

# Short introduction to Software Measurement

Ciprian Radu  
Software Engineer, Ropardo SRL

Based on:  
Effective Measurement of the Software Process  
by  
Prof. Eng.: Alberto Sillitti, Giancarlo Succi  
University of Bolzano, Italy,  
CASE Summer School, 2009

# Outline

- Measures
- Software development and measurement
- Personal Software Process
- Costs estimation
- PROM
- SyQL

# If you cannot measure, you can not improve!

- Software development is a combination of mutually dependent activities
- Activities often overlap in time and scope
- Good organization is the key towards success
- Q: How to decide WHAT, HOW and WHEN to produce?
- A: We can use **measures**.

# Measures

- Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules (N. Fenton, S. L. Pfleeger, 1997)
- To measure = to evaluate a problem with quantitative and qualitative analysis
- Purposes of measuring:
  - Planing
  - Managing
  - Reviewing

# You cannot control what you cannot measure

- Software development = dealing with 3 major problems (occur *before*, *during* and *after*):

	Causes	Before	During	After
Uncertainty	Difficulty of communication among different parties	Customers yet to decide what they want exactly	Unexpected problems with supporting materials (eg. libs) discovered by developers	Bugs, crashes discovered by users
Irreversibility	Time and most resources are not recoverable	Customers' definition of the expected quality cannot be reversed	System structure and unit tests; choices of design patterns and tools	Decision of allowing upgrades and the level of support provided can't be changed
Complexity	Difficulties of mastering all the details	Coordination between multiple kind of stakeholders	Lots of information kept for a project. Easy to modify and rearrange: cowboy coders	Coordination between multiple kind of stakeholders



# Measure to control software development

- Measurement helps in getting a better understanding of what is around us
  - reduces uncertainty
- We may build easily valid measures,
  - e.g.: number of non commented LOC with number of commented LOC
- but
  - Ask why we are measuring and the what we are measuring

# Measures and Agile Methods

- Collecting metrics in a traditional way is against the basic principles of the AMs
- Developers should focus on what provides value to the customer not waste time in metrics collection
- However, without measures it is not possible to follow the principles of the lean management
- So, what to do?

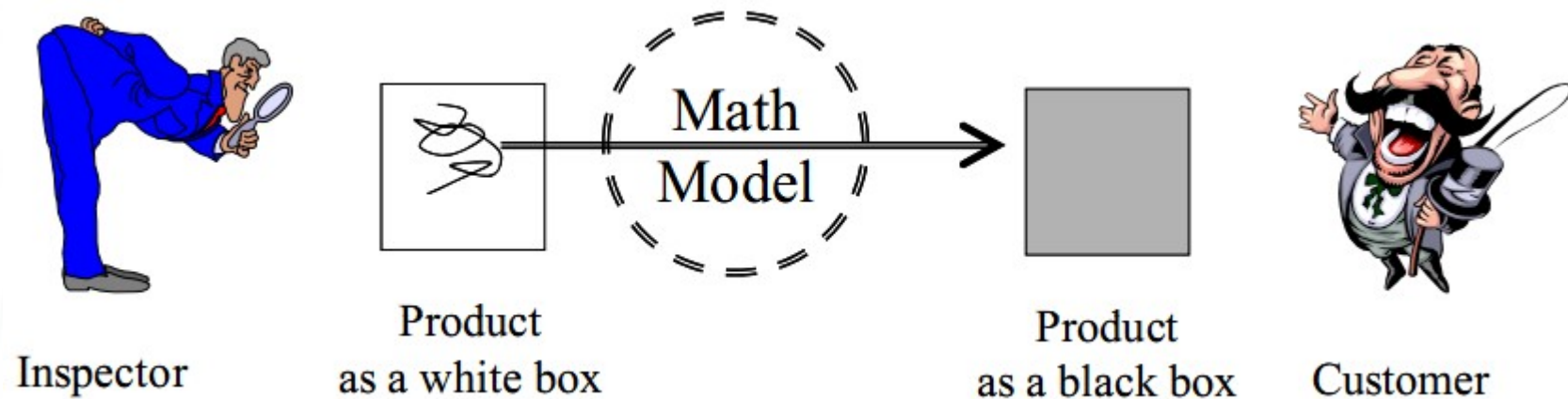
# Types of measures

- **Objective:** the same each time they are taken (automatically collected by some device)
- **Subjective:** manually collected by direct human evaluation
- e.g.: temperature of coffee  $\leftrightarrow$  taste of a coffee
- **Direct:** from a direct process of measuring
- **Indirect:** from a mathematical equation in the world of symbols



# Goal of measurement

- To select those internal attributes that can be relevant for assessing / predicting the external properties of the final system




# Prediction classes

- Class 1: from internal attributes of early life-cycle products to internal attributes of later life cycle products
  - e.g.: final size of a software system (LOC) ← size of the design document
- Class 2: from early life cycle measures of processes / resources to later life cycle measures of processes / resources
  - e.g.: implementation cost ← effort spent in formal review
- Class 3: from internal attributes of products to attributes of processes
  - e.g.: effort in testing ← complexity of the code

# Personal Software Process

- The PSP is a personal framework that software developers can apply improve:
  - Predictability
  - Quality
  - Productivity
- PSP includes of a set of methods, forms, and scripts:
  - How to plan
  - How to measure
  - How to manage their work

# Personal Software Process

- The PSP shows developers how to estimate and plan their work
  - Effort
  - Size
  - Defect

A diagram consisting of three horizontal arrows pointing to the right. The top arrow starts from the word 'Effort', the middle from 'Size', and the bottom from 'Defect'. All three arrows converge towards the word 'measurement' which is positioned to the right of the middle arrow.
- The keys to making better estimates and plans are
  - relevant historical data
  - statistically sound methods



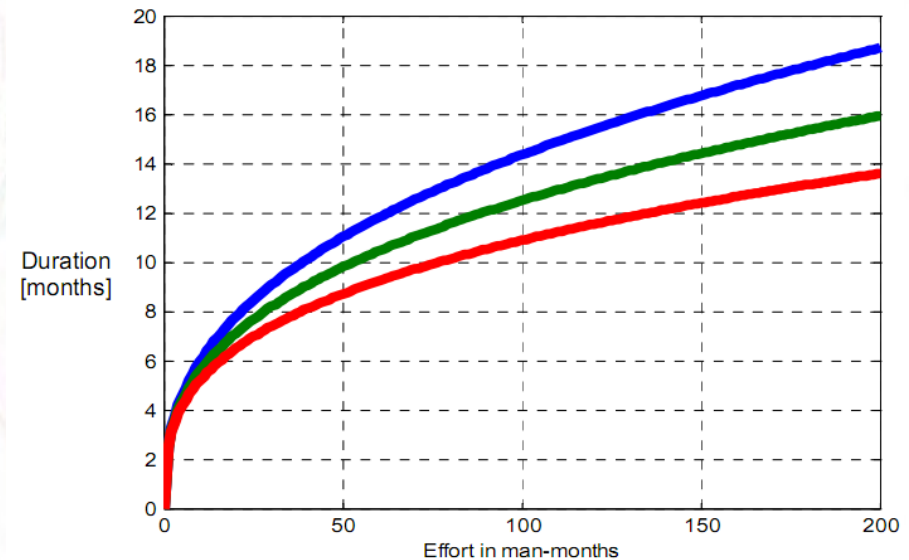
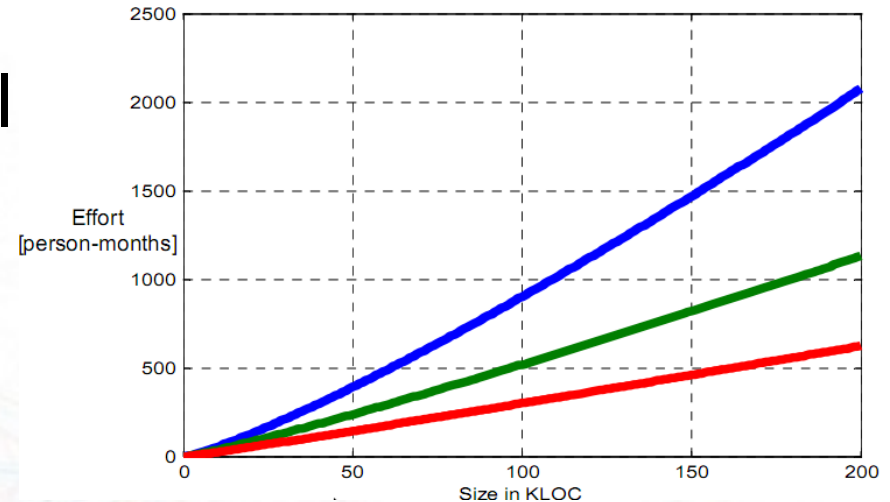
# Cost Estimation

- Project Costs = Hardware + Effort + Training + Traveling +  
+ ...
- Common mistakes
  - Underestimating of time and effort required
  - Imprecise and changing requirements
  - Insufficient experience and human bias
- Cost estimation methods
  - Mathematical models
  - Expert opinion based on personal experience
  - Analogy from previous projects

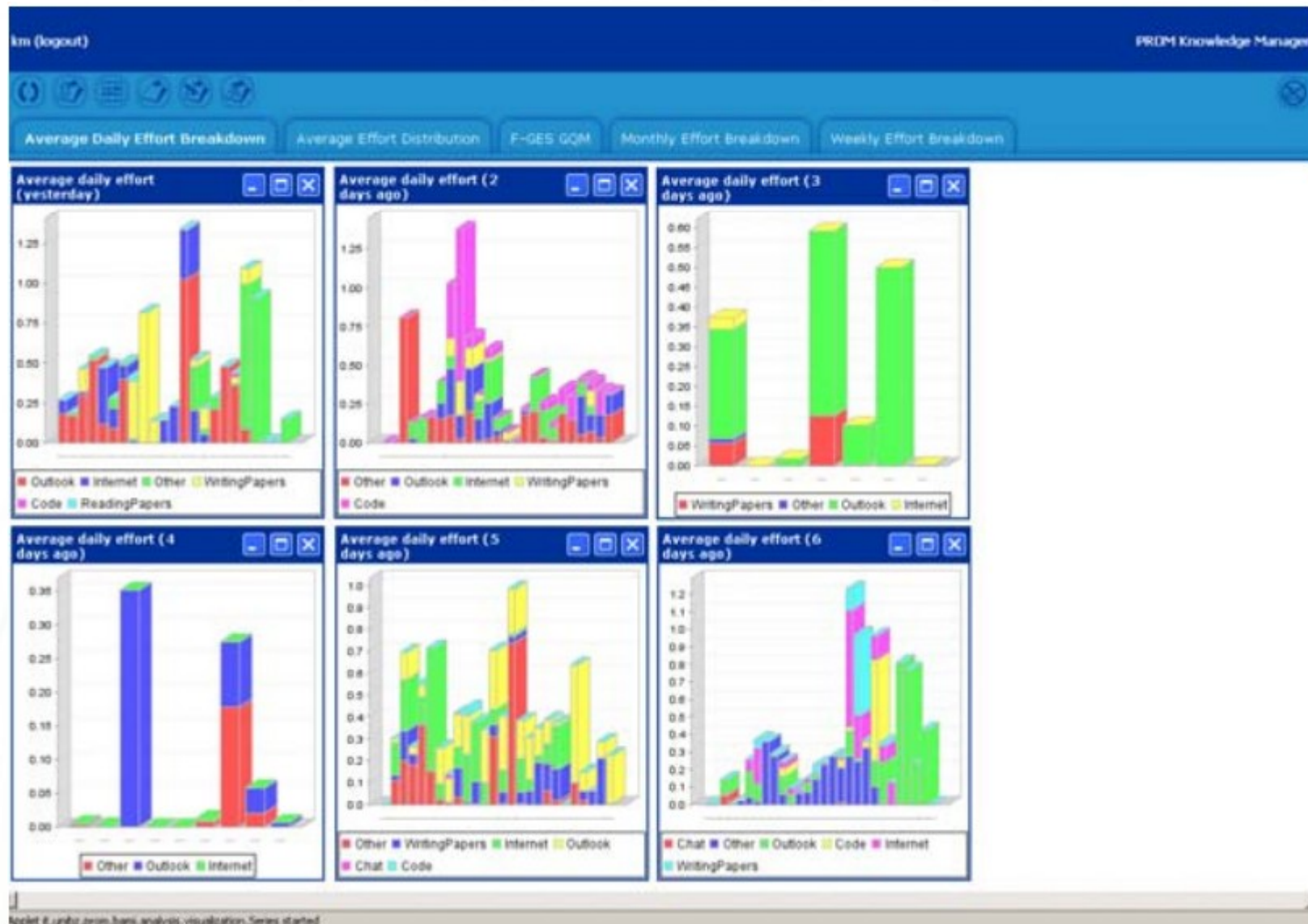


# COCOMO

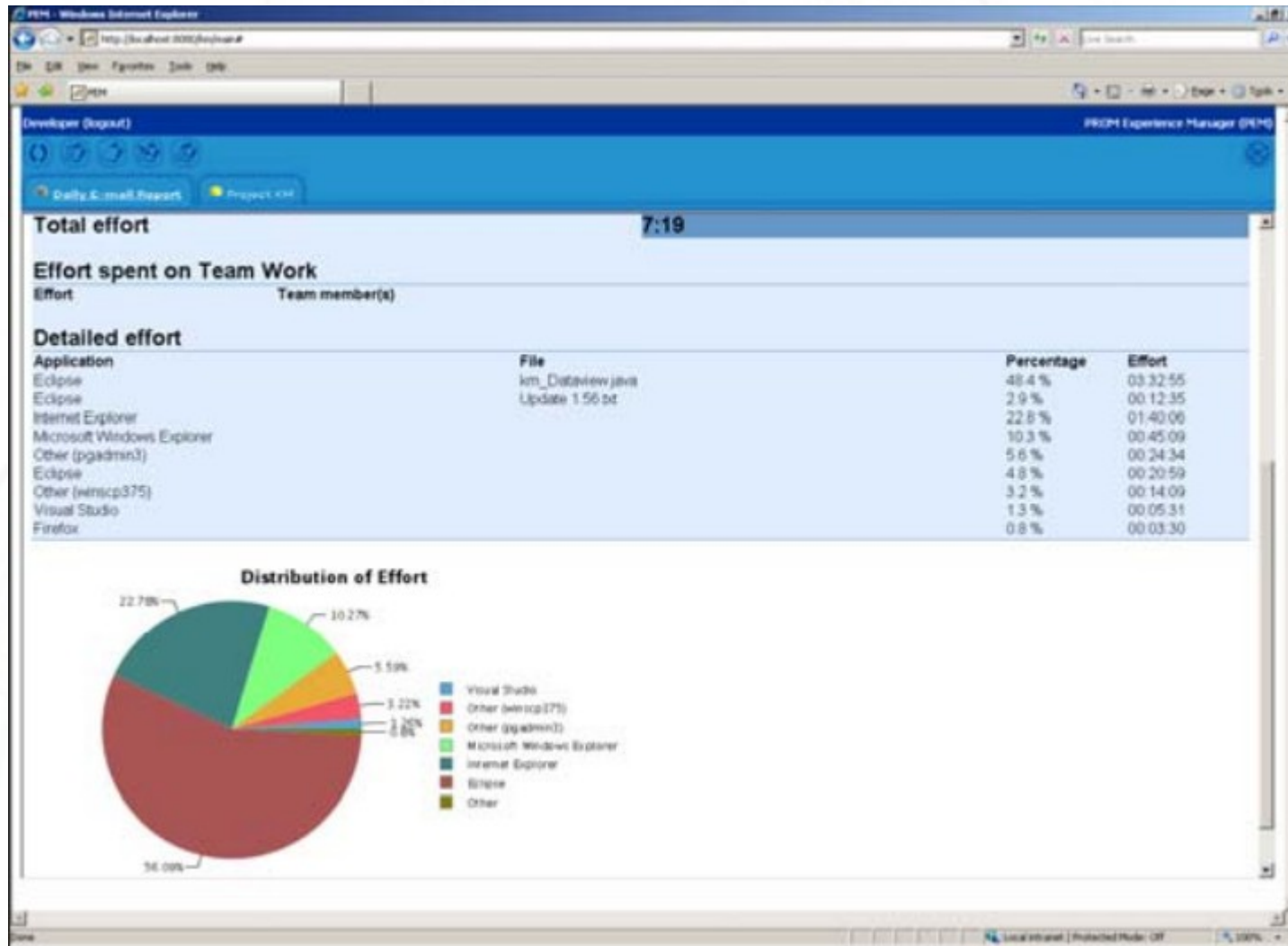
- COnstructive COst Model
- Effort estimation
- Duration estimation



# PROM



# PROM



# SyQL

- **S**ystem **Q**uery **L**anguage
- Data manipulation language to analyze fuzzy-temporal databases of software artifacts
- SyQL is designed to address the problem of information retrieval on a data warehouse that contains software metrics, software structure and effort data



# SyQL

FROM **Class c, Method m** ← Concept declarations

WHERE **c.getFullName() = m.getDefClassFullName()** ← Method Call **AND**

**c.getEffort( YESTERDAY ) IS High** ← Fuzzy Equal Condition

SELECT **c.getFullName(),**

**c.getEffort( TODAY - 1'day' ),** ← Temporal expression

**COUNT(m)** ← Aggregate Function

GROUP BY c.getFullName();



# Summary

- Measures in context of software development
- Data collection techniques
  - Personal Software Process
- Costs estimation
  - COCOMO
- PROM
- SyQL
  - fuzzy-temporal query language against a metrics data warehouse