Software Metrics

Software Engineering 2008

Anton Litvinenko

- Co-founder and CTO at 

  - Automated analytics for remote software development
  - Key competence: software metrics and indicators
- 5 years of software development at Mobi and MicroLink
- MSc in computer science at Tartu University

Are software metrics “good” or “bad”?

“You can’t control what you can’t measure”

Tom DeMarco
“Controlling Software Projects”

Measurement

- Assignment of quantitative indications to object's attributes
  - “Mapping from real world to numbers”
  - Preserving empirical relation
- Quantitative attribute – “*objective*”
  - Categories, numbers
  - Stand in relation to one another
- Not quantitative - *“subjective”*
  - Feelings, emotions

Example: Height

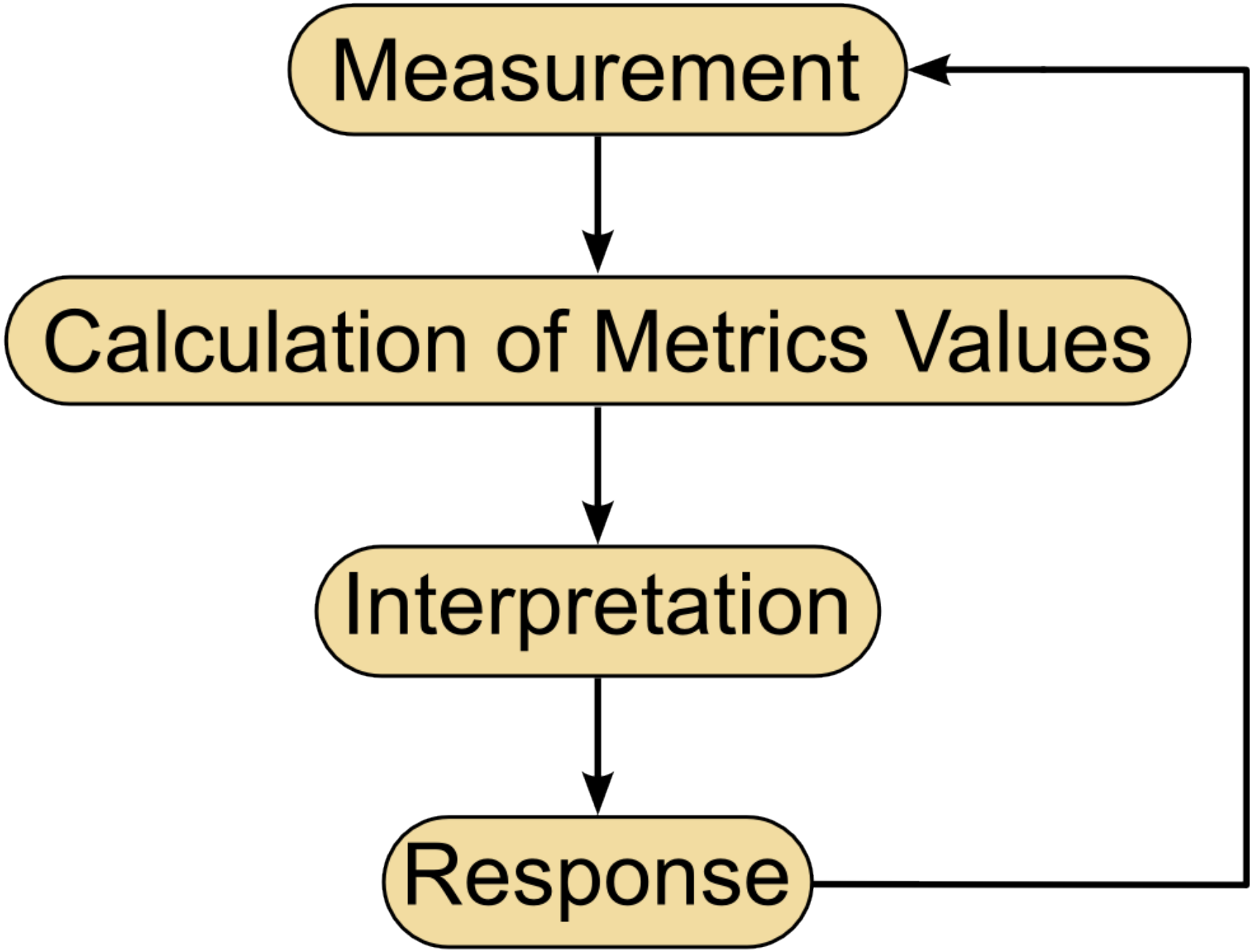
- Subjective:
  - I am tall
- Empirical relation:
  - I am higher than Bill Gates

Example: Height

- Subjective:
  - I am tall
- Empirical relation:
  - I am higher than Bill Gates
- Measurement:
  - I am 192cm
- Relation is preserved:
  - Bill Gates is 178cm

Metrics – quantitative representation of otherwise vague attributes

GOAL – IMPROVEMENT


Measurement
Calculation of Metrics Values
Interpretation
Response

Applications Of Metrics in Software Development

- Project progress/cost estimation
  - Product size, velocity, ...
- Evaluation of tools and methods
  - Growth of productivity and quality, ...
- Product quality evaluation
  - Quality of design and code, number of bugs, ...
- Provide feedback to project team
  - Developer reputation, ...

Can anybody name any software metric?

Lines Of Code (LOC) – Product Size

```
1  public static void main(String args[]) {
2
3      final SalesDomainController domainController =
4          new SalesDomainControllerImpl();
5
6      if (args.length == 1 && args[0].equals("console")) {
7          // a small console UI
8          ConsoleUI cui = new ConsoleUI(domainController);
9          cui.run();
10     } else {
11         // Swing UI
12         final SalesSystemUI ui = new SalesSystemUI(domainController);
13
14         ui.setVisible(true);
15     }
16
17     log.info("SalesSystem started");
18 }
```

- 12
- 14
- 18

Lines Of Code

```
1  public static void main(String args[]) {
2      /*
3          Creating instance of the controller.
4          Some more comments...
5      */
6      final SalesDomainController domainController =
7              new SalesDomainControllerImpl();
8      if (args.length == 1
9              && args[0].equals("console")) {
10         ConsoleUI cui =
11                 new ConsoleUI(domainController);// a small console UI
12         cui.run();
13     } else {
14         // Swing UI
15         final SalesSystemUI ui =
16                 new SalesSystemUI(domainController);
17         ui.setVisible(true);
18     }
19     log.info("SalesSystem started");
20 }
```

Lines Of Code – Summary

- Accurate, easy to measure

- How to interpret ...
  - Empty lines
  - Comments
  - Several statements on one line
- Language dependent
- Doesn't respect complexity and content

Function Points (FP) – Product Size

- Evaluation of use-cases
- **Step 1:** Screen each use-case to identify:
  - User inputs (information input):
    - *add new purchase, search, wizard*
  - User inquiries (no derived data, data retrieval):
    - *view customer data,*
  - User outputs (includes derived data, algorithms):
    - *monthly purchase breakdown, notification message*
  - Files: *db table with customer data*
  - External Interfaces: *integration with warehouse*

Function Points

- **Step 2:** Assign complexity for each component
- **Step 3:** Calculate number of unadjusted points:

	Count		Simple	Average	Complex		
User inputs	___	x	3	4	6	=	___
User outputs	___	x	4	5	7	=	___
User inquiries	___	x	3	4	6	=	___
Files	___	x	7	10	15	=	___
External interfaces	___	x	5	7	10	=	___
Number of unadjusted points (**unadjusted-total**)						=	Sum()

Function Points

- **Step 4:** Calculate complexity adjustment values: $\mathbf{Sum(F_i)}$

F_i:

- Complex processing → _
- Distributed data processing → _
- Performance → _
- Heavily used configuration → _
- Online data entry → _
- Distributed transactions → _
- Installation ease → _
- ...

- No Influence – 0
- Incidental – 1
- Moderate – 2
- Average – 3
- Significant – 4
- Essential – 5

Function Points

- **Final**: Calculate total functions points:

$$FP = \text{unadjusted-total} \times (0.65 + 0.01 \times \text{Sum}(F_i))$$

Function Points – Summary

- Independent of language
- Possible to apply early in the project
  - No source code required

- Manual
- Complexity estimation is subjective
- No physical meaning

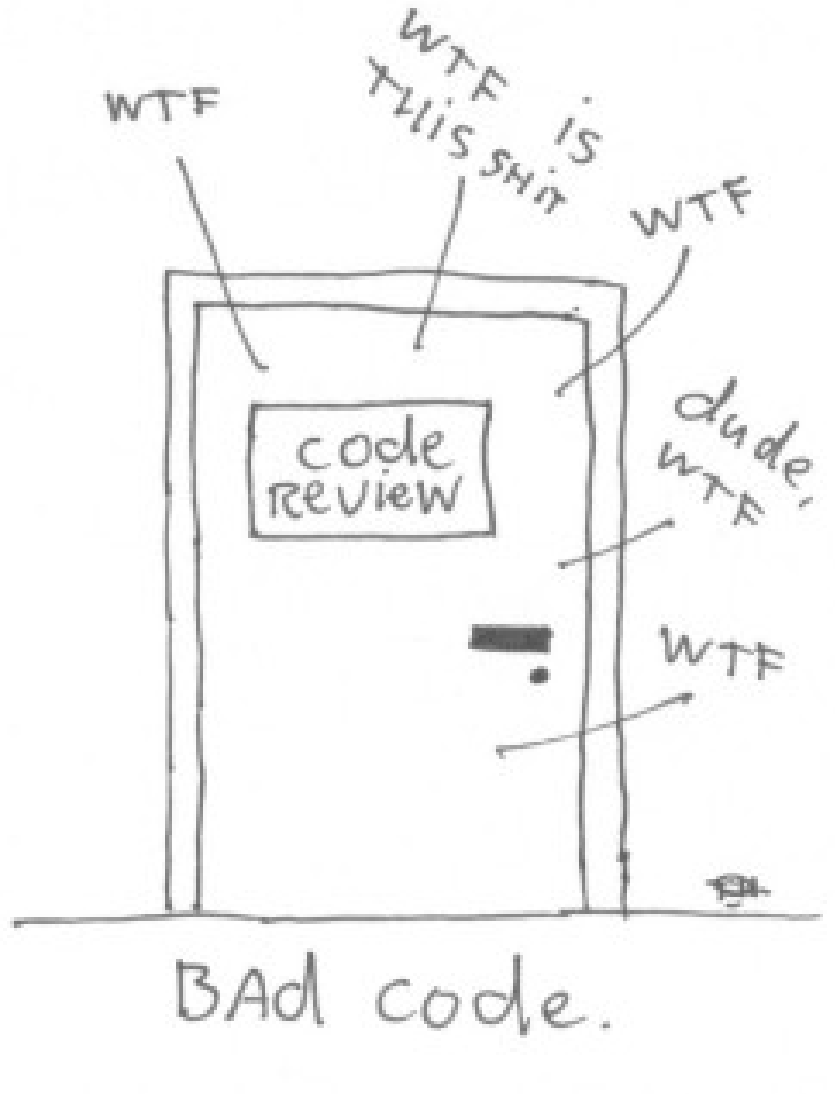
LOC and FP

- Size and productivity estimation and evaluation

- Last project
  - 1337 FP-s and 24 man-months

- New project
  - 14230 FP-s → ~ 240 man-months

The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE



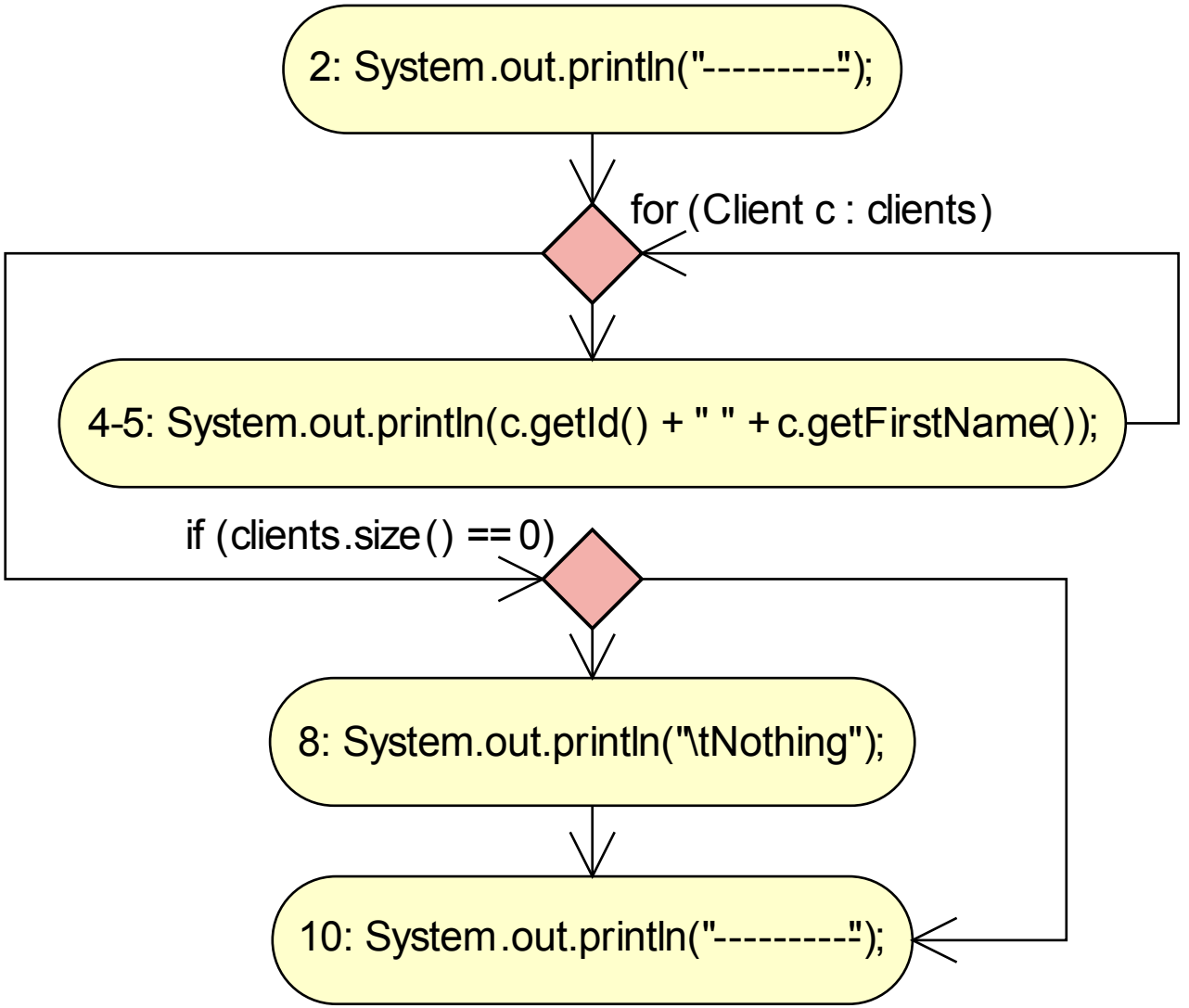
McCabe's Cyclomatic Complexity

- Thomas McCabe, 1976
- Complexity of a program
  - Number of linearly independent paths through a function
  - Usually calculated using flow graph
- **`V(G) = e - n + 2p`**
  - **`e`** – num of edges, **`n`** – num of vertices, **`p`** – num of unconnected parts of graph

McCabe's Cyclomatic Complexity

```
1  private void showClients(List<Client> clients) {
2      System.out.println("----------------------");
3      for (Client c : clients) {
4          System.out.println(c.getId() + " "
5                  + c.getFirstName());
6      }
7      if (clients.size() == 0) {
8          System.out.println("\tNothing");
9      }
10     System.out.println("----------------------");
11 }
```

McCabe's Cyclomatic Complexity



- `e = 7`
- `n = 6`
- `p = 1`
- `V(G) = 3`

Cyclomatic Complexity – Summary

- Automated
- Maintainability
  - V(G) > 10 → Probability of defects rises
- Testability
  - V(G) is an upper bound for the branch coverage
    - Each control structure was evaluated both to true and false
  - V(G) is a lower bound for the path coverage
    - All possible paths were executed
- Doesn't respect other types of complexity
  - Data structure, data flow, interfaces

Does usage of software metric help me to be a better engineer?

Metrics At Technical Level

- Immediate benefit
  - Design improvement
    - Identify code smells
  - Development effort planning
    - Identify most error prone components
    - Identify most likely to change requirements
    - Plan defect fixing effort in advance

- *Toolkit: Eclipse + Metrics extension*

Does usage of software metric help me to be a better project manager?

Metrics At Project Level

- Time to benefit: weeks → current state?
- Continuous monitoring
- Delivering the product
  - Overall quality
  - Meaningful estimates
  - Teamwork

- *Toolkit: Ohloh, Programeter, 6th Sense Analytics*

Does usage of software metric help me to be a better boss ?

Metrics At Organization Level

- Time to benefit: months → current state?
- Continuous monitoring
- Strategic thinking
  - Improvement of software development process
  - Decisions to offshore/outsource
  - Know-how management

- *Toolkit: Programeter, Yammer, 6th Sense Analytics*

Classification of Software Metrics

Subject of measurement

Subject: Development Process

- Measuring the efficiency of process application

- Examples of metrics
  - Length of (development) iteration
  - Number of changes in requirements
  - Number of finished tasks

Subject: Resources

- Measuring usage of resources and their properties

- Examples of metrics
  - Developer competency
  - Developer fluctuation
  - Developer productivity and know-how in the project
  - Maturity of the code written by developer

Subject: Product

- Measuring product attributes
  - Size, complexity, scalability
- Examples of metrics
  - LOC, commented lines of code, function points
  - McCabe's cyclomatic complexity
  - Code coverage with test
  - Code stability

Classification of Software Metrics

"Lines of Code" vs "Quality"

Direct Metrics

- Directly measurable

- Examples of metrics:
  - LOC, function points,
  - McCabe's cyclomatic complexity
  - Number of requirements

Indirect Metrics

- Not possible to measure directly
  - Derived from other properties

- Examples of metrics
  - Code quality, code readability
  - Developer productivity, efficiency
  - Reliability

Classification of Software Metrics

(In)dependency on the measurement context

Internal Attributes

- Measurement context/environment is not relevant

- Examples of metrics
  - LOC
  - McCabe's cyclomatic complexity
  - Code coverage with tests

External Metrics

- Measured with respect to environment/context

- Examples of metrics
  - Software reliability
  - Developer productivity
  - Source code comprehensibility
  - Usability

Is it possible to evaluate a developer using a single metric?


OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
S. Adams E-mail: SCOTTADAMS@AOL.COM
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
© 1995 United Feature Syndicate, Inc.(NYC)
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!

Criticism

- People evaluation by numbers
  - Emotions, people adjust
- Easy to misuse
- Accurate vs meaningful
  - LOC is accurate, but meaningless
  - FP is meaningful, but not accurate

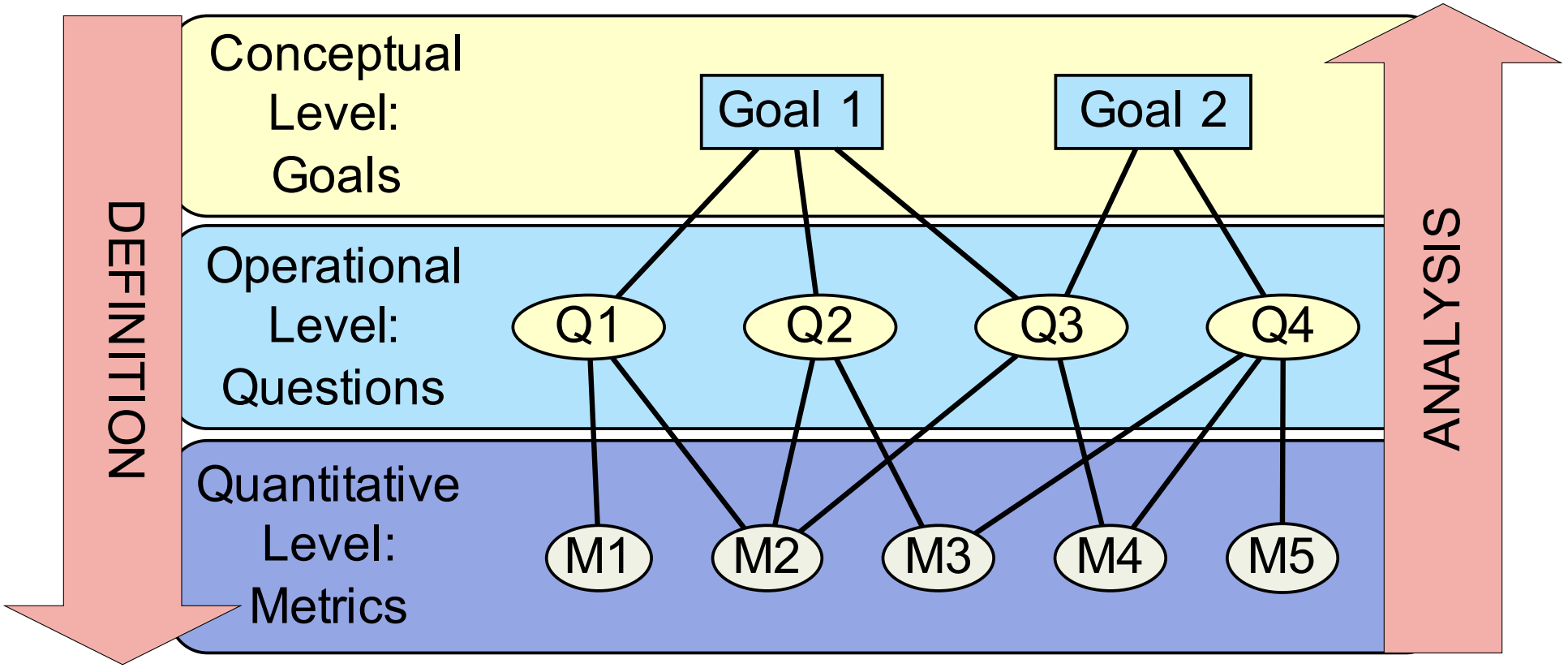
- Solution: set up clear goals and choose corresponding metrics → frameworks

Software Metrics Frameworks

- GQM:
  - Goal – question – metric
- PSM:
  - Practical software measurement
- AMI:
  - Application of metrics in industry
- ...

Goal – Question – Metric

- Victor Basili



“A good guideline is that measures of individual productivity give you questions to ask but they don’t give you the answers”

Steve McConnell

Software Metrics – Summary

- Metric is ...
  - Goal: ...
  - The most important aspect of application: ...

- If you are interested in software measurement and metrics:

Anton Litvinenko

http://programeter.com

email: anton.litvinenko@programeter.com

skype: anton.a.litvinenko

References

- Wikipedia
  - http://en.wikipedia.org/wiki/Software_metrics
- C. Lange, Metrics in software architecting
  - http://www.win.tue.nl/~mchaudro/sa2007/Metrics%20Architecting%202005.pdf
- M. Gökmen, Software process and project metrics
  - http://www3.itu.edu.tr/~gokmen/SE-lecture-2.pdf
- H. Nestra, Metrics, Software engineering 2005
  - http://courses.cs.ut.ee/2005/tvt/uploads/Main/software_engineering_21.pdf
- GQM approach
  - http://www.goldpractices.com/practices/gqm/index.php
- More
  - http://www.laynetworks.com/Software%20Engineering.htm
  - http://www.parlezuml.com/metrics/OO%20Design%20Principles%20&%20Metrics.pdf
  - http://www.parlezuml.com/metrics/Metrics%20Definitions.pdf

References II

- Lines of code
  - http://en.wikipedia.org/wiki/Source_lines_of_code
- McCabe's cyclomatic complexity
  - http://en.wikipedia.org/wiki/Cyclomatic_complexity
  - http://www.stsc.hill.af.mil/crosstalk/1994/12/xt94d12b.asp
  - http://www.answers.com/topic/cyclomatic-complexity
- Function points
  - http://www.softwaremetrics.com/fpafund.html
  - http://www.softwaremetrics.com/Function%20Point%20Training%20Booklet%20New.pdf
  - http://www.codeproject.com/KB/architecture/Softwarecosting.aspx

Pause for 10 min

- Form teams
- Get exercise sheet
- Solve exercises
- Hand in your solutions

Ex. 1 – Code Metrics (15 min)

- Calculate LOC
- Draw a flow graph
- Calculate McCabe's cyclomatic complexity

Code snippet

Ex. 2 – Function Points (15 min)

- Calculate function points for use-case
- Manager can change item price, amount and name
  - Manager selects the item
  - System displays item with editable fields
  - Manager inputs new values
  - Manager clicks on the save button
  - System validates new values and saves changes to db
- Alternative scenarios:
  - Manager clicks on the cancel button → nothing is saved
  - Values don't pass validation → system asks to fix input

Ex. 3 – GQM (15 min)

- Draw a Goal – Question – Metric graph with 2-3 questions and 5-7 metrics
- Goal: Improve points per lab of my team by 1 point

Ex. 4 – Metrics (10 min)

- Make up 5 metrics for evaluation of this workshop (lecture, speaker, exercises, ...)
- For each metric clearly state attribute being represented
- Example:
  - Number of slides – size of the presentation
  - Ratio of pictures to the number of slides – presentation comprehensibility

Ex. 5 – Metrics Again (15 min)

- Devise formulas/algorithms for calculating values of the following metrics:
  - Developer's contribution size
  - Developer's efficiency
  - Source code stability

Ex. 6 – People Productivity (15 min)

- Last month developers in
  - US offices have written 450K lines of code
  - India offices have written 1M lines of code
- While interpreting I have to take into account
  - People factor: size and expertise of the development team
  - ...
- Name 3 factors more that you have to take into account while interpreting these numbers

Ex. 6 – People Productivity

- Factors influencing productivity:
  - People factors: size and expertise of the development organization
  - Problem factors: complexity and number of changes in design constraint and requirement
  - Process factors: analysis and design techniques, languages and CASE tools
  - Product factors: reliability and performance of computer based systems
  - Resource factors: availability of hardware and software resources

Bonus Ex. – Bugs (5 min)

- How do you think where is the highest probability of undetected bug? Why?
  - In components showing LARGE number of known bugs
  - In components showing SMALL number of known bugs

Bonus Ex. – Bugs

"as the number of detected errors in a piece of software increases the probability of the existing of more undetected errors also increases"
Glenford Myers, 1976

Bugs tend to cluster → as you find a bug you should stop and write more tests for components where you have found a bug

Thank you for your time and attention!