



# Measuring and Improving Productivity

*Dr. Christof Ebert, Alcatel  
SET 2006, Zurich, 09. May 2006*




All rights reserved © 2006, Alcatel

## What Do We Address Here?

Page 2

- ➔ Part 1  
What is Software Productivity?
- Part 2  
How to Measure Productivity?
- Part 3  
How to Improve Productivity?
- Conclusions



CTO, R&D Effectiveness  
C.Ebert, May 2006, productivity-metrics.ppt

All rights reserved © 2006, Alcatel

## Why Bother About Productivity?

Page 4

ICT has been and will be a **major driver of productivity growth**.

Virtually all technologies will become information technologies.

- Improving automation in all areas of economy
- Facilitating information search and flow
- Person-based knowledge management
- International allocation of resources

**BUT:** "Past growth had been achieved mostly by hardware, while software is still a craft industry."

Sources: OECD 2004, European Commission 1999, Kurzweil 2005, Gartner 2004

## What is Productivity?

Page 5

Productivity is defined as **Output over Input**.

### Output:

- the **value delivered**.

### Input:

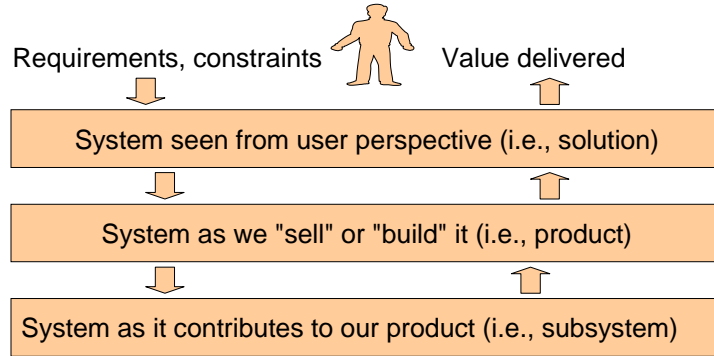
- the **resources** (e.g., effort) spent to generate the output,
- the **influence of environmental factors** (e.g., complexity, quality constraints, time constraints, process capability, team distribution, interrupts, feature churn, tools, design)



# The Value Perspective Creates a Systems View

Page 6

**Value exists only in the eyes of the beholder** (i.e., customer)



**Recursion applies** (i.e., your solution is your customer's subsystem)

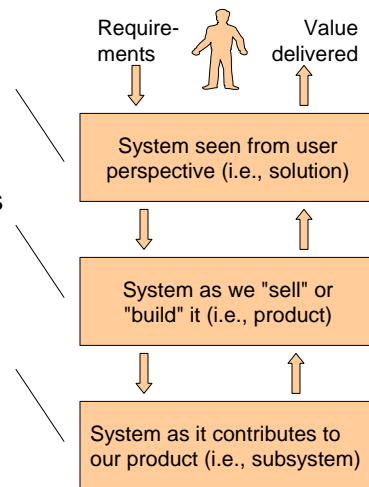
# Productivity: A Systems Perspective

Page 7

1. **Productivity for which customer pays for.** Relates to solution and tangible value. Independent of influences of lower levels.

2. **Productivity within supplier.** This relates to product development, thus defined as functional size over engineering effort.

3. **Productivity of subsystems for which supplier pays for.** We decide make, reuse, buy to improve productivity on next higher level.



# Productivity Directly Influences Feasibility

## Quantitative planning inputs:

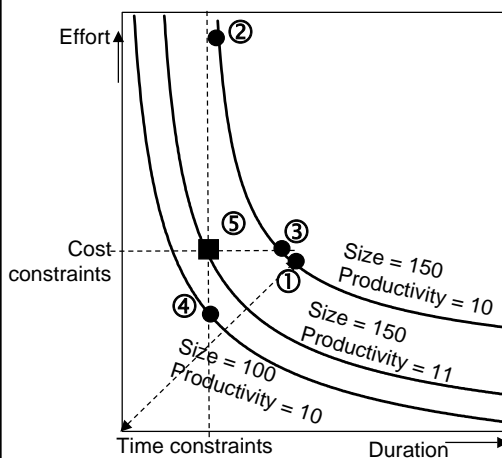
- Productivity expectation affects estimates of new projects
- Actual performance validates estimates (or not)
- Own experiences help refine productivity expectations

## Project feasibility and project set-up

- Project size and productivity → What are optimal sizes?
- Project complexity and productivity → Which environmental factors (maturity, time pressure, etc.) affect performance?
- Percentage outsourced and productivity → At which level of outsourcing do we see optimal project performance?

# Example: Planning MUST Consider Productivity

## Project Scenarios ① to ⑤



Based on Raleigh distribution  
(„Putnam formula“):

$$\text{effort} \sim \frac{\text{size}^3}{\text{productivity}^3 * \text{time}^4}$$

Scenarios:

- ① Optimized solution  
(smallest distance to curve)
- ② Effort optimized for schedule target  
(hier: Zielwert wird nicht erreicht)
- ③ Time optimized for cost target
- ④ Within time and cost constraints  
(reduced size)
- ⑤ Within time and cost constraints  
(increased productivity)

# What Do We Address Here?

Part 1  
What is Software Productivity?



Part 2  
How to Measure Productivity?

Part 3  
How to Improve Productivity?

Conclusions

# Before Asking the How, Let's Understand Our Own Goals

- **Reduce cost**  
e.g., Develop more valuable products for lower costs
- **Support estimation**  
e.g., Provide consolidated and parameterized history data
- **Benchmark / assess design center or supplier capability**  
e.g., Rationalize higher capital-to-staff investments
- **Optimize processes**  
e.g., Early defect correction
- **Streamline software operations**  
e.g., Identify production bottlenecks or underutilized resources



## Using GQM: Goals and Questions around Productivity

Page 13

- Reduce cost
  - How can we improve productivity?
  - What is the value-add of getting to higher maturity levels?
- Support estimation
  - Are there simple rules of thumb estimation mechanisms?
  - Project schedules: do we overcommit?
  - Outsourcing contracts: will we get what we ask for?
- Benchmark / assess design center or supplier capability
  - What can we learn from those who are best in class?
  - How to select a supplier on more than initial contract offer?

## Attention: Mere GQM Considered Harmful

Page 14

Goal-oriented measurement has become the mantra of practically all measurement discussion.

- Goals are important, because often metrics are defined without understanding business needs and overarching drivers.

However, mere GQM bears risks

- Too much "top-down"
- One-dimensional, unbalanced
- Too much theory



**Combine goal-orientation with decision-support and other operational management techniques.**

# The Essential Difficulty of Productivity Measurement

Page 15

Based on what we learned so far, we deduct:

- Productivity is a derived metric:  
**function (output, input, environment)**
- Input is easy to measure (e.g., activity-based controlling)
- Output and environment are difficult to describe and measure.



## Output

Page 16

### Output is delivered value

- Value exists only in the **beholder's perspective**

Output and environment:

- delivered source statements, function points, components, documents, artifacts, ...
- with a certain quality, complexity, ...
- in a certain environmental setting such as skills, pressure, tool support, computing platform, frequency of requirements changes, ...
- having created application-domain and technical knowledge

## Typical Productivity Measurements –1

Page 17

Size over effort.

- Most popular because of ease of calculation.
  - “It I an imperfect measure of productivity but one that could be measured consistently” (MacCormack et al, 2003)
- Problems:
  - The more verbose the programmer, the more productive.
  - The lower the programming language, the higher is productivity
  - Does not consider environmental complexity and impacts.
- Alternative sizing:
  - Function points, requirements, use cases.
  - IEEE Std 1045 Standard for Software Productivity Metrics provides a framework for measuring and reporting software productivity. It is meant for those who want to measure the productivity of the software process in support of their software product.

## Typical Productivity Measurements –2

Page 18

Function Points over effort.

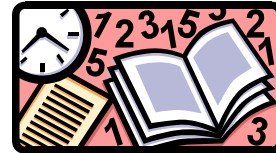
- Motivation: Functionality captures “value delivered” better than size
- Based on a combination of program characteristics, each associated with a weight
  - external inputs and outputs;
  - user interactions;
  - external interfaces;
  - files used by the system.
- The total function point count is modified by complexity of the project
- Problem: FPs are more subjective than LOC
- A  $FP^x$  approach is used to indicate productivity

## Typical Productivity Measurements –3

Page 19

### Embedded techniques

- Productivity = Adjusted size / effort  
Adjusted size is the estimated effort based on history and constraints.  
Productivity is a normalization comparing estimated to actual effort
- QSM SLIM, CoCoMo or SPR KnowledgePlan  
Relates size to effort ...  
... and expands this to a “Productivity Index” that encompasses the general development environment and time pressure
- Earned value.  
Weighted effort of completed work products versus planned/realized effort.



**A lot of possible productivity measurements to choose from.**

## Why “Output” Is Difficult –1

Page 20

Example:

Embedded controller with sonar system

Two different programs with exactly the same functionality

```
BSP1: PROC;  
TAKE [X,Y] FROM INTERFACE;  
IF (X > 0)  
THEN  
IF (Y > 0)  
THEN  
SEND '00'B1 TO LENKUNG;  
ELSE  
IF (Y < 0)  
THEN  
SEND '01'B1 TO LENKUNG;  
ELSE /* Y = 0: DEFAULT */  
SEND '00'B1 TO LENKUNG;  
FIN;  
FIN;  
ELSE  
IF (X < 0)  
THEN  
IF (Y > 0)  
THEN  
SEND '10'B1 TO LENKUNG;  
ELSE  
IF (Y < 0)  
THEN  
SEND '11'B1 TO LENKUNG;  
ELSE /* Y = 0: DEFAULT */  
SEND '00'B1 TO LENKUNG;  
FIN;  
FIN;  
ELSE /* X = 0: DEFAULT */  
SEND '00'B1 TO LENKUNG;  
FIN;  
FIN;  
END: /*BSP1*/
```

```
BSP2: PROC;  
DCL QUADRANT BIT(2);  
TAKE [X,Y] FROM INTERFACE;  
IF (X**Y == 0)  
THEN /* X=0 oder Y=0: DEFAULT */  
QUADRANT:= '00'B1;  
ELSE  
IF (X > 0) /* Bit 1 setzen */  
THEN  
QUADRANT.BIT(1):= '0'B1;  
ELSE  
QUADRANT.BIT(1):= '1'B1;  
FIN;  
IF (Y > 0) /* Bit 2 setzen */  
THEN  
QUADRANT.BIT(2):= '0'B1;  
ELSE  
QUADRANT.BIT(2):= '1'B1;  
FIN;  
FIN;  
SEND QUADRANT TO LENKUNG;  
END: /*BSP2*/
```

## Why “Output” Is Difficult –2

Page 21

Assumption : Same effort to design these two programs

- LOC based sizing:  
Counting the executable source code lines  
=> Code 1 has 50% higher productivity.
- Function Points:  
Functionality of both program segments is equal  
=> Productivity of both program segments is equal  
(assuming that the outputs are always combined)
- Feature Points:  
Double amount of decisions in code 1 means higher algorithmic complexity  
=> Code 1 has higher productivity
- Project productivity:  
Double amount of decisions in code 1 means higher test effort to achieve same C1 coverage  
=> Code 1 has lower productivity

## Benchmarking with Productivity Data

Page 23

Benchmarking uses lots of productivity data

- “better”, “faster”, “cheaper” must be **translated to a hard currency** (i.e., €)
- We do not need primarily reports or slogans, but **ways to learn and to improve**

### Challenges:

- Compare apples and apples  
All things equal applies hardly
- Avoid the traps of scientific hype (my theory is better) and industry hype (we can do it better)
- Relate productivity gains to the actual change and not to a mixture of changes overlaid with Hawthorne-like effects, scope changes, restructuring, cost reduction, etc.



## Case Study: Benchmarking with Productivity Data –1

Page 24

An initial study was published in 1984 by Capers Jones when he worked in ITT Advanced Technology Center (today Alcatel)

- Systematic data on programming productivity, quality, and cost was collected from 44 switching related projects in 17 corporate subsidiaries in 9 countries, accounting for 2.3M LOC and 1500 person years of effort.
- *Finding:* product-related and process-related factors account for approximately the same amount (~33%) of productivity variance.
- *Finding:* **you can distinguish productivity factors that can be controlled (process-related: accidental) from those that cannot (product-related: essential).**

## Case Study: Benchmarking with Productivity Data –2

Page 25

### **Process-related factors (more easily controlled)**

- multiple dependencies (e.g., HW-SW co-development)
- engineering environments
- requirements churn
- processes and tools
- staff, team composition

### **Product-related factors (not easily controlled)**

- nonfunctional constraints
- solution complexity
- customer participation
- size of product

## Case Study: Benchmarking with Productivity Data –3

Page 26

ISBSG (Int. Software Benchmarking Standards Group)

- Not-for-profit organization founded by several software measurement organizations (e.g., NASSCOM, IFPUG, etc.)
- [www.isbsg.org](http://www.isbsg.org)
- The ISBSG Repository claims to represent the best performing 25% of industry projects. It allows to benchmark projects against this history.
- “Reality Checker” for own project data

Step-1 Determine the software size of the intended application

Step-2 Enter the relevant project parameters into the Reality Checker  
i.e., Project Size (functional units), Platform (MF, MR, PC), Language (3GL, 4GL, Application Generator), Type (development type - New, Re or Enhancement)

Step-3 Conduct a ‘reality check’ on the two primary expectations (i.e., Total developer effort (Project Work Effort), Elapsed Time (months))

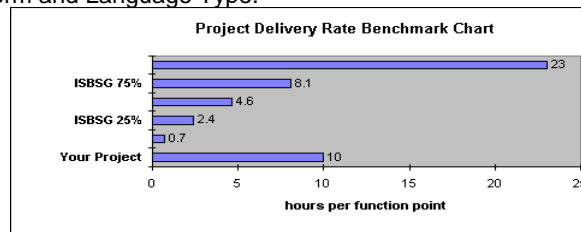
Step-4 Interpret the results and decide intended actions (if any)

## Case Study: Benchmarking with Productivity Data –4

Page 27

Usage:

- Project parameters
  - Development Platform = Mid-range,
  - Methodology = Developed in-house
  - Language Type = 4GL,
  - Maximum Team size > 8,
  - Your Project Delivery Rate, (PDR), was 10.0 hours per function point
- For the two factors with the most significant impact on productivity, Development Platform and Language Type, the chart below shows how your Project Delivery Rate compares to projects with the same Development Platform and Language Type.



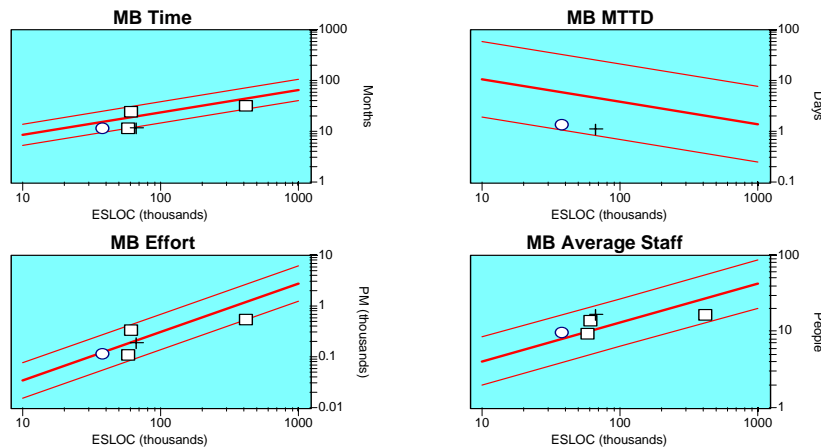
Source: ISBSG,  
Reality Checker,  
2005

# Case Study: Benchmarking with Productivity Data –5

QSM (Quantitative Project Management) or SPR

- Commercial enterprises
- [www.qsm.com](http://www.qsm.com) (SPR: [www.spr.com](http://www.spr.com) )
- SLIM Estimate, SLIM Control tools, SPR KnowledgePLAN
- Offers a variety of tools (such as estimation, project control, benchmarking, bid evaluation, supplier management, project / portfolio metrics suite) and services (training, consulting)
- QSM and SPR databases are proprietary and contain >10000 projects.

# Case Study: Benchmarking with Productivity Data –6




# What Do We Address Here?

Page 30

Part 1  
What is Software Productivity?

Part 2  
How to Measure Productivity?

 Part 3  
How to Improve Productivity?

Conclusions

# Hardware and Software

Page 32

The anomaly is not that software progress is so slow, but that computer hardware progress is so fast

No other technology since civilization began has seen seven orders of magnitude price-performance gain in 50 years

In hardly any technology can one choose to blend the gains from improved performance and reduced costs

**We cannot expect to see two-fold productivity gains every two years in other engineering fields.**



# Software: There Ain't Be No Silver Bullet

Page 33

**There is no single development,  
in either technology or  
management technique,  
which by itself promises even  
one order-of-magnitude  
improvement within a decade in  
productivity,  
in reliability,  
in simplicity.**

- Fred Brooks, 1975



# Is Software Really so Different?

Page 34

Some schools of thought argue that software productivity improvement is **impossible due to "natural laws"**.

They claim software inherent difficulties as not solvable:

- Complexity (not two parts are alike, large number of states and interactions, nonlinearities, communication difficulties)
- Conformity (embedded into many other systems to which it has to conform, interfaces)
- Changeability (SW changes easily but with unknown ripple effects, pressure to change especially after delivery)
- Invisibility (SW is not structured, impossible to draw, difficult to communicate)



Source: Gail Kaiser: Advanced Software Engineering, Lecture, March 2005

## A Personal Advice

Page 35

**BUT: Instead of blaming “natural laws” let’s rather learn from “natural intelligence” how to deal with complexity.**

Examples:

- Try to model the distributed intelligence of termites, which build huge settlements without the central architect and planning
- Try to extract the conformance requirements in swarms which interact and interface in multiple dimensions
- Try to describe change management of viruses, such as cholera which adapts to all type of environmental conditions

All these trials must fail because we impose accidental models – and still it works in nature perfectly.

## Proof of Concept: Productivity Can Be Improved!

Page 36

Remove **accidental** barriers

- Engineering and management discipline, processes and tools
- Standards – from cradle to grave (languages, templates, IDEs, etc.)
- Design to quality, change, cost, ...
- Lean concepts: smaller teams, components, iterations, etc.

Reduce **essential** barriers

- Domain understanding
- Modelling languages with “unified” understanding
- Self-generating and self-updating software
- Reuse of components

# What Do We Address Here?

Page 38

Part 1  
What is Software Productivity?

Part 2  
How to Measure Productivity?

Part 3  
How to Improve Productivity?



Conclusions

# Software Productivity Laws

Page 39

**LAW 1 – *Reduce essential and accidental complexity***

Manage risks and reduce surprises. Deliver added value with increments. Stop this firefighting and replace it with accountability. (i.e., your mom was right with that discipline thing)

**LAW 2 – *SOME schedule compression can be bought***

Schedule can be reduced – to a point – by adding people and putting more pressure (i.e., those MBA guys have their point).

**LAW 3 – *Better technology and processes improve performance***

Any project can excel if engineered and managed intelligently (i.e., CMMI has direct productivity impact).

**LAW 4 – *Work expands to fill the available volume***

Plan and track earned value. Don't put too much extra buffers on the critical path. (i.e., Parkinson's law).

## Software Productivity Laws (continued)

Page 40

### LAW 5 – *Staffing can be optimized*

Small collocated teams cross-fertilize and boost the productivity of each individual person (i.e., agile development)

### LAW 6 – *Follow where the money goes*

The biggest impact on productivity improvement has a profound process analysis. Look on cost distribution. Reduce your cost of non-quality. (i.e., these consultants have their value after all)

### LAW 7 – *No “silver bullets”*

There is no methodology, tool, or process improvement strategy out there that yields revolutionary improvements in project efficiency (i.e., sales prospects are often hype).

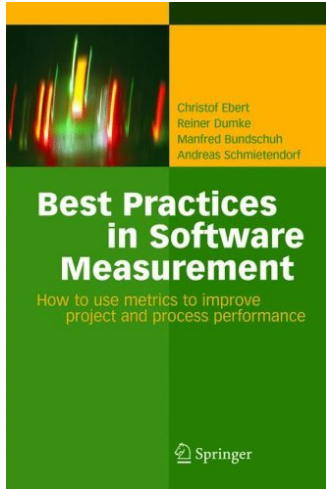
## Conclusions

Page 41

- It is vital in today business climate to **understand and improve your productivity**.
- Productivity is impacted by external factors, such as pressure, outsourcing, requirements changes, etc. **Avoid comparing or benchmarking things which are different!**
- As an organization, you must seek to **build up a normalized expertise** in productivity analysis.
- Effectively **work with productivity benchmarks** to scrutinize project plans and assess subcontracting offers
- **Own development and sourcing can be optimized** by first understanding your own productivity and then evaluating productivity and costs of external / internal projects.

# Measurement Reading

Page 42



## Best Practices in Software Measurement

How to use metrics to improve project and process performance

By: Christof Ebert, Reiner Dumke, et al  
296 Seiten  
Springer, 2004.  
ISBN: 3-540-20867-4

A useful and relevant introduction to measuring in software and IT. The book contains tons of practical guidance which allow fast deployment to concrete needs. With their different backgrounds in measurement and practical implementation the four authors contribute towards practical relevance.

# THANK YOU!

B R O A D E N   Y O U R   L I F E

[www.alcatel.com](http://www.alcatel.com)