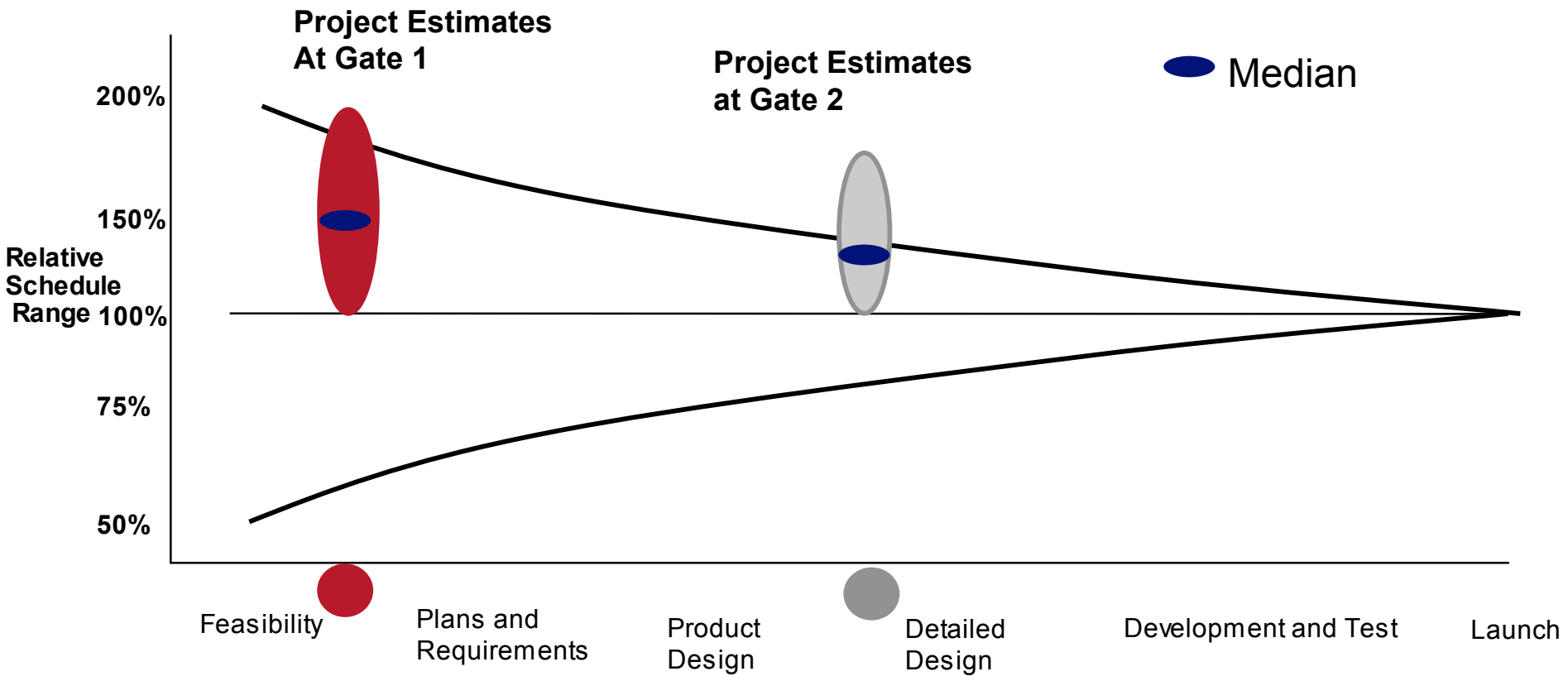


Growth of the Code Base

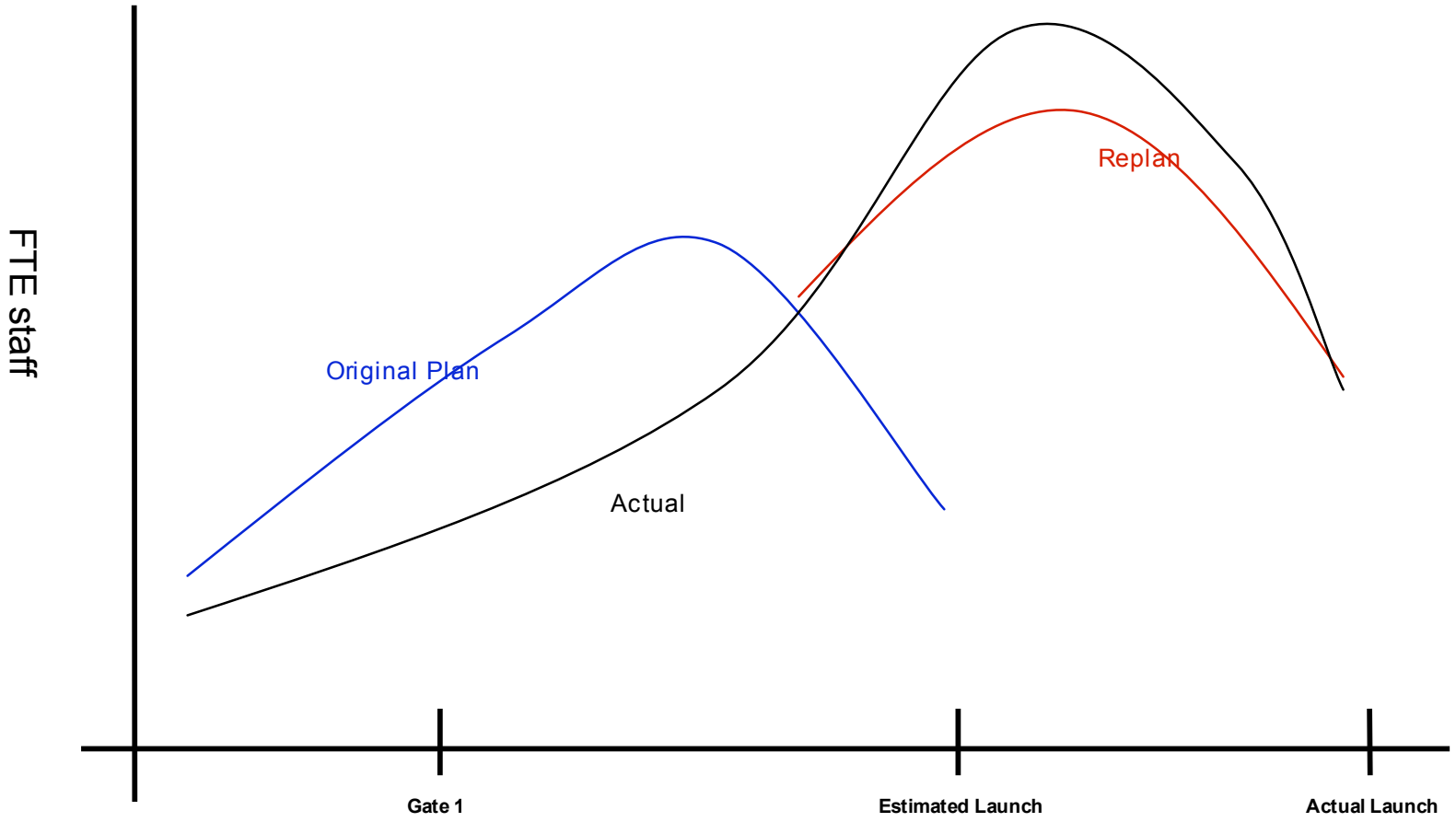


Predictability

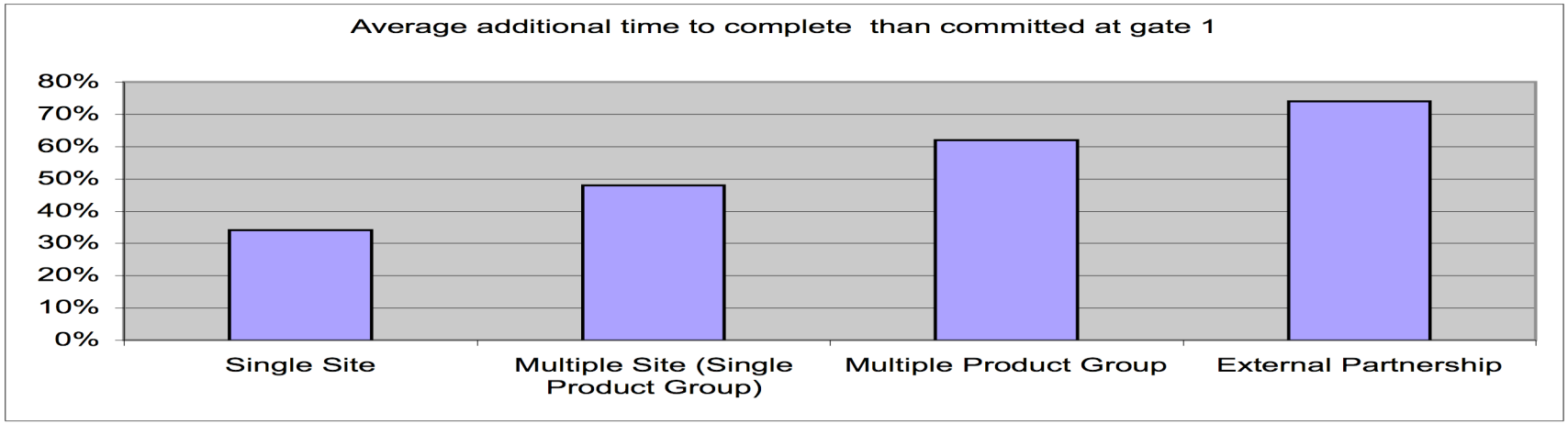
Predicting Software Development (50 sampled projects)

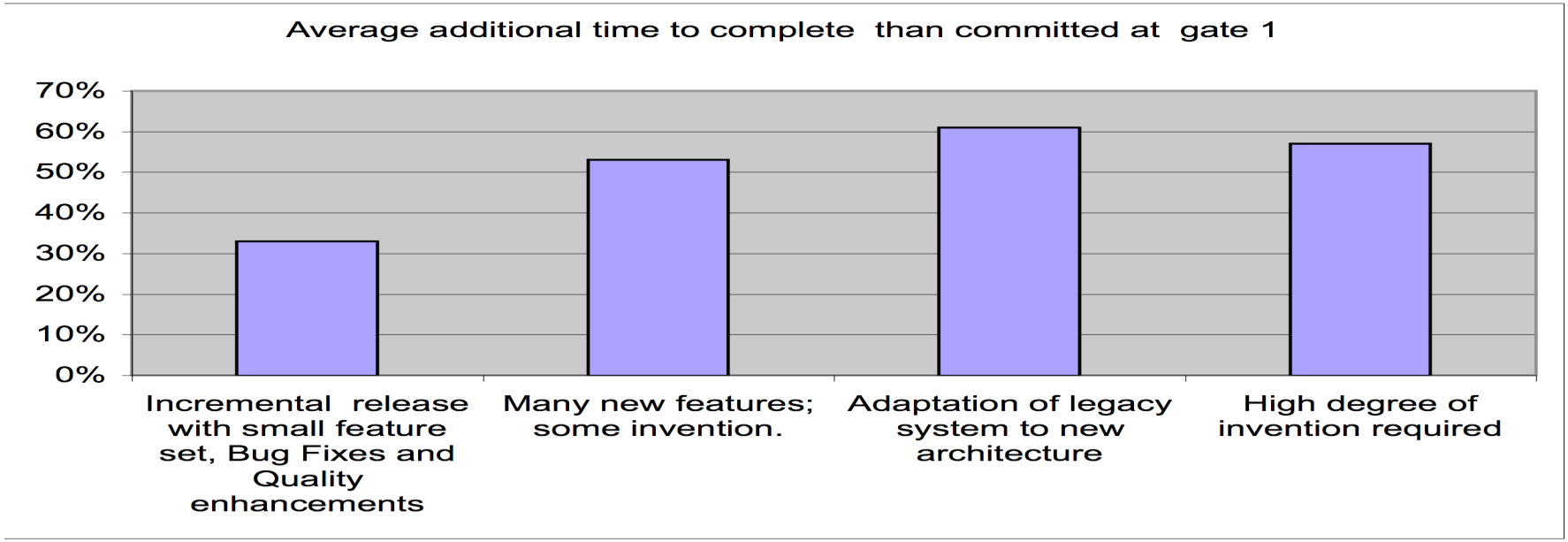


Example Staffing Profile



Distributed development, innovation, new features, legacy adaptation all contribute to delays





Interval Quality

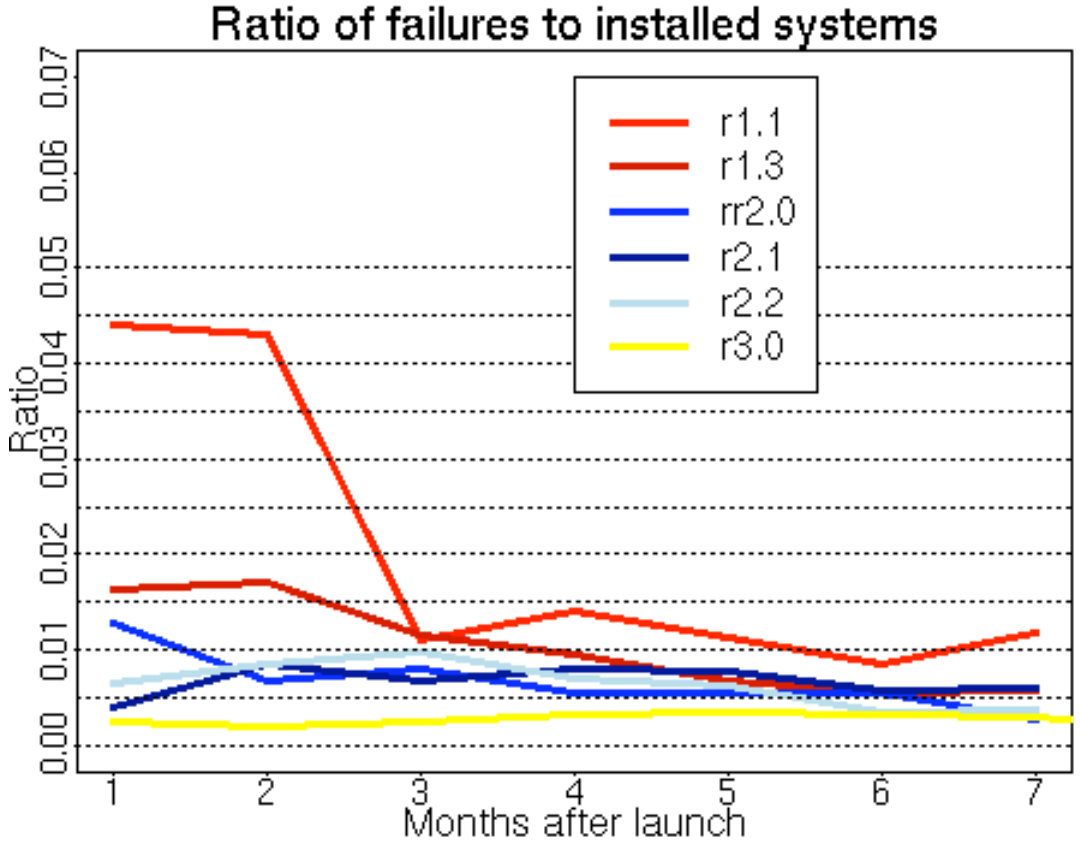
Context of quality measurement

- **Primary question:**
  - **Is the quality (reliability/availability) experienced by customers increasing/decreasing?**
- **Data sources**
  - **Customer inventory**
  - **Service calls, system alarms**
  - **Software changes**
- **Primary challenges**
  - **Storing, cleaning, and linking data sources**
  - **Designing a simple to understand and use quality measure**

The probability of a customer observing a failure

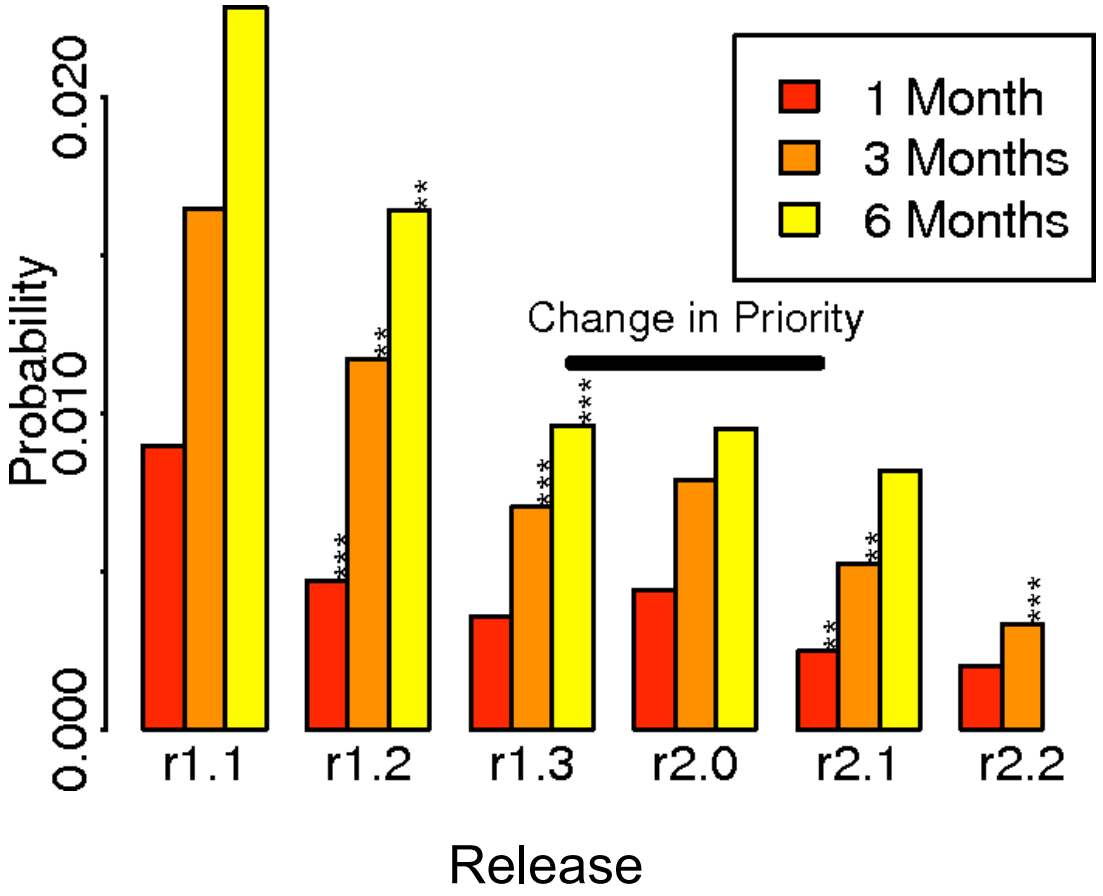
- Is affected by:
  - Major/minor release
  - How soon after launch the system was installed
  - How long the system was running
  - The size and utilization of the system

The graph shows two factors:
- time after launch
- release



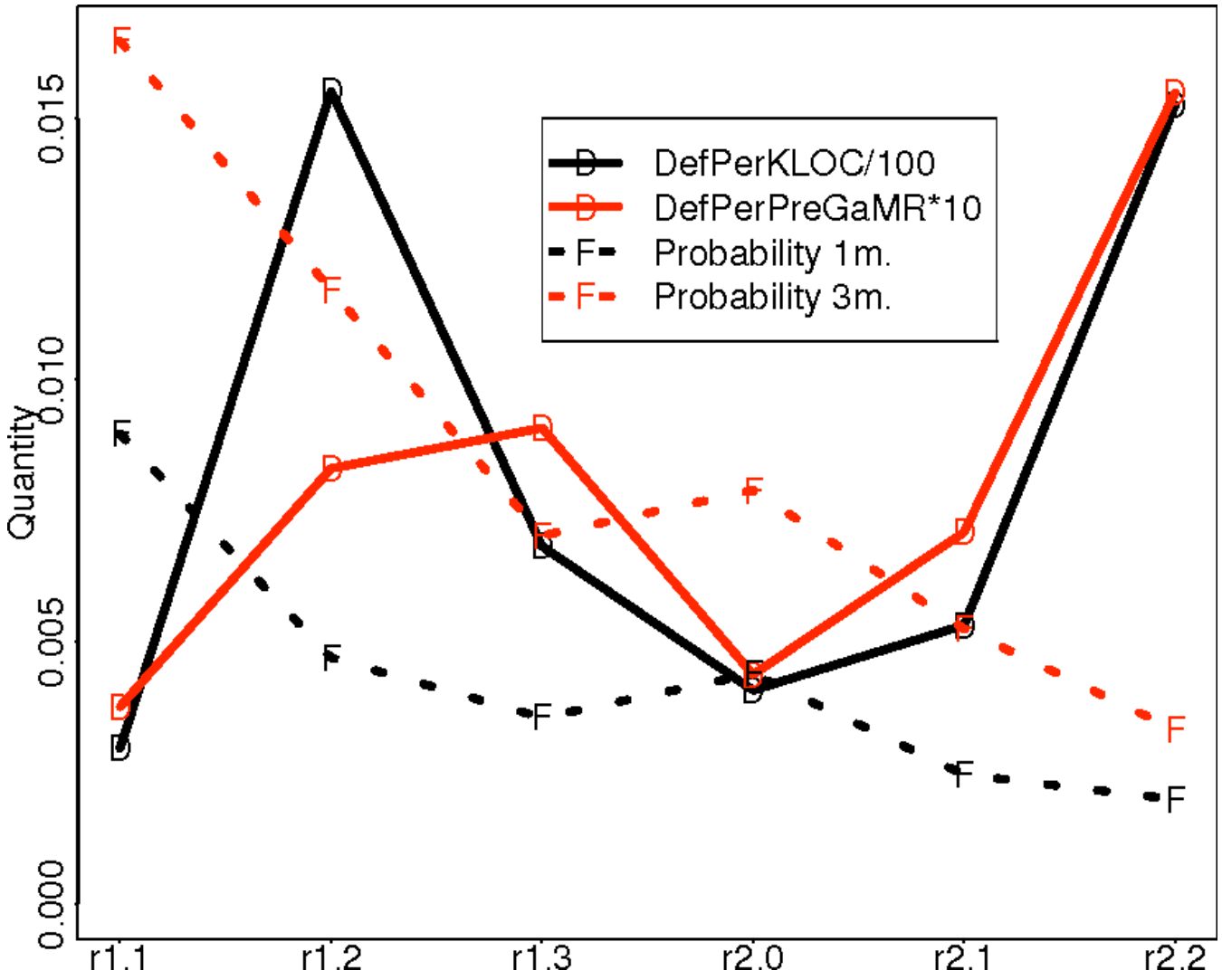
Interval Quality

- **Probability that a customer observes a failure within one, three, and six months after installation**
  - **1 month**
    - **more noisy**
    - **allows seeing trends earlier**
  - **6 months**
    - **more stable**
    - **have to wait for results**
- **Drawback**
  - **does not account for the proximity to launch**
- **Significant differences are marked with *, **, and ***
- **Priorities changed from time-to-market to quality**



Interval Quality and Defect Density

- **X-axis:**
  - **Releases**
- **Y-axis:**
  - **Four measures**
- **Features:**
  - **negative correlation**
  - **major releases look better in terms of defect density**



Introducing New Technology

Context for Refactoring a Telecommunications Domain

- One domain of Avaya's IP telephony software
- 30 KLOC C++, ASN.1 generated code, 3rd party protocol stack within 7 MLOC system
- 40 different developers over 5 years
- Design degradation
- Constant change
  - inflow of defects from 5+ deployed releases
  - changes to implement new functionality for 2+ future releases

Software Refactoring

- For migrating legacy code to a target design
- Improve code structure without changing external behavior
- Sequence of simple behavior preserving code transformation steps
- For instance: “Extract Method”: Turn a code fragment into a method whose name explains the purpose of the method

Refactoring Hypotheses

- **H1: The customer reported defect rate will improve**
  - **Better (“collaboration”-based) design**
  - **Refactoring exposed pre-existing issues**
- **H2: The refactoring reduces the effort required to make changes**
  - **Information hiding**
  - **If design is good changes will be confined**

Measures

- **H1: The number of field MRs found and the root cause of these problems**
- **H2: Change effort and the amount of code that needs to be inspected to make the change**

Defect Density

- The number of defects depends on release size
- Reported defects and submitted changes in registration domain
- Four pre- and one post-refactoring release

	Release Size	Field defects
pre-Refactoring	526	41
post-Refactoring	80	0

- Adjust for the shorter exposure of the last release: assume only 50% of defects in the first 7 months (41/2=20)
- Fisher's exact test p-value 0.06

Change Effort

Stage	#changes	avg(log(PersonMonths))
Pre-Ref.	292	1.12
Post-Ref.	151	1.23

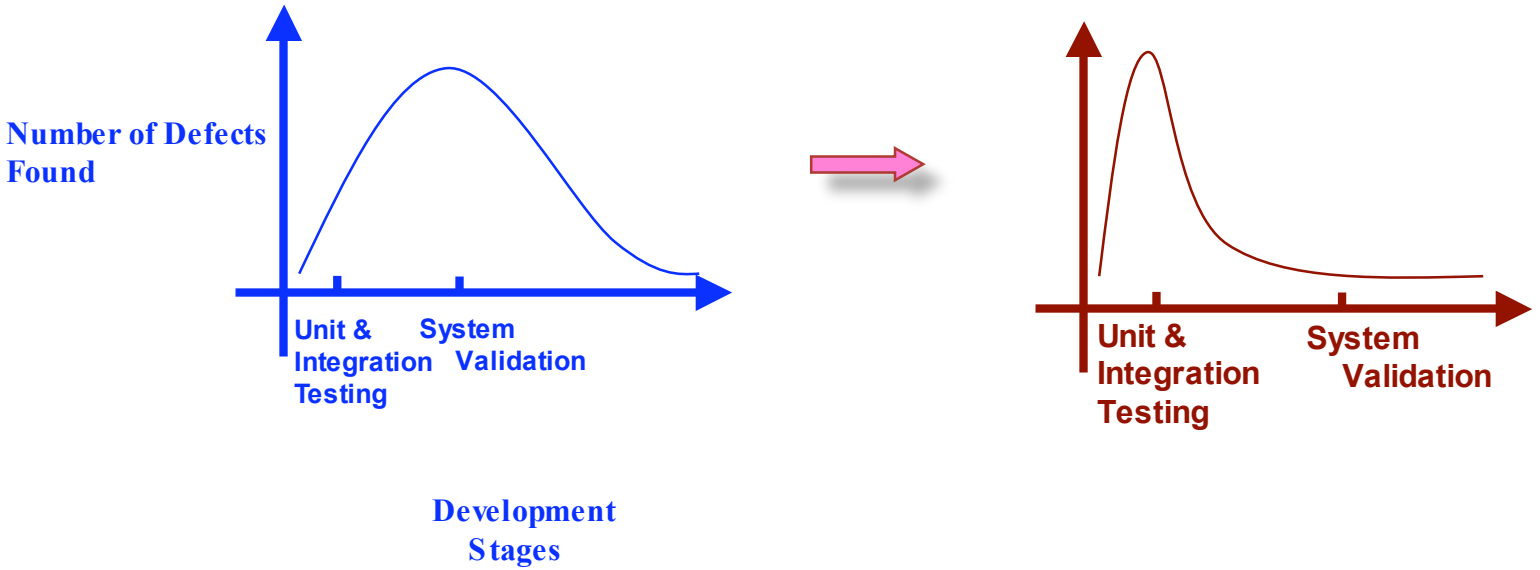
- two-sample t-test of log(effort) p-value=.06
- Mann-Whitney of log(effort) p-value=.06
- The LOC in the refactored area decreased by 50%

Validation

- **Reality**
  - **Verified the process**
  - **Verified selection of relevant changes (MRs)**
  - **Manually inspected all field MRs**
  - **Several operationalizations**
- **Modeling**
  - **Distribution: take logs or use nonparametric tests**
  - **Normalize by size where needed**
  - **Apply relevant models**
- **A case study precludes causal inference**

Automated Test Coverage: Goals, Questions

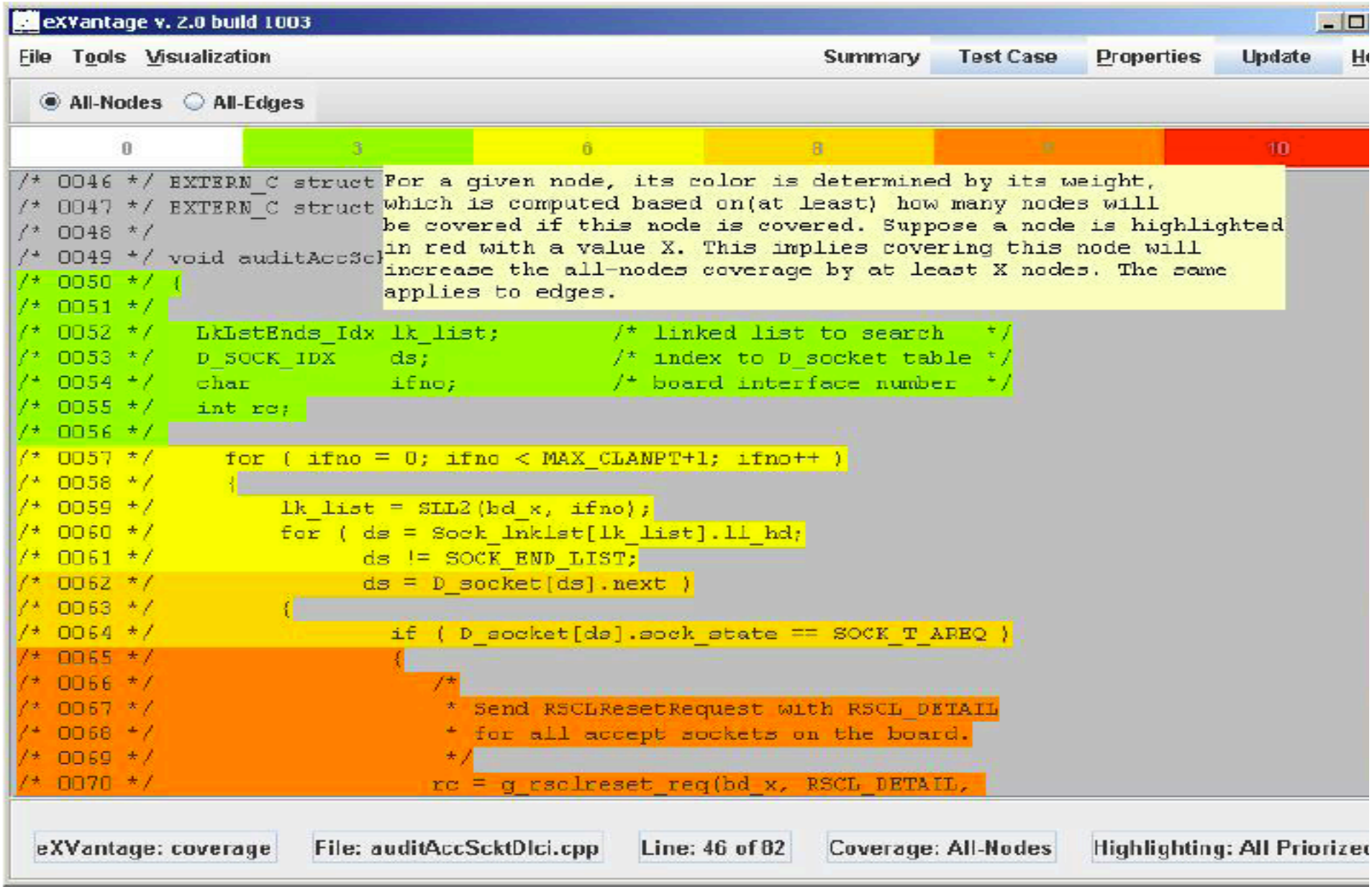
- Estimate impact of introducing new tools, techniques
- Test coverage: Move detection of defects earlier



- Do we see the expected impact?
- What is the effect on effort, quality, schedule?

Coverage Report (Batch)

Source Code View (GUI - Initial)



Automated Test Coverage: Feasibility

- Measured introduction of a test coverage/slicing tool
  - Usage logged: date, IP, login, invocation options
  - Changes to the codebase: login, file, date, size
  - Changes to the test code (JUnit) base
  - MRs: date, origin, developer
- Expected outcome
  - Logins with higher test tool usage have fewer MRs raised in testing and post-launch
- Complications
  - The coverage tool was run as a part of build process to create reports, so it was impossible to determine who used the reports
  - There was limited understanding about potential uses of the tool among developer population, some important functions were not utilized

Summary

- Why measure?
  - Estimate parameters important to business
    - Customer satisfaction, predictability, time and resources needed to create products
  - Evaluate progress on particular projects
    - When will it be ready? How many architects, developers, testers will we need?
  - Estimate capabilities and needs to understand areas for improvement
    - What problems do we need to solve to improve?
    - What is the impact of introducing new technology, methods?
  - Personal, Business, Country, World
- What's a good model for measurement?
  - Define goals first, then ask questions needed to evaluate progress towards achieving goals
    - Goals change over time - interval, quality, cost
  - What are characteristics of industrial measurement?
    - Change data as a key information source
    - Automatibility, nonintrusiveness
- Some examples
  - Interval Quality
  - Registration Refactoring
  - Introduction of Test Coverage Tools