AUTOMATED SOFTWARE MEASUREMENT

Richard W. Selby
University of California
Dept. of Information & Computer Science
Irvine, CA 92717

714-856-6326
selby@ics.uci.edu

Selby/Oct-94
OVERVIEW

- Introduction
- Measurement usage scenarios
- Key concepts and terminology
- Metric data categories
- Measurement system design principles
- Example measurement system: Amadeus
- Measurement transition and adoption
- Toward a national metric database (NSDIR)
- Summary
AUTOMATED SOFTWARE MEASUREMENT

MEASUREMENT FEATURES AND BENEFITS

- Better visibility into software engineering processes
- Increased confidence in management and technical decisions
- Improved cost and schedule performance measurement of software engineering organizations
- Enhanced quality measurement of software products and processes
- Measurement of heterogeneous product and process representations
- Framework that integrates metric data from processes, products, projects, teams, and organizations
- A flexible set of metrics --- users can extend the baseline set by defining, customizing, and adding their own metrics
- An open architecture for extensibility and incorporation of foreign tools

Selby/Oct-94
USE AUTOMATED SOFTWARE MEASUREMENT TO:

Solve the Product and process analysis and evaluation Problem . . .
Use measurement to evaluate or compare software design and code

Solve the Improvement Problem . . .
Use measurement to enable process and product improvement

Solve the Status and progress report generation Problem . . .
Use measurement to summarize weekly, monthly, or interactive project status and progress

Solve the Metric integration Problem . . .
Use measurement-based analysis to integrate product, process, project, team, and organization metrics

Solve the Resource allocation Problem . . .
Use measurement to focus development and testing effort, reviews, and specialized analyses
OVERVIEW

- Introduction
- Measurement usage scenarios
- Key concepts and terminology
- Metric data categories
- Measurement system design principles
- Example measurement system: Amadeus
- Measurement transition and adoption
- Toward a national metric database (NSDIR)
- Summary
EXAMPLE MEASUREMENT SCENARIOS ----

PEOPLE WHO BENEFIT FROM MEASUREMENT AND WHY

- Developers benefit from measurement tools that analyze and evaluate designs, code, etc.

- Testers and developers benefit from measurement analyses and predictions of high risk (e.g., high error, high effort) components that focus resources

- Managers and developers benefit from measurement tools that investigate project status interactively
EXAMPLE MEASUREMENT USAGE SCENARIOS ----

PEOPLE WHO BENEFIT FROM MEASUREMENT AND WHY

- SEPG personnel benefit from measurement tools that collect metrics for baselines and monitor progress for improvement.

- Process engineers benefit from measurement tools that instrument projects for feedback and improvement.

- QA personnel benefit from measurement tools that generate weekly, monthly, etc. status reports for internal or customer use.
EXAMPLE MEASUREMENT USAGE SCENARIOS ----

PEOPLE WHO BENEFIT FROM MEASUREMENT AND WHY

• Managers benefit from measurement reports of predicted vs. actual project schedules and costs that evaluate project status

• Executives benefit from measurement analyses and visualizations that summarize and compare projects

• Proposal writers benefit from metric data repositories for estimation of costs, schedules, etc.

• Software council members benefit from measurement tools that monitor “program health” metrics
AUTOMATED SOFTWARE MEASUREMENT

METRIC SETS SHOULD BE FLEXIBLE

The metrics are as simple or complex as you wish
• Reuse or tailor existing metrics or define your own metrics

Incremental gain for incremental investment
• Initially "turn on" just a few metrics, such as size or changes

Adopt company-wide or industry-wide metric set
• Example metric sets: SEI core metrics, STEP metrics, Air Force metrics

Support measurement at all levels of Capability Maturity Model (CMM)
OVERVIEW

- Introduction
- Measurement usage scenarios
- Key concepts and terminology
- Metric data categories
- Measurement system design principles
- Example measurement system: Amadeus
- Measurement transition and adoption
- Toward a national metric database (NSDIR)
- Summary
KEY MEASUREMENT CONCEPTS AND TERMS - (1 of 2)

• Metric -- A metric is a numeric, ordinal, interval, or symbolic abstraction of a software product, process, project, team, or organization.

• Event -- The term "event" is used synonymously with the term "metric." E.g., an event occurrence in a process is a metric on that process; the metric value is the timestamp of the event occurrence.

• Monitor -- A process that automatically generates events corresponding to, e.g., accesses and modifications to files or calendar time abstractions such as each hour, day or week.

• Statistic -- A single metric derived from a set of metrics, such as average, current_value, or total.

• Agent -- A metric collection, analysis, reporting, graphing, or prediction tool (actually any tool or process that can be executed at the operating system command level can serve as an agent).
• Specification -- A condition-action pair that defines the empirical metric condition that should cause an action (i.e., agent) to be executed.

• Template -- A pre-defined description of an interactive graphical user interface panel for generating metric graphs and reports, exporting metric data, importing metric data, or entering metric data.

• Repository -- A database of metric data along with the templates, specifications, and agents that collect or analyze that data.

• Metric interpreter -- A server tool that manages access and use of a repository, including execution of its templates, specifications, and agents. There is a one-to-one relationship between repositories and metric interpreters.
OVERVIEW

• Introduction
• Measurement usage scenarios
• Key concepts and terminology
• **Metric data categories**
• Measurement system design principles
• Example measurement system: Amadeus
• Measurement transition and adoption
• Toward a national metric database (NSDIR)
• Summary
EXAMPLE METRIC CATEGORIES AND ANALYSES

• Example metric categories:

<table>
<thead>
<tr>
<th>Changes</th>
<th>Cycletime</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>Documentation</td>
<td>Schedule</td>
</tr>
<tr>
<td>Computer_resource</td>
<td>Effort</td>
<td>Size</td>
</tr>
<tr>
<td>Cost</td>
<td>Errors</td>
<td>Structure</td>
</tr>
</tbody>
</table>

• Example fields for a metric data record:

<table>
<thead>
<tr>
<th>Organization</th>
<th>Artifact</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Artifact_type</td>
<td>Month</td>
</tr>
<tr>
<td>Release</td>
<td>Host</td>
<td>Weekday</td>
</tr>
<tr>
<td>Subrelease</td>
<td>Context</td>
<td>Day</td>
</tr>
<tr>
<td>Phase</td>
<td>Process</td>
<td>Hour</td>
</tr>
<tr>
<td>Activity</td>
<td>Metric_name</td>
<td>Minute</td>
</tr>
<tr>
<td>Project_start_date</td>
<td>Metric_origin</td>
<td>Second</td>
</tr>
<tr>
<td>Date</td>
<td>Metric_status</td>
<td>Time</td>
</tr>
<tr>
<td>Person</td>
<td>Metric_value</td>
<td>Timestamp</td>
</tr>
<tr>
<td>Abstraction level</td>
<td>Comment</td>
<td>Millitimestamp</td>
</tr>
</tbody>
</table>

• Example graphs and reports:

- Errors by week, Errors by week and by severity, Predicted vs. actual effort by phase, Lines of code growth by month, Cyclomatic complexity by component, Project status report
METRIC CATEGORIES FOR AUTOMATION AND INTERACTIVE DATA ENTRY

• Example metric categories that can be automated (stand-alone):
  -- requires: File system monitor and agents

  | Changes   | Documentation | Structure |
  | Complexity| Size          |           |

• Example metric categories that can be automated (after instrumentation):
  -- requires: Instrumentation to track begin-end pairs

  | Computer_resource | Cycletime |

• Example metric categories that require interactive data entry:
  -- requires: Interactive data entry panels

  | Cost | Errors | Schedule |
  | Effort | Requirements | |
  |       |          |         |

Selby/Oct-94
MEASUREMENT ANALYSES AND VISUALIZATIONS

Analyses

• **Baseline analysis** --- Collect data to establish baseline for understanding and as foundation for improvement
• **Comparative and summary analysis** --- Summarize projects or compare them relative to baselines, past projects, or user-defined criteria
• **Predictive analysis** --- Identify components or process steps that are likely to have user-specifiable properties, e.g., error-prone or costly

Visualizations

• **Real-time displays** --- Automatically updated metric data displays
• **Graphical** --- Graphical metric data displays
• **Tabular** --- Tabular metric data displays
• **Sequential monitoring** --- Displays of traces of collected data

Frameworks that integrate process, product, project, team & organization metrics
OVERVIEW

• Introduction
• Measurement usage scenarios
• Key concepts and terminology
• Metric data categories
• Measurement system design principles
• Example measurement system: Amadeus
• Measurement transition and adoption
• Toward a national metric database (NSDIR)
• Summary
AUTOMATED SOFTWARE MEASUREMENT

AUTOMATED SOFTWARE MEASUREMENT --- WHAT IS IT?

• An automated system for software
  • metric collection,
  • metric analysis,
  • metric reporting,
  • metric graphing, and
  • metric prediction

• Enables software process and product analysis and improvement.

• Needs to be very flexible, have an open architecture, and have a low entry barrier to usage.

• Example metrics supported are:
  • Size
  • Structure
  • Changes
  • Errors
  • Effort
  • Cycle time
  • Cost
  • Schedule
Fundamental Principle: Make Measurement Active

- In the past, measurement has primarily been used in a passive manner
- Make measurement active, not passive
- Integrate measurement into processes to create empirically guided process programs

Examples:

- Guide development and evolution processes by
  --- identifying components likely to have integration faults
  --- suggesting components likely to be reusable
  --- identifying components likely to be delivered late
- Adapt and improve processes by
  --- Executing specialized tools and processes on high risk components
- Analyze processes by
  --- graphically tracking progress relative to baselines
  --- comparing types of faults detected across phases
Empirically Based Techniques

- Scalable to large projects
- Calibratable to new environments
- Measurements are integratable
- Leverage previous experience
- Process, product & team attributes

Focus on High-Payoff Areas: the 80:20 rule

Example:

```
Attribute-i
  /   \
/     \ \
0-3    4-5

Attribute-j
  /   \
/     \ \
0-12  >12

Attribute-k
  /   \
/     \ \
0-18  >18

Attribute-l
  /   \
/     \ \
Real-time  Non-real time
0-150 >150

Attribute-m
  /   \
/     \ \
0-3    4-5

"+" : Classified as likely to have property P (e.g., integration errors)

"-" : Classified as unlikely to have property P
```
The Problem:

- Numerous processes can not be prescribed completely in advance
- Processes need guidance and improvement during execution
- Significant aspects of guidance and improvement are inherently empirical (vs. analytical)
- Measurement capabilities can provide the guidance and improvement, but they are not yet integrated with processes
- And how do we make use of the vast spectrum of data available?

Focus on large-scale systems and projects (e.g., exploit the 80:20 rule)

Approach: Develop architectural principles and abstract interfaces for measurement-driven analysis and feedback systems that integrate measurement and process, and enable empirical process adaptation, guidance, improvement, and understanding

... iterative investigations, prototyping, and evaluations ...
Measurement System Design Principles

Overall:
• Integration of measurement, process, and environment

Measurement:
• Proactive measurement, analysis, and guidance
• User-specifiable goals, with some auto. instrumentation
• Extensible integration framework
• Support for all four measurement abstractions
• Scalable, calibratable, concurrent data collection & analysis

Tool Integration:
• Separation of event and agent specification
• Dynamic interpretation of process, object, and time events
• Synchronous or asynchronous agent execution
• Low entry barrier with incremental activation of scripts

Heterogeneity:
• Multiple languages for product and process representation
• Agent set is extensible and can include foreign tools
What Measurement Systems Do Not Do

Measurement systems should not dictate:
- a fixed set of metrics
- a fixed set of analysis techniques
- a fixed set of properties to analyze and predict
- a fixed set of analysis goals
- a fixed process language
- a fixed programming language
- a fixed collection of historical metric data

.... because each of these aspects are different in different environments.

The system is extensible in each of these aspects, but does provide a baseline set of capabilities.
Measurement Architecture Issues

- **Goal:** Separate **agents** (measurement collection, analysis, feedback and improvement tools) and **events** (process, object, time, compound)

- **Agents** embody tools or processes

- **Events** originate from:
  - --- proc call in process language (Appl/a, Anna, Ada, C, C-shell scripts, [interactive,] Amadeus scripts)
  - --- operation on relation
  - --- time (at command representation)

- **Scripts** specify mapping from event specifications to agents.

  script = (  
    • event specification = event_name.artifact_instance.artifact_kind  
    • guard (event filter) = binary function of environment state  
    • blocking factor = asynch or synch spawn of agent  
    • agent = arbitrary tool, process, or visualization
  )

  Dynamic, incremental activation of newly created, modified, or reused scripts registers them with a server.

- **Server** matches event occurrences with event specifications, evaluates guards, and spawns agents for active scripts.
Interconnectivity Analysis

Applications:
- Reverse engineering
- Software structure evaluation
- Multiple interconnection criteria
- Multiple visualizations of system structure

In one application, technique was successfully used to locate components that were six times more error-prone than other components.
Integration With Measurement Systems

- **Originator of events**
  - Requires a language binding for procedure-call interface
  - Issues: (1) coarse vs. fine granularity of process/tool events and object events (e.g., before and after operation invocations on objects)
  - (2) automation of event instrumentation
  - (3) synchronous vs. asynchronous event interpretation

- **Manipulator of scripts**
  - Requires a language binding for procedure-call interfaces

- **Agent within scripts**
  - Requires scripts that invoke particular agent, which is any process/tool that is executable at the Unix shell level.
  - Issue: (1) access and representation of agent output objects
AUTOMATED SOFTWARE MEASUREMENT

COMPARING MEASUREMENT SYSTEMS

- Powerful?
  - Does it provide capabilities for metric collection, analysis, reporting, graphing, and process guidance?

- Ease of use?
  - How easy is its graphical user interface (GUI) and interactive command invocation?

- Fit into your project or organization?
  - Does it capture a wide spectrum of user metric goals and is it user-extensible?

- Flexible?
  - Does it provide incremental, dynamic activation of metrics?

- Automated?
  - Does it provide unobtrusive project monitoring and metric tool execution?

- Open architecture?
  - Does it provide metric data import and export capabilities?
Related Work

Measurement:

- Basili's improvement paradigm and Tame system
- Torii's Ginger approach and system
- RADC's ASQS and QUES approach and systems
- NASA's SME approach and system
- Army's STEP approach and system
- Humphrey/SEI's process maturity levels 4 and 5
- Weiderman/SEI's environment evaluation methodology

- Metric analyses at many organizations --- e.g., NASA & CSC, TRW, Hughes, IBM FSD

Tool integration:

- Event-based approaches and systems --- e.g., Reiss (Field), HP (SoftBench), Garlan (Forest), Notkin, Balzer (AP5), Taylor (Chiron), Osterweil (Appl/a)
OVERVIEW

- Introduction
- Measurement usage scenarios
- Key concepts and terminology
- Metric data categories
- Measurement system design principles
- Example measurement system: Amadeus
- Measurement transition and adoption
- Toward a national metric database (NSDIR)
- Summary
AMADEUS AUTOMATES METRICS

- Amadeus is an automated metric collection, analysis, reporting, graphing, and prediction system
- Flexible, unobtrusive measurement system with low entry barrier
- Open architecture emphasizing user-extensibility and tailorability
- Enables process and product improvement

Selby/Oct-94
AMADEUS PROVIDES FEEDBACK AND ENABLES IMPROVEMENT

Amadeus Measurement System
- Process metrics
- Product metrics
- Project metrics
- Team metrics
- Organization metrics

Measurement templates
Measurement specifications and agents

Resource, schedules
Processes, tools, user sessions
Objects, files

Metric-driven process guidance
Metric graphs
Metric reports
Metric data
AUTOMATED SOFTWARE MEASUREMENT

AMADEUS OPERATION

[Diagram showing various components and flows such as Active Processes, Tools, & User Sessions, Activated Templates & Specs, Graphs & Reports, Developer/Analyst, Templates & Specifications, Repository, etc.]
Validation Studies

- Goal: Identify components within two target classes ---
  top 25% of faults and top 25% of effort

- 16 NASA systems

- Correctness: 89.6%  
  \[ \text{Correctness} = \frac{a+d}{a+b+c+d} \times 100 \]

- Consistency: 79.5%  
  \[ \text{Consistency} = \frac{a}{a+b} \times 100 \]

- Completeness: 69.1%  
  \[ \text{Completeness} = \frac{a}{a+c} \times 100 \]

<table>
<thead>
<tr>
<th>actual</th>
<th>predicted</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>a+b</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>a+b+c</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>a+b+c+d</td>
</tr>
</tbody>
</table>
OVERVIEW

• Introduction
• Measurement usage scenarios
• Key concepts and terminology
• Metric data categories
• Measurement system design principles
• Example measurement system: Amadeus
• Measurement transition and adoption
• Toward a national metric database (NSDIR)
• Summary
GETTING STARTED WITH MEASUREMENT (1 of 2)

First day:
- Install measurement system
- Start to collect metric data to establish baseline
- Start to analyze or evaluate code, designs, etc.

After one week (and continuing weekly):
- Automatically generate weekly status reports and graphs for feedback and improvement

After two weeks (and continuing weekly):
- Automatically generate comparative metric reports and graphs for feedback and improvement
- Monitor progress relative to baselines
- Track predicted vs. actual schedules, costs, etc.

At end of month (and continuing monthly):
- Automatically generate metric reports and graphs for customer reporting requirements
GETTING STARTED WITH MEASUREMENT (2 of 2)

At end of three months (and continuing monthly until month 6):
  • Start automatically generating preliminary predictive analyses of high-risk components

At end of six to twelve months (and continuing weekly or monthly):
  • Automatically generating predictive analyses of high-risk components
  • Automatically analyze and summarize metric data for proposal cost, schedule, etc. estimation

Ongoing:
  • Interactively check project status

Periodically:
  • Automatically analyze metric data to update baselines

At major project milestones:
  • Automatically summarize and compare projects
METHODOLOGY FOR USING MEASUREMENT (1 of 2)

... a cyclic, incremental methodology ...

... many iterative cycles are generally preferable to fewer heavyweight cycles ...

(1) In the context of project or organization goals and constraints, such as improvement, characterization, or understanding, determine your measurement goals. Example measurement goals are baseline, summary, or predictive analysis of changes, schedule, cycletime, or product size growth. The measurement goals may include definition of custom metrics or adoption of a particular metric set, such as a company- or industry-wide metric set.

(2) Start the measurement system.

(3) Define, tailor, or reuse metric templates and specifications that collect, analyze, report, graph, and predict the metric data supporting the measurement goals, and activate the templates and specifications.
AUTOMATED SOFTWARE MEASUREMENT

METHODOLOGY FOR USING MEASUREMENT (2 of 2)

(4) If necessary, insert probes to detect process or object events that support the templates and specifications activated.

(5) Use measurement results from the measurement system as basis for feedback to both project or organization goals and constraints and measurement goals.

(6) Continue cycle at step (1).
# Measurement System Usage Migration Path

<table>
<thead>
<tr>
<th>Event type</th>
<th>Requirement</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st) time events</td>
<td>---</td>
<td>periodic analysis, guidance</td>
</tr>
<tr>
<td>2nd) object events</td>
<td>object monitoring</td>
<td>object analysis, guidance</td>
</tr>
<tr>
<td>3rd) process events</td>
<td>process/tool instrumentation</td>
<td>process analysis, guidance</td>
</tr>
</tbody>
</table>
AUTOMATED SOFTWARE MEASUREMENT

ADOPTION OF MEASUREMENT SYSTEM:
WHO DOES WHAT? --- USER and SYSTEM

USER:
- "Turns on", "turns off", tailors, reuses, etc. metric templates and specifications
- Instruments process events
- Provides data from off-line events or off-line data sources
- Provides data for subjective metric values

MEASUREMENT SYSTEM PROVIDES:
- Example and reusable metric templates and specifications
- Process event monitoring
- Object events and event monitoring
- Time events and event monitoring
- Metric infrastructure tools for automation, data storage, etc.
- Metric collection tools (called "agents") and capabilities
- Metric analysis tools
- Metric reporting tools
- Metric graphing tools
- Metric prediction tools
- Graphical user interface
AUTOMATED SOFTWARE MEASUREMENT

CRITERIA FOR HIGH ADOPTION RATE FOR INNOVATIONS

Relative advantage --- The innovation is technically superior (in terms of cost, functionality, "image", etc.) than the technology it supersedes.

Compatibility --- The innovation is compatible with existing values, skills, and work practices of potential adopters.

Lack of complexity --- The innovation is not relatively difficult to understand and use.

Trialability --- The innovation can be experimented with on a trial basis without undue effort and expense; it can be implemented incrementally and still provide a net positive benefit.

Observability --- The results and benefits of the innovation's use can be easily observed and communicated to others.

OVERVIEW

• Introduction
• Measurement usage scenarios
• Key concepts and terminology
• Metric data categories
• Measurement system design principles
• Example measurement system: Amadeus
• Measurement transition and adoption
• Toward a national metric database (NSDIR)
• Summary
NSDIR Background

- NSDIR Concept of Operation defines short-term operational approach for the Order-I Prototype
  - working on the document.
- NSDIR Strategic Plan defines long-term vision and direction.
- Status of NSDIR Strategic Plan: DRAFT.
- Definition of the NSDIR Strategic Plan is a community-driven activity, including government, industry, academia, and research labs.
- Cooperstown II Working groups will address further definition, modification, and evolution of a NSDIR Strategic Plan.
NSDIR Goals

National Goals

- Establish a national-level resource that supports US decision makers, practitioners, and researchers
  - calculate baselines, characterize problem domains, profile projects, track trends, evaluate techniques, calibrate predictions

- Provide services and products to support the long-term collection, analysis and management of software measurement data
  - facilitate goal-driven measurement, consistent definitions, data validation, meaningful interpretation

Adapted from Chruscicki, Andrew, "Creating a National Vision and Force in Software Through Software Measurement", Rome Labs, AUG 93.
Repository Perspectives

National Metrics Needs

- Socio-Economic perspective focuses on census-like, social and economic data.
- Macro-Level perspective focuses on US software industry as a whole.
- Micro-Level perspective focuses on US software domains and projects.
- Technology/Methodology perspective allows comparisons between state-of-the-art and state-of-the-practice.
- Research-Level perspective focuses on detailed measures about software engineering practices.

Together, these perspectives address much of the needs of the NSDIR and its users.
Repository Growth Path

- **Industrialization**
  - Innovative uses of information products
  - Quantitative comparison to other global forces
  - Measure output of US software machine

- **Realization**
  - Industry-level focus
  - Specialization & differentiation of services
  - Metrics & analysis marketplace
  - Fee-for-service

- **Stratification**
  - Statistical analysis of significant strata of data
  - Modeling methods and techniques
  - Significant expansion of services

- **Collaboration**
  - Facilitation of industry standards and policies
  - External business & government alliances
  - Normalization within a narrow domain
  - Identification of meaningful strata
  - Identification of barriers and issues to address

- **Foundation**
  - Identification of target suppliers & consumers
  - Clarification of information needs
  - Identification of roles and responsibilities
  - Initiate outreach to involve community of interest

- Industrialization spawns competition and force for services—quantitative measurement of US industrial S/W output.
- Realization senses the emerging marketplace viability of services and products begin to charge for services.
- Stratification leverages insights into appropriate strata of metrics data to support reliable approaches to modeling and prediction.
- Collaboration relies on broad community involvement to begin to address standardization and comparison issues.
- Foundation looks inward, focusing on structure, content, requirements to build some level of critical mass.
- Goals and objectives of a NSDIR recognize relationship between:
  - Varying categories of users and repository perspectives.
  - Varying repository perspectives and growth path.
Perspectives
- fill a niche.
- grow at different rates and paths.
- high likelihood others exist.

Growth stages
- reflect scope and role(s).
- set short-term objectives and define migration path.
- other granularities needed?

What are the activities per perspective/growth stage?

What are the critical success factors?

Are there dependencies between perspectives?

Evolutionary Framework for Repositories

Repository Growth Path

- Foundation
- Collaboration
- Stratification
- Realization
- Industrialization

Critical Success Factors

Perspectives
- less mature
- more mature

Activities
- less detail
- more detail

Repository Perspectives
- Socio-Economic
- Macro
- Micro
- Technology
- Research
Conceptual Architecture for Envisioned NSDIR

One plausible “goal” architecture 10 - 20 years from now.

- Information Base provides results, statistics, norms, benchmarks, etc. derived from Meta-level and Local Database.
- Meta-level database is a virtual database with “pointers” into other collaborating organization metrics repositories.
- Local Database represented stored and managed metrics data submitted to national repository.
- Distributed Data Repositories:
  - owned and operated by individual organizations.
  - export perspectives of their metrics data to the national meta-level.
  - may or may not contain metrics data for all perspectives.
OVERVIEW

• Introduction
• Measurement usage scenarios
• Key concepts and terminology
• Metric data categories
• Measurement system design principles
• Example measurement system: Amadeus
• Measurement transition and adoption
• Toward a national metric database (NSDIR)
• Summary
BENEFITS OF SOFTWARE MEASUREMENT

• Basis for software comparison
• Software project description, prediction, and control
• Systematic process and product improvement
• Paradigm shift to quantitative software engineering
• Understandings of fundamental nature of software
AUTOMATED SOFTWARE MEASUREMENT

MEASUREMENT ENABLES PROCESS AND PRODUCT IMPROVEMENT

• SUMMARY: Automated metric collection, analysis, reporting, graphing, and prediction system that enables software understanding and improvement

• Flexible, unobtrusive measurement system with low entry barrier

• Advantages of software measurement automation:
  • Automation of metric collection, analysis, reporting, graphing, and prediction
  • “End-to-end” metric support
  • Consistency of metric collection within and across projects and organizations
  • Comprehensive data collection based on processes, objects, and time
  • Scalable up to large projects or scalable down to small projects
  • Cost-effective metric data collection, analysis, etc.