2006

Crafting A Measurement Framework Using A goal-Question-Metric Approach

Panos K. Linos
Butler University, linos@butler.edu

Recommended Citation

"Crafting A Measurement Framework Using A goal-Question-Metric Approach" Indiana Quality Assurance Association (IQAA) Indianapolis July 20, 2006

CeASER

Center for Applied Software Engineering Research

Crafting a Measurement Program Based on a Question-Goal-Metric Approach

Panos Linos
Center For Applied Software Engineering
Department of Computer Science and Software Engineering
Butler University

Email: linos@butler.edu
Web: ceaser.butler.edu

Agenda

- A bit about myself
- A bit about Software Engineering Measurement (SEM) programs and their rate of success*
- A *lot* about the *GQM approach***
- A bit about how to jump-start a SEM program***
- A *lot* about *addressing key issues early* to ensure success
- A bit about tools
- A summary of suggestions to get you started

**Measurement Programs in Software Development: Determinants of Success* by A. Gopal et al, IEEE TSE Vol.28, No. 9, Sept. 2002.

***The Goal-Question-Metric Paradigm* by V. Basili et al, Encyclopedia of Software Engineering, Vol. 1, John Wiley and Sons, 1994.

****Experiences in Implementing Measurement Programs* by Wolfart Goethert and Will Hayes, SEI Technical Report: CMU/SEI-2001-TN-026

Survey

- With your permission, I would like to conduct a brief survey first.
- Collect, summarize and discuss your responses at the end of my talk.
- Sounds good?

Who am I?

- A “9-month academic”
  - Teaching
  - Research
  - Service
- A “3-month practitioner”
  - Consultant
  - Contractor

Examples Software Engineering Measurement (SEM) Programs*

- **Example 1: Establish measurements across a global enterprise (i.e. business units in Tokyo, Singapore, Hong Kong, India, Argentina and USA)**
  - Ability to assess the progress of the overall enterprise
  - Ability to evaluate new technologies, methods and practices by
    - Collecting identical measures to enable meaningful comparisons and trend analysis
    - Create a large pool of projects data from which similar projects can be chosen for comparison purposes
  - Specific business goals articulated by the CTO
    - Increase productivity by a factor of 2 over 5 years
    - Improve quality by a factor of 10 over 7 years
    - Improve predictability to within 5% over 7 years
    - Reduce development time by 40% over 7 years
    - Reduce maintenance effort by 40% over 7 years

**Experiences in Implementing Measurement Programs* by Wolfart Goethert and Will Hayes, SEI Technical Report: CMU/SEI-2001-TN-026

More examples of SEM efforts

- **Example 2: Enterprise performance management, a local perspective**
  - To establish a common basis for comparing information across a widely distributed organization is a major concern.
  - To support management in workload balance and effective project management in the context of an ongoing SPI program.
  - To enforce standardization of measurement across the organization.
  - To relate the performance of small technical groups to the mission of the enterprise.
- **Example 3: Assessing the impact of software process improvement**
  - Since ongoing implementation of SPI activities are in place, the schedule, cost and quality of future software projects are expected to be significantly better than previous efforts.

Reality Check

- Only <u>one out of five</u> SEM programs succeed [SEI-2001-TN-026]
- The reasons for failure are not technical but organizational:
  - No sustained management commitment and support
  - Expensive, cumbersome
  - Metrics data not tied to business goals
  - Metrics irrelevant, not understood, resisted, perceived to be unfair

Why should we care about SEM programs?

- Others do
  - It is considered one of the 16 Most Critical Software Practices^{TM} for performance-based project management.
  - project management network www.spmn.com
- The experts recommend it
  - It is part of SEI’s CMMI requirements.
- Become proactive rather than reactive
  - Create a measurement program to monitor issues and determine the likelihood of risks occurring.

What do successful SEM programs have in common?

- **Embrace measurements as part of the organization and assign organizational responsibility for**
  - Identification
  - Collection
  - Analysis
  - Reporting of metrics

- **Make measurements part of the business and include them in the**
  - Definition of process
  - Identification of risks or issues
  - Determination of project success factors
  - Decision making

- **Effectively align measurements and business goals**
- **Practice measurements on a continuous basis**
- **Make the goal to be “improvement” rather than “measurement”**

Key questions to ask before initiating a SEM program

- **Can you find the right people?**
- **Can you select the right measurements and collect data to provide insight into the following areas?**
  - The quality of your products
  - The effectiveness of your processes
  - The conformance to your process
  - Early indications of problems
  - The provision of a basis for future estimation of cost, quality, and schedule (e.g. benchmarks, thresholds, etc.)
- **Can you establish benchmarks for the above measurements by initially using suggested industry norms?**
- **Can your own benchmarks and thresholds evolve over time, based upon experience?**
- **Can you make all metrics data available to all project personnel along with the latest revision of project plans (e.g. measures of progress on schedule, risks, effort expenditures)?**

How do I jump-start a SEM program?

- **Explore various existing measurement programs**
  - Contact other people who have experience with SEM programs
  - External consultation
  - Literature review
  - Guide books and corporate reports

- **Understand and customize a *Goal-Question-Metric (GQM)* approach for establishing a measurement initiative to include:**
  - A list of improvement goals
  - A list of quantifiable questions
  - A minimal set of (core) measurement areas
  - A list of measurement application levels
  - A list of metrics
  - A measurement and validation process

- **Create and propose a go-forward plan to**
  - Implement a first-cut measurement framework
  - A transition to a long-term SEM program

Typical starting state

- Ad hoc measurement initiatives already in place
- Islands of measurement data
- Lack of awareness of measurement activities
- Lack of understanding, knowledge and training surrounding a measurement program
- Apparent need for an integrated framework to identify, capture, analyze and report measurement data

The Goal-Question-Metric (GQM) approach

- A goal-driven method for developing and maintaining a meaningful SEM program that ensures:
  - Measurement alignment with organization business and technical goals
  - Software process improvement
  - Risk management
  - Product quality improvement

Benefits of GQM

- Achievement of improvement goals
- Increased quality awareness and quality assurance involvement
- Increased capability to conduct improvement initiatives
- Improved group synergy
- Financial gains

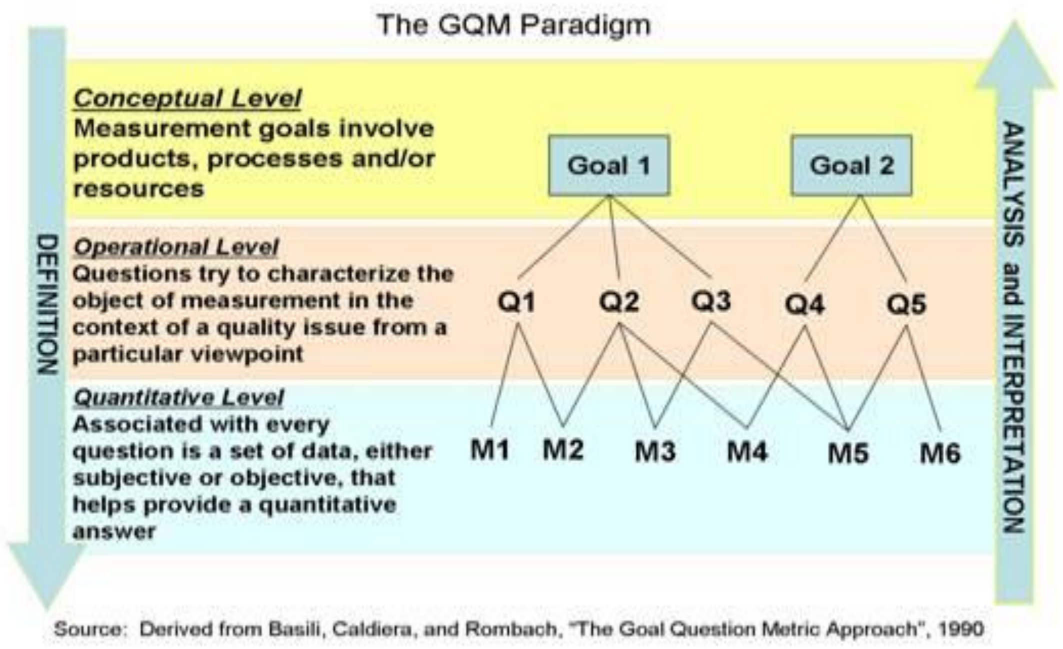
Why GQM?

- It is simple, easy to understand, learn and apply
- It leads to the creation of a "goal-oriented" measurement program rather than a "metrics-based" program
- It allows for tailoring to the organization's own business needs and objectives
- It is the common underlining motif among most of the successful measurement programs
- It is recommended by the SEI's CMMI for transition to level 2 maturity
- It results in meaningful measurements for
  - tracking projects
  - collecting metrics to support decision making
  - managing how to meet improvement goals

The GQM 6-step process

1. Develop a set of corporate, division and project business goals and associated measurement goals for productivity and quality.
2. Generate questions that define those goals as completely as possible in a quantifiable way.
3. Specify the measures needed to be collected to answer those questions and track process and product conformance to those goals.
4. Develop mechanisms for data collection.
5. Collect, validate and analyze the data in real time to provide feedback to projects for corrective action.
6. Analyze the data in a postmortem fashion to assess conformance to the goals and to make recommendations for future improvement.

Levels of GQM


The GQM Paradigm
DEFINITION
Conceptual Level
Measurement goals involve products, processes and/or resources
Operational Level
Questions try to characterize the object of measurement in the context of a quality issue from a particular viewpoint
Quantitative Level
Associated with every question is a set of data, either subjective or objective, that helps provide a quantitative answer
Goal 1
Goal 2
Q1
Q2
Q3
Q4
Q5
M1
M2
M3
M4
M5
M6
ANALYSIS and INTERPRETATION
Source: Derived from Basili, Caldiera, and Rombach, "The Goal Question Metric Approach", 1990

CIO, CTO, VPs, Stakeholders

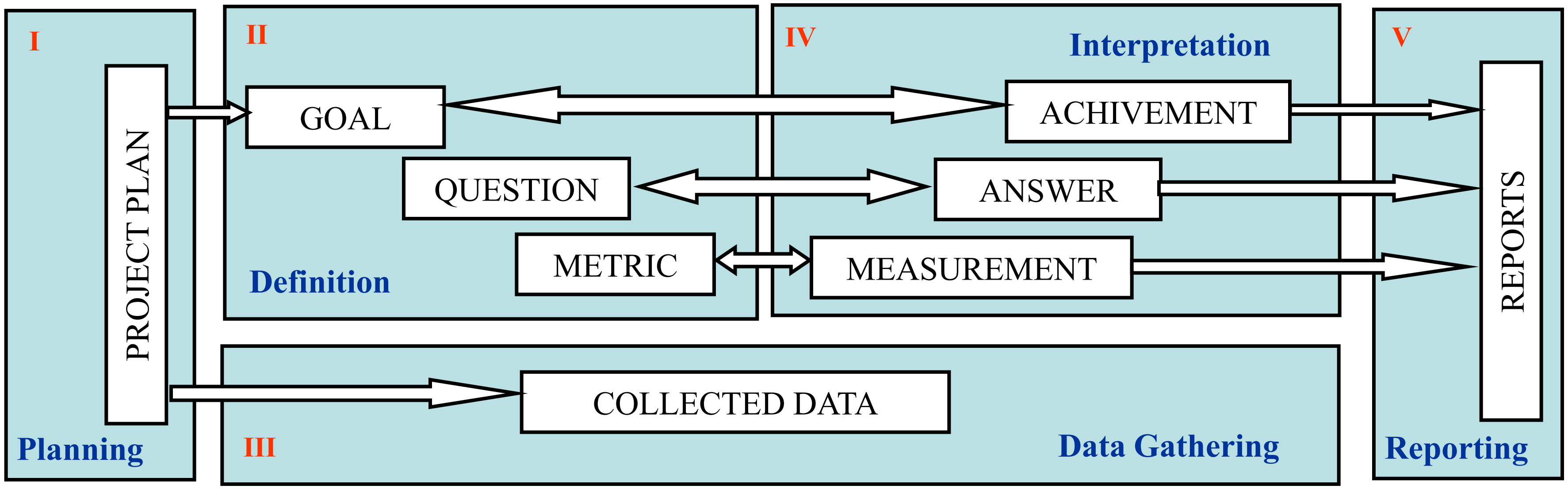
Project managers, developers, customers, other stakeholders

Development team: practitioners, project managers, developers

Implementing GQM

- Select the right people (at all levels)
- Secure management commitment to support measurement results
- Set and state explicit measurement goals
- Thoroughly plan the measurement program and document it (explicit and operational definitions)
- Don’t create false measurement goals
- Acquire implicit quality models from the team
- Consider context
- Derive appropriate metrics
- Stay focused on goals when analyzing data
- Let the data interpreted by the people involved
- Integrate the measurement activities with regular project activities
- Don’t use measurements for other purposes
- Establish an infrastructure to support the measurement program
- Ensure that measurement is viewed as a vehicle not the end goal
- Get training in GQM before moving forward

The GQM Template of Activities



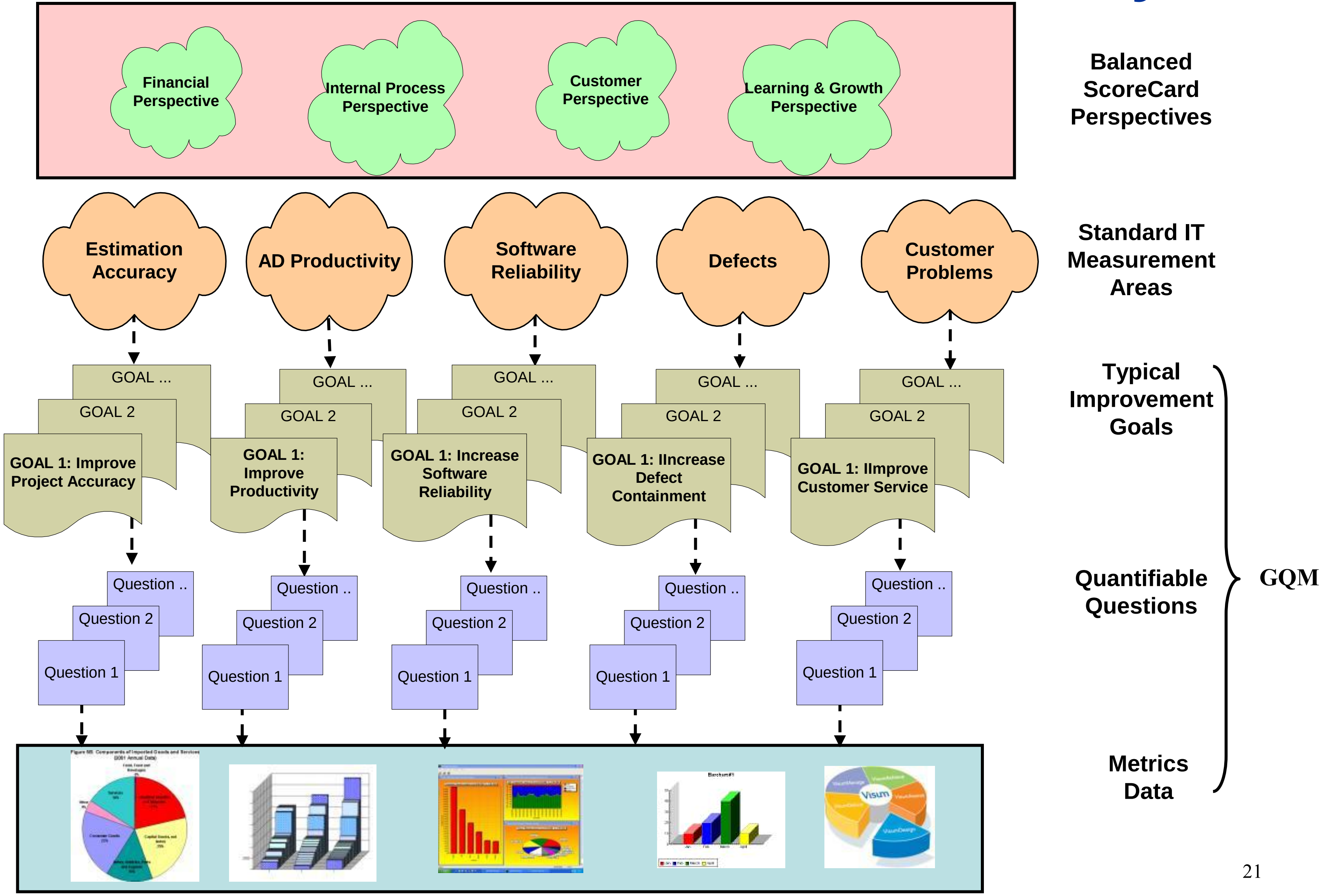
STAGE	ACTIVITIES
I. Planning	Preparation
II. Definition	Specification of GQM
III. Data Gathering	Measurement & Validation
IV. Interpretation	Analysis and feedback
V. Reporting	Present results

Does GQM align with industrial needs?

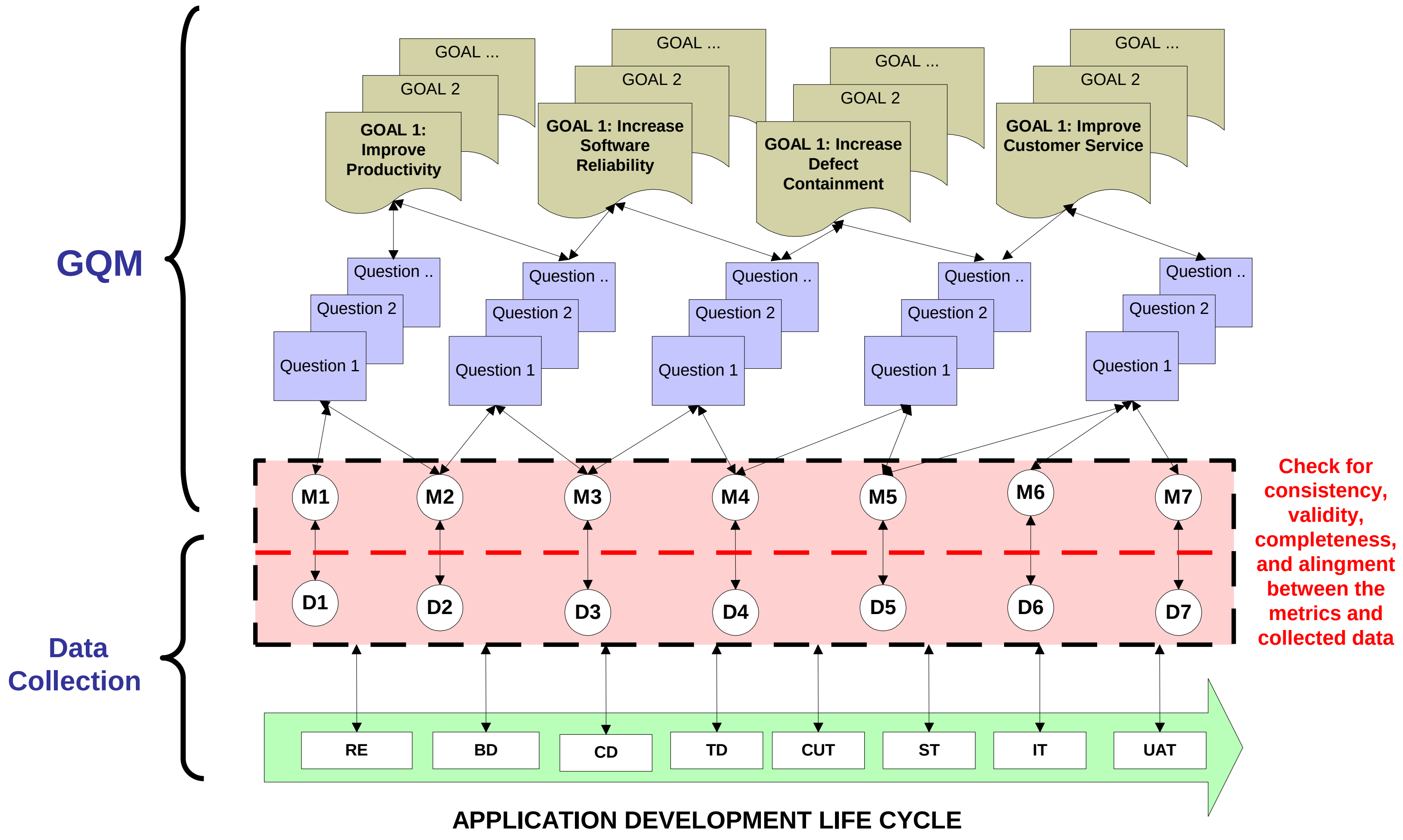
YES, because

- **Industry is goal oriented, so is GQM**
- **GQM supports multiple goals**
- **It solves relevant and practical problems**
- **It is both process and product-oriented**
- **Focus on team and project support**
- **It ensures clear understanding and use of collected data**
- **May be integrated with QA and testing efforts**

Measurement Areas Hierarchy


Financial Perspective
Internal Process Perspective
Customer Perspective
Learning & Growth Perspective
Balanced ScoreCard Perspectives
Estimation Accuracy
AD Productivity
Software Reliability
Defects
Customer Problems
Standard IT Measurement Areas
GOAL ...
GOAL 2
GOAL 1: Improve Project Accuracy
GOAL 1: Improve Productivity
GOAL 1: Increase Software Reliability
GOAL 1: IIncrease Defect Containment
GOAL 1: IImprove Customer Service
Typical Improvement Goals
Question ..
Question 2
Question 1
Quantifiable Questions
GQM
Metrics Data

Aligning GQM with the software development life cycle

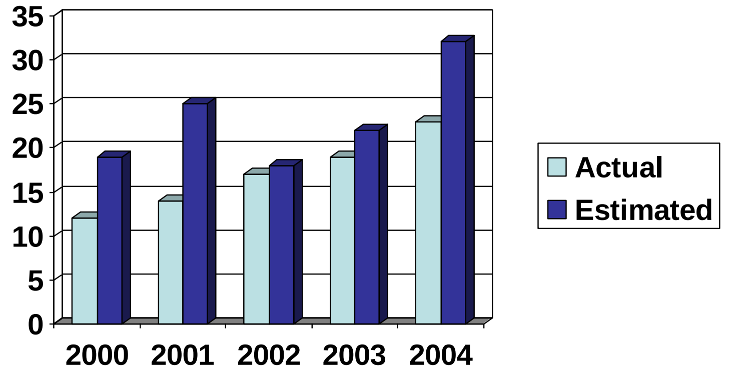

GQM
Data Collection
GOAL ...
GOAL 2
GOAL 1: Improve Productivity
GOAL ...
GOAL 2
GOAL 1: Increase Software Reliability
GOAL ...
GOAL 2
GOAL 1: Increase Defect Containment
GOAL ...
GOAL 2
GOAL 1: Improve Customer Service
Question ..
Question 2
Question 1
M1
M2
M3
M4
M5
M6
M7
D1
D2
D3
D4
D5
D6
D7
Check for consistency, validity, completeness, and alingment between the metrics and collected data
RE
BD
CD
TD
CUT
ST
IT
UAT
APPLICATION DEVELOPMENT LIFE CYCLE

LEGEND
M = Metrics
D = Data

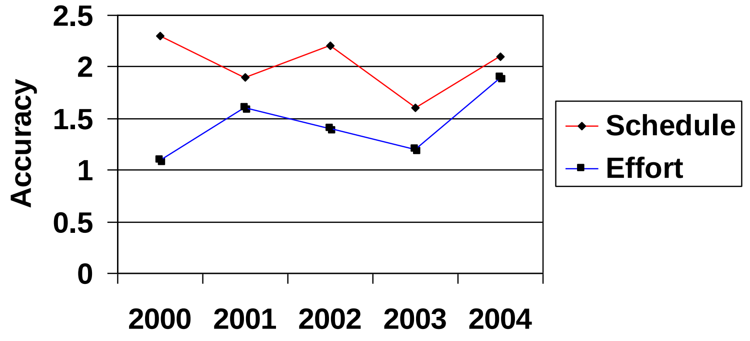
GOAL 1: Improve Project Accuracy

GOAL	QUESTIONS	METRICS
Improve Project Accuracy	**Q1:** What is the accuracy of estimating the actual value of project schedule?	**M1: Schedule Estimation Accuracy** = actual project duration / estimated project duration
	Q2: What is the accuracy of estimating the actual value of project effort?	**M2: Effort Estimation Accuracy** = actual project effort / estimated project effort

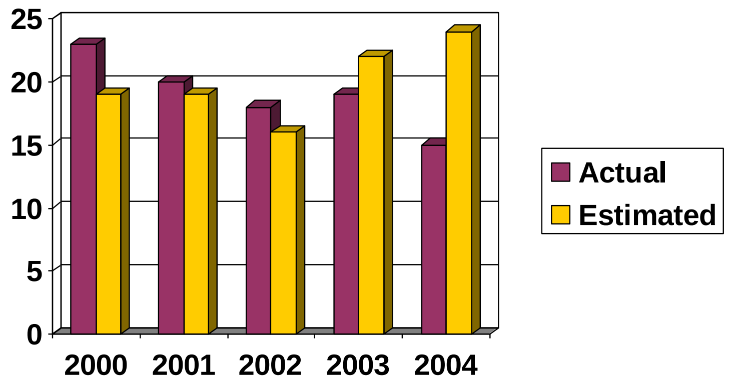
Schedule Estimation Accuracy



Estimation Accuracy (Goal = 1)



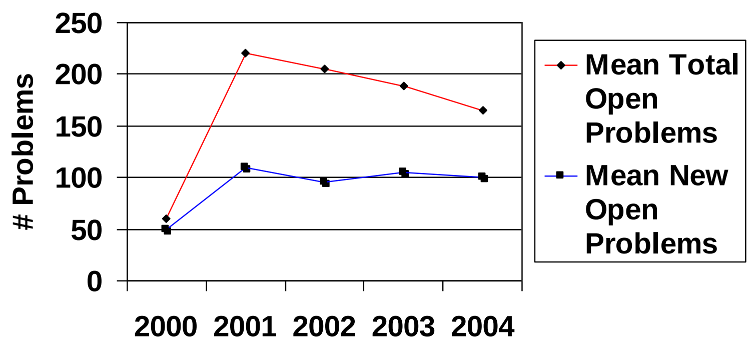
Effort Estimation Accuracy



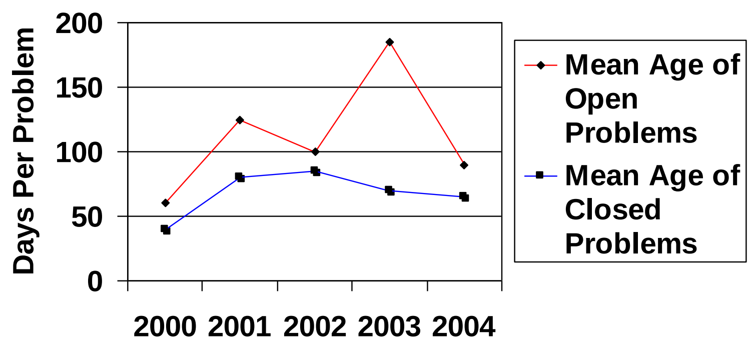
GOAL 2: Improve Customer Service

GOAL	QUESTIONS	METRICS
Improve Customer Service	**Q1:** What is the number of new problems that were opened during a time unit?	**M1: New Open Problems** = total new post-release problems opened during the time unit
	Q2: What is the total number of open problems at the end of a time unit?	**M2: Total Open Problems** = total number of post-release problems that remain open during the time unit
	Q3: What is the mean age of open problems at the end of a time unit?	**M3: Age of Open Problems** = total time post-release problems remaining open at the end of the time unit have been open / number of open post-release problems remaining open by the end of the time unit
	Q4: What is the mean age of the problems that were closed during a time unit?	**M4: Age of Closed Problems** = total time post-release problem closed within the time unit were open / number of post-release problems closed within the time unit

Post-release Problem Report Activity

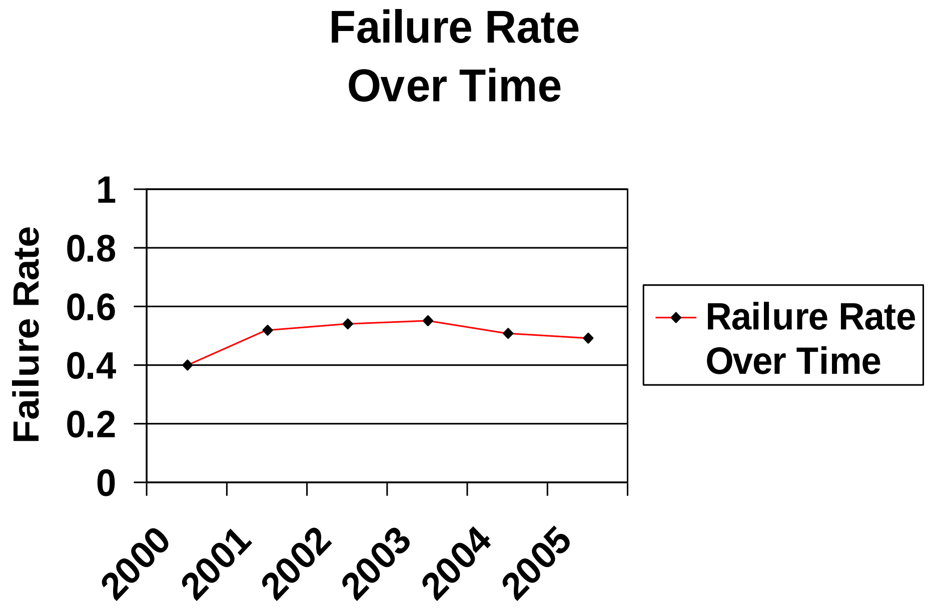


Post-release Problem Report Aging



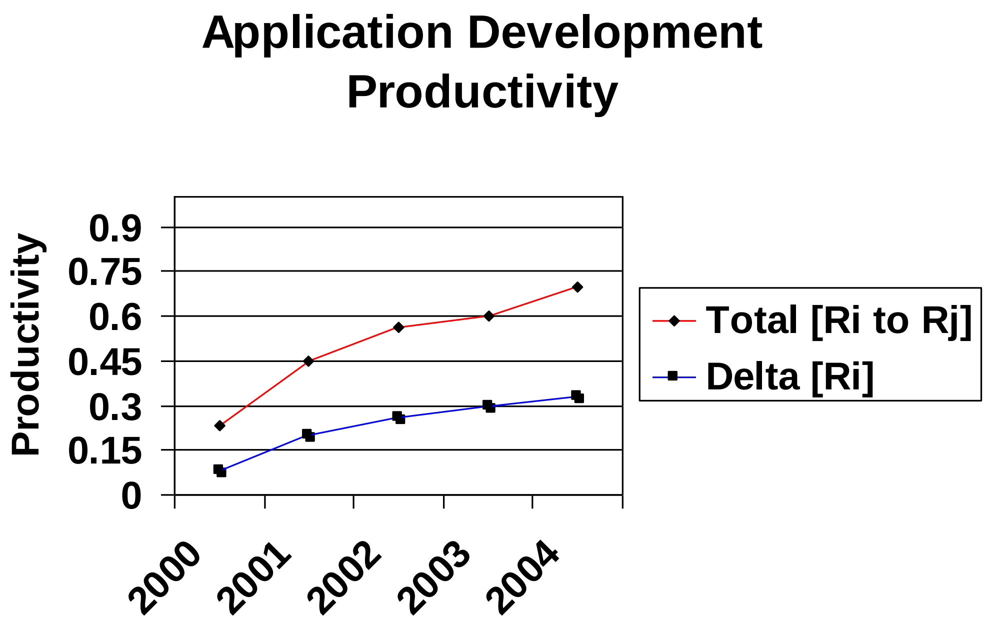
GOAL 3: Increase Software Reliability

GOAL	QUESTION	METRIC
Increase Software Reliability	**Q:** What is the rate of software failures, and how does it change over time?	**M: Failure Rate** = number of failures / execution time



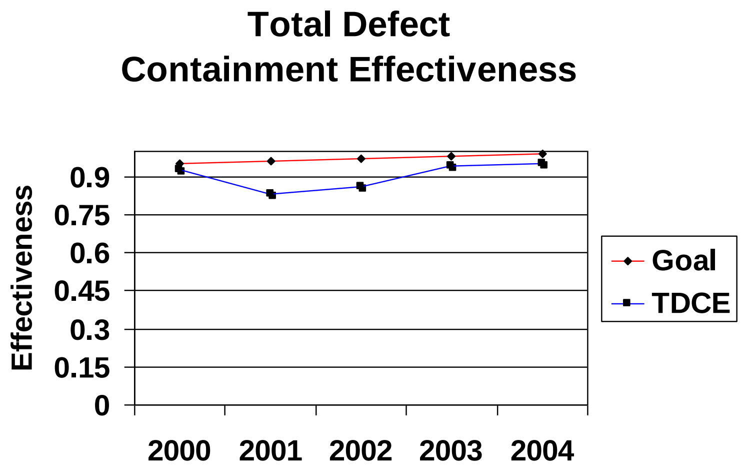
GOAL 4: Increase Productivity

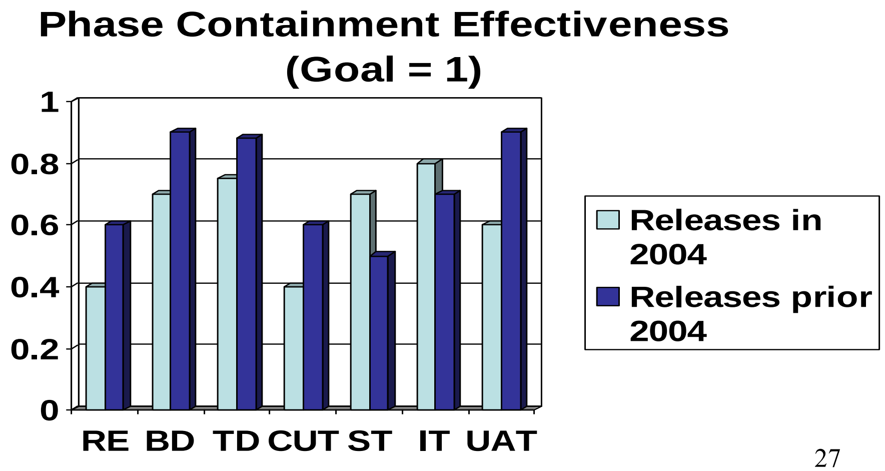
GOAL	QUESTION	METRICS
Increase Application Development Productivity	**Q:** What was the productivity of AD projects (based on source size)?	**M1: Total AD Productivity** = total source size / incremental AD effort [from Release i to Release j]
		M2: AD Productivity-delta = delta source size / AD effort [for Release i]



GOAL 5: Increase Defect Containment

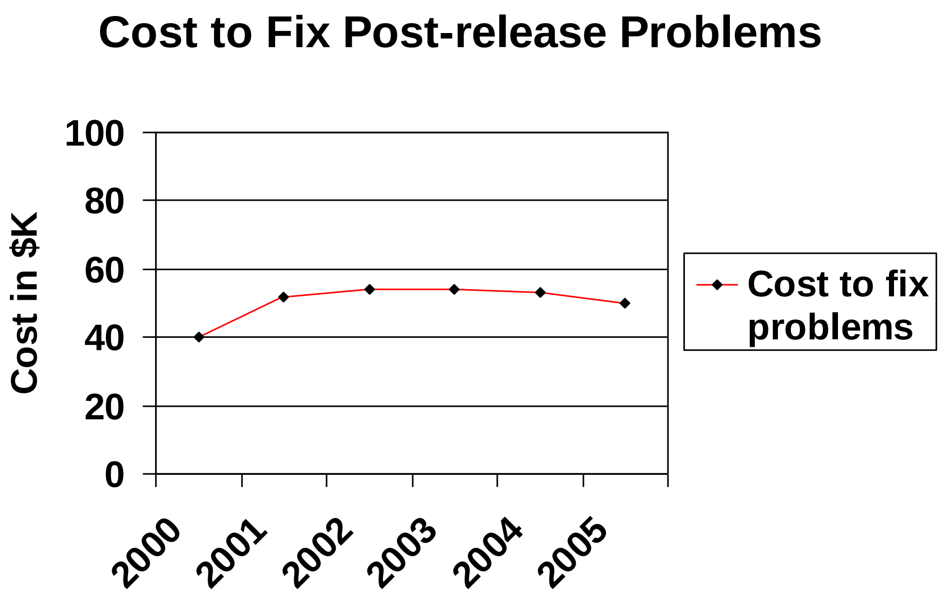
GOAL	QUESTIONS	METRICS
Increase Defect Containment	**Q1:** What is the currently known effectiveness of the defect detection process prior to release?	**M1: Total Defect Containment Effectiveness** = number of pre-release defects / number of pre-release defects + number of post-release defects
	Q2: What is the currently known containment effectiveness of faults introduced during each constructive phase of AD for a particular application?	**M2: Phase Containment Effectiveness** = number of phase i errors / number of phase i errors + number of phase i defects
	Q3: What is the relative time spent dealing with defects with respect to the overall AD time?	**M3: Relative Defect Time** = time spent dealing with defects / time spent for AD



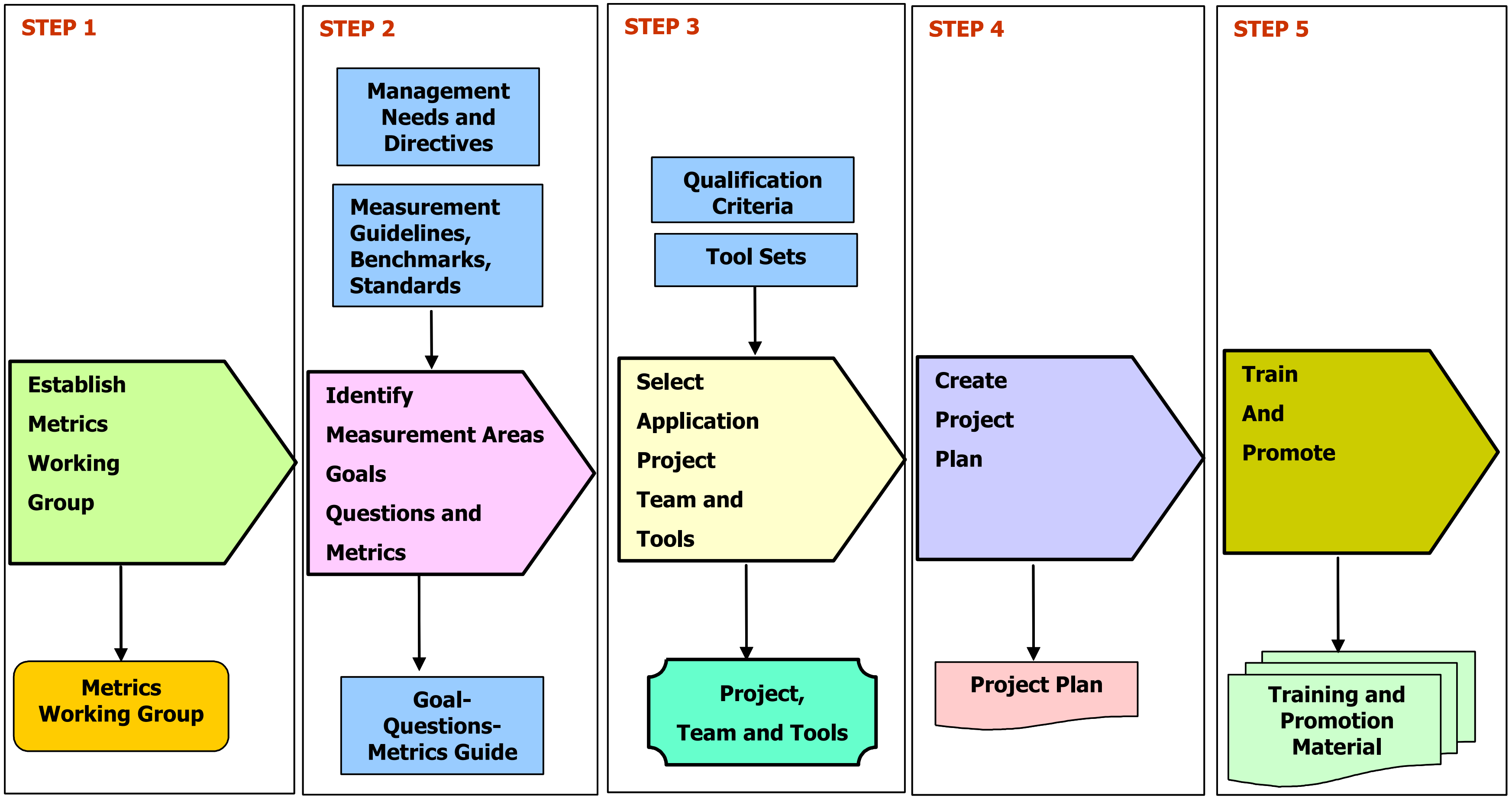


GOAL 6: Reduce Cost of Nonconformance

GOAL	QUESTION	METRIC
Reduce the Cost of Nonconformance	**Q:** What was the cost to fix post-release problems during a time unit?	**M: Cost of Fixing Problems** = dollar cost associated with fixing post-release problems within a time unit



Plan of Action


STEP 1
Establish Metrics Working Group
Metrics Working Group
STEP 2
Management Needs and Directives
Measurement Guidelines, Benchmarks, Standards
Identify Measurement Areas Goals Questions and Metrics
Goal-Questions-Metrics Guide
STEP 3
Qualification Criteria
Tool Sets
Select Application Project Team and Tools
Project, Team and Tools
STEP 4
Create Project Plan
Project Plan
STEP 5
Train And Promote
Training and Promotion Material

Metrics Working Group: Deliverables

- **A list of measurement areas tailored to your organization**
- **A list of derived improvement goals for**
  - Quantitative tracking
  - Management of software projects
  - Quality assessment
- **A list of quantifiable questions**
- **A minimal set (core) of metrics to measure achievement of the improvement goals**
- **A first-cut metrics implementation guide**
- **A pool of candidate projects**
- **A list of recommended metrics tools**

Tools evaluation criteria

Tool User Perspective	Manager/Buyer Perspective	Tool Builder Perspective
Training	Cost	Scalability
Intuitiveness	Increase productivity	Performance
Ease-to-use	System performance	Parsing
Ease-to-learn	Product maturity	Browsing, navigation, edit
Functionality	Standard compliance	View transformation
Response time	Vendor's viability	Filtering, screening
Customer support	Platform/architecture	Architecture
Documentation	Impact on existing business process	Visualization/presentation
	Risk	Storage/retrieval
		Layout
		Web features
		Architecture
		Graphical abstractions
		Innovation

Summary of Suggestions

- Address the *culture* issue first. Respect people's needs.
- Invite the *right people* to join you in this effort.
- Think *global*. Talk to other people. Read, read, read…
- Study, *understand* and then customize the GQM approach to your needs.
- Convince yourself and others that the goal is "improvement" not 'measurement"
- Ensure *alignment* between business needs, improvement goals and metrics data.
- Take *baby steps*. Start small and build on success. Be ready to show some short-term success to management.
- Use *pilot projects* to verify feasibility and to test definitions, checklists and templates.
- Select the *right tools* that fit your process.

Sample Goals, Questions and Metrics

IMPROVEMENT GOALS	QUESTIONS	METRICS
1. Increase AD Productivity	1.1: What was the productivity of AD projects (based on source size)?	1.1a: ADP = total source size / incremental AD effort [Ri to Rj] 1.1b: ADP-delta = delta source size / AD effort [for Ri]
2. Improve Project Predictability	2.1: What was the accuracy of estimating the actual value of project schedule? 2.2: What was the accuracy of estimating the actual value of project effort?	2.1: SEA = actual project duration / estimated project duration 2.2: EEA = actual project effort / estimated project effort
3. Increase Software Reliability	3.1: What is the rate of software failures, and how does it change over time?	3.1: FR = number of failures / execution time
4. Increase Defect Containment	4.1: What is the currently known effectiveness of the defect detection process prior to release? 4.2: What is the currently known containment effectiveness of faults introduced during each constructive phase of AD for a particular application? 4.3: What is the relative time spent dealing with defects with respect to the overall AD time?	4.1: TDCE = number of pre-release defects / number of pre-release defects + number of post-release defects 4.2: PCEi = number of phase i errors / number of phase i errors + number of phase i defects 4.3: RDT = time spent dealing with defects / time spent for AD
5. Decrease Software Defect Density	5.1: What is the normalized number of in-process faults, and how does it compare with the number of in-process defects? 5.2: What is the currently known defect content of software delivered to customers, with respect to source size? 5.3: What is the currently known customer-found defect content of delivered software, with respect to source size?	5.1a: IPF = in-process faults caused by incremental AD / delta source size 5.1b: IPD = in-process defects caused by incremental AD / delta source size 5.1c: IPE = in-process errors caused by incremental AD / delta source size 5.2a: TRD-total = number of released defects / total source size 5.2b: TRD-range = number of released defects caused by incremental AD / total source size [from Ri to Rj] 5.2c: TRD-delta = number of post released defects / delta source size 5.3a: CFD-total = number of customer found defects / total source size 5.3b: CFD-range = number of customer found defects caused by incremental AD / total source size [from Ri to Rj] 5.3c: CFD-delta = number of customer found post-release defects / delta source size
6. Improve Customer Service	6.1: What is the number of new problems that were opened during a time unit? 6.2: What is the total number of open problems at the end of a time unit? 6.3: What is the mean age of open problems at the end of a time unit? 6.4: What is the mean age of the problems that were closed during a time unit?	6.1: NOP = total new post-release problems opened during the time unit 6.2: TOP = total number of post-release problems that remain open during the time unit 6.3: AOP = (total time post-release problems remaining open at the end of the time unit have been open) / (number of open post-release problems remaining open by the end of the time unit) 6.4: ACP = total time post-release problem closed within the time unit were open / number of post-release problems closed within the time unit
7. Reduce the Cost of Nonconformance	7.1 What was the cost to fix post-release problems during a time unit?	7.1: CFP = dollar cost associated with fixing post-release problems within a time unit

Note: See next slide for Acronyms & Terminology

Terminology and Acronyms

LEGEND

- **AD** = Application Development
- **ACP** = Age of Closed Problems
- **AOP** = Age of Open Problems
- **CFD** = Customer-found defects
- **CFP** = Cost of Fixing Problems
- **EEA** = Effort Estimation Accuracy
- **FR** = Failure Rate
- **IPD** = In-process Defects
- **IPE** = In-process Errors
- **IPF** = In-process Faults
- **LOC** = Line of Code (excluding lines that contain only comments or blanks)
- **NOP** = New Open Problems
- **PCE** = Phase Containment Effectiveness
- **RDT** = Relative Defect Time
- **SEA** = Schedule Estimation Accuracy
- **SP** = Software Productivity
- **TDCE** = Total Defect Containment Effectiveness
- **TOP** = Total Open Problems
- **TRD** = Total Release Defects
- **TSS** = Total Source Size of the release
- **TSS-delta** = The size of the source code added, deleted and modified from the previous release

TERMINOLOGY

- ***Software Problem*** = a discrepancy between a delivered artifact of an AD phase and its :
  - documentation
  - the product of an earlier phase
  - user requirements
- ***Problem Status*** = a problem can be
  - *open* (i.e. the problem has been reported)
  - *closed-available* (i.e. a tested fix is available)
  - *closed* (i.e. a tested fix has been installed)
- ***Error*** = A problem found during the review of the phase where it was introduced.
- ***Defect*** = A problem found later than the review of the phase where it was introduced.
- ***Fault*** = Errors and defects are considered faults.
- ***Failure*** = Inability of software to perform its required function.
  - It can be caused by a defect encountered during software execution (i.e. testing and operation).
  - When a failure is observed, problem reports are created and analyzed in order to identify the defects that are causing the failure.
- ***Software Release*** = It has entered the phase of beta test and operation.