Software Measurements and -Metrics

Prof. Dr.-Ing. Steffen Becker

Some slides are based on the lecture material provided by Ian Sommerville at [http://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Presentations/PPTX/Ch24.pptx]

Contents

- Definition: Metric
- Types of Metrics
- Simple SW-Metric
- OOP-Metrics
- Test-Metric

Metric: Definition

Pingo-Time

- What is a metric?
- Pingo: http://pingo.upb.de/100663

What is a measure?

- A measure is a **procedure** to assign a **number** to an **attribute of a physical object**
- **I.e., Measure: Object → Value**
- Examples:
  - Height → 5 meters
  - Temperature → 20 degrees Celsius

Relational systems

- A set of objects and relations between these is called a relational system
- Examples
  - Set of people, Relation isMale
  - Temperature, Relation isWarmerThan
  - Height, Relation isTallerThan
  - Age, Relation isTwiceAsOld
  - ...

Homomorphism

- In other words, a measure relates an empirical relational system to a formal relational system
  - Example: Age of a Person related to natural number of the years spent on earth
  - Example: Temperature related to a rational number dividing different melt points
  - ...
- We want a relationship so...
  - ... That statements on relationships in the empirical world can be translated to statements on relationships in the formal world
  - E.g.: Anton is twice as old as Egon ←→ Age(Egon) * 2 = Age(Anton)
  - → A homomorphism

Different scales

- A tuple of
  - an empirical relational system
  - an formal relational system
  - and a homomorphism between them is a scale
- Different scale types
  - Nominal (equal, not equal)
  - Ordinal (equal, not equal, <, >)
  - Interval (equal, not equal, <, >, +, -)
  - Ratio (equal, not equal, <, >, +, -, *, /)

Types of software measurements

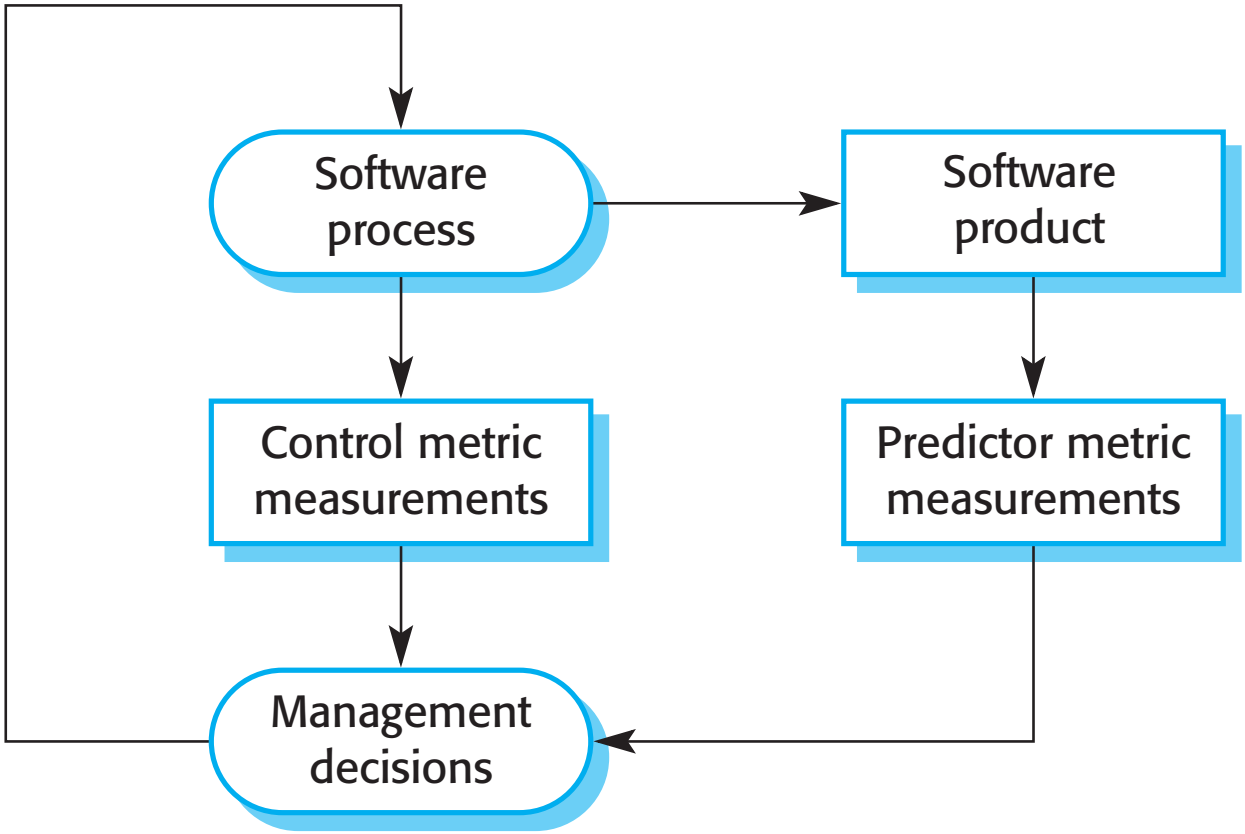
Software measurement and metrics

- Software measurement: deriving a numeric value for an attribute of a software product or process.
- Aim: comparisons between techniques and processes.
- Although some companies have introduced measurement programmes, most organisations still don't make systematic use of software measurement.
- There are few established standards in this area.

Software metric

- Any type of measurement which relates to a software system, process or related documentation
  - Lines of code in a program, the Fog index, number of person-days required to develop a component.
- Allow the software and the software process to be quantified.
- May be used to predict product attributes or to control the software process.
- Product metrics can be used for general predictions or to identify anomalous components.

Predictor and control measurements



Use of measurements

- To assign a value to system quality attributes
  - By measuring the characteristics of system components, such as their cyclomatic complexity, and then aggregating these measurements, you can assess system quality attributes, such as maintainability.
- To identify the system components whose quality is sub-standard
  - Measurements can identify individual components with characteristics that deviate from the norm. For example, you can measure components to discover those with the highest complexity. These are most likely to contain bugs because the complexity makes them harder to understand.

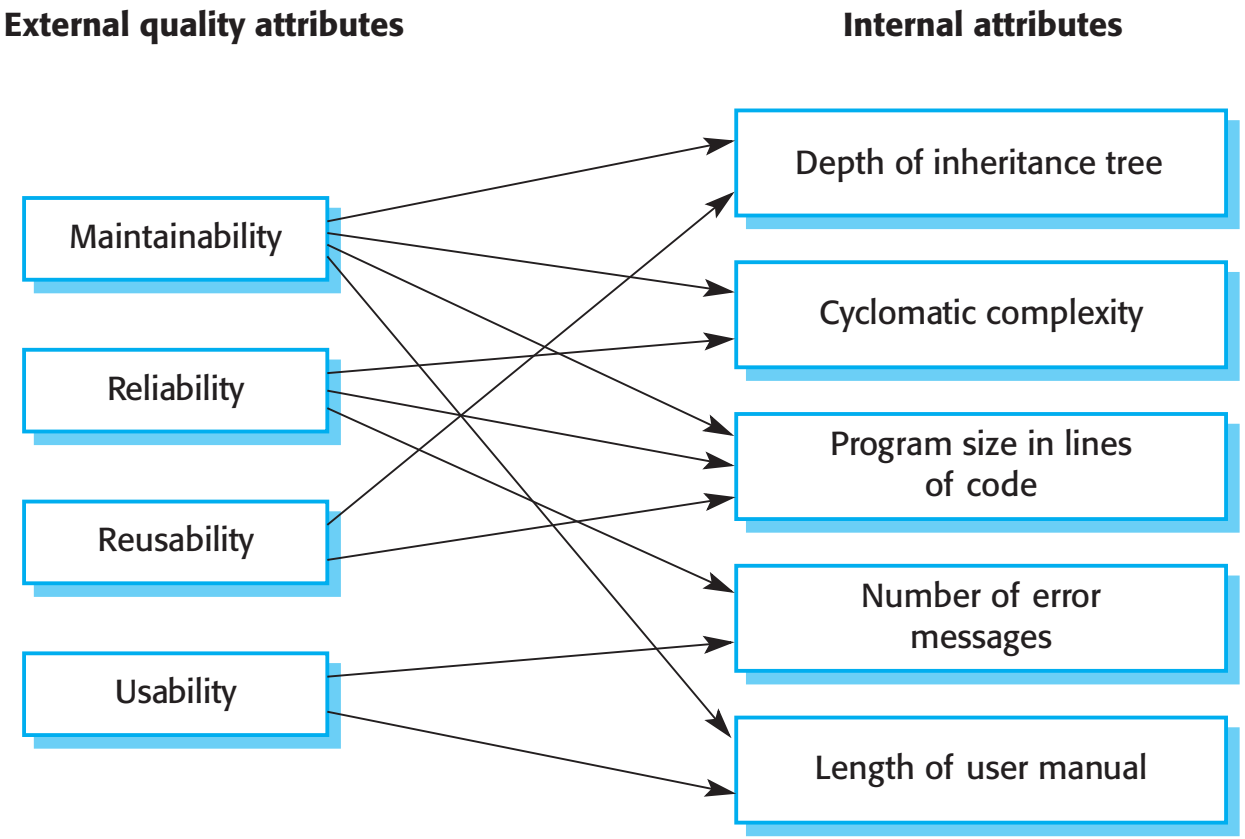
Metrics assumptions

- Software property can be measured.
- Relationship exists between what we can measure and what we want to know. We can only measure internal attributes but are often more interested in external software attributes.
- This relationship has been formalised and validated.
- It may be difficult to relate what can be measured to desirable external quality attributes.

Pingo-Time

- Flashback: What are external quality attributes?
- Pingo: http://pingo.upb.de/100663

Relationships between internal and external software



Problems with measurement in industry

- Impossible to quantify RoI of introducing an organizational metrics program.
- No standards for software metrics or standardized processes for measurement and analysis.
- Often software processes not standardized and poorly defined and controlled.
- Most work on software measurement has focused on code-based metrics and plan-driven development processes. However, more and more software is now developed by configuring ERP systems or COTS.
- Introducing measurement adds additional overhead to processes.

Product metrics

- A quality metric should be a predictor of product quality.
- Classes of product metrics
  - Dynamic metrics which are collected by measurements made of a program in execution;
  - Static metrics which are collected by measurements made of the system representations;
  - Dynamic metrics help assess efficiency and reliability
  - Static metrics help assess complexity, understandability and maintainability.

Dynamic and static metrics

- Dynamic metrics are closely related to software quality attributes
  - It is relatively easy to measure the response time of a system (performance attribute) or the number of failures (reliability attribute).
- Static metrics have an indirect relationship with quality attributes
  - You need to try and derive a relationship between these metrics and properties such as complexity, understandability and maintainability.

Simple SW metrics

Static software product metrics

Software metric	Description
Fan-in/Fan-out	Fan-in is a measure of the number of functions or methods that call another function or method (say X). Fan-out is the number of functions that are called by function X. A high value for fan-in means that X is tightly coupled to the rest of the design and changes to X will have extensive knock-on effects. A high value for fan-out suggests that the overall complexity of X may be high because of the complexity of the control logic needed to coordinate the called components.
Length of code	This is a measure of the size of a program. Generally, the larger the size of the code of a component, the more complex and error-prone that component is likely to be. Length of code has been shown to be one of the most reliable metrics for predicting error-proneness in components.

Static software product metrics

Software metric	Description
Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability. I discuss cyclomatic complexity in Chapter 8.
Length of identifiers	This is a measure of the average length of identifiers (names for variables, classes, methods, etc.) in a program. The longer the identifiers, the more likely they are to be meaningful and hence the more understandable the program.
Depth of conditional nesting	This is a measure of the depth of nesting of if-statements in a program. Deeply nested if-statements are hard to understand and potentially error-prone.
Fog index	This is a measure of the average length of words and sentences in documents. The higher the value of a document's Fog index, the more difficult the document is to understand.

Counting Cyclomatic Complexity

Cyclomatic Complexity M:

- Each method starts at 1
- +1 for every if, while, do, for, ?:, catch, case.
- +1 for each occurence of the operators && or ||

Alternative: M = e – n + 2p

- e: Edges in control flow graph
- n: Nodes in control flow graph
- p: Number of nodes without outgoing edges

Cyclomatic Complexity Example

[https://www.holisticon.de/2011/05/softwaremetriken-zyklomatische-komplexitat/]

```
/**
 * Print out a list of messages.
 * @param tokens
 */
// complexity++
public final void iterateComplex(final List<String> tokens) {
    if (tokens == null) {                                  // complexity++
        return;
    }
    for (final String eachToken : tokens) {                // complexity++
        if (eachToken != null &&          // complexity++ complexity++
                ! "".equals(eachToken.trim())) {
            System.out.println(eachToken);
        }
    }
    return;
}
```

Cyclomatic Complexity Refactored

[https://www.holisticon.de/2011/05/softwaremetriken-zyklomatische-komplexitat/]

```
/**
 * Null-check for List of messages.
 * @param messages
 */
// complexity++
public final void callIterateSimple(final List<String> messages){
    if (messages != null){                              // complexity++
        iterateSimple(messages);
    }
}

/**
 * Print out a list of messages.
 * @param messages
 */
// complexity++
public final void iterateSimple(final List<String> messages) {
    for (final String eachMessage : messages) {         // complexity++
        if (! StringUtils.isBlank(eachMessage)) {       // complexity++
            System.out.println(eachMessage);
        }
    }
    return;
}
```

OOP Metrics

The CK object-oriented metrics suite

Object-oriented metric	Description
Weighted methods per class (WMC)	This is the number of methods in each class, weighted by the complexity of each method. Therefore, a simple method may have a complexity of 1, and a large and complex method a much higher value. The larger the value for this metric, the more complex the object class. Complex objects are more likely to be difficult to understand. They may not be logically cohesive, so cannot be reused effectively as superclasses in an inheritance tree.
Depth of inheritance tree (DIT)	This represents the number of discrete levels in the inheritance tree where subclasses inherit attributes and operations (methods) from superclasses. The deeper the inheritance tree, the more complex the design. Many object classes may have to be understood to understand the object classes at the leaves of the tree.
Number of children (NOC)	This is a measure of the number of immediate subclasses in a class. It measures the breadth of a class hierarchy, whereas DIT measures its depth. A high value for NOC may indicate greater reuse. It may mean that more effort should be made in validating base classes because of the number of subclasses that depend on them.

The CK object-oriented metrics suite

Object-oriented metric	Description
Coupling between object classes (CBO)	Classes are coupled when methods in one class use methods or instance variables defined in a different class. CBO is a measure of how much coupling exists. A high value for CBO means that classes are highly dependent, and therefore it is more likely that changing one class will affect other classes in the program.
Response for a class (RFC)	RFC is a measure of the number of methods that could potentially be executed in response to a message received by an object of that class. Again, RFC is related to complexity. The higher the value for RFC, the more complex a class and hence the more likely it is that it will include errors.
Lack of cohesion in methods (LCOM)	LCOM is calculated by considering pairs of methods in a class. LCOM is the difference between the number of method pairs without shared attributes and the number of method pairs with shared attributes. The value of this metric has been widely debated and it exists in several variations. It is not clear if it really adds any additional, useful information over and above that provided by other metrics.

Software component analysis

- System component can be analyzed separately using a range of metrics.
- The values of these metrics may then compared for different components and, perhaps, with historical measurement data collected on previous projects.
- Anomalous measurements, which deviate significantly from the norm, may imply that there are problems with the quality of these components.

Test metrics for white box tests

Glass-Box-Testen

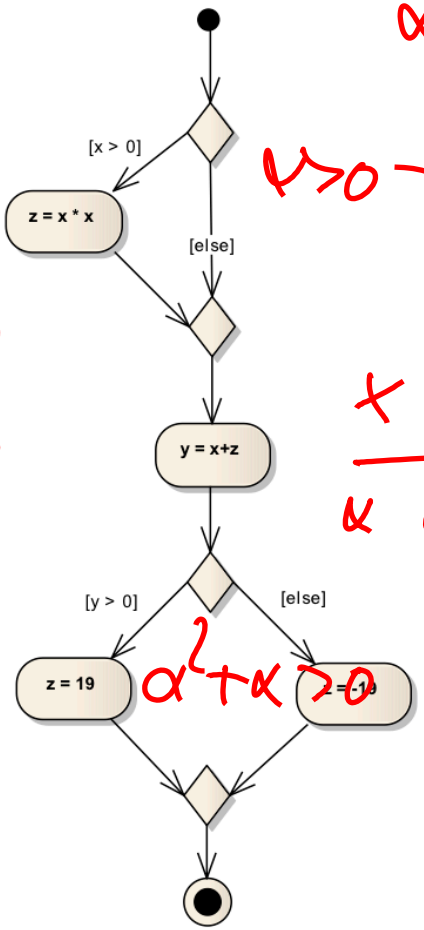
For whitebox tests, we produce test data based on the program source code.

The data is constructed such that a defined path through the program is executed

Coverage criteria

- **Statement coverage** (C_0)
  Each statement is executed once
- **Branch coverage** (C_1)
  Each condition once fulfilled and once not
- **Path coverage** (C_∞)
  each path covered at least once
  (often not possible in practice)

White-Box-Test



Approach

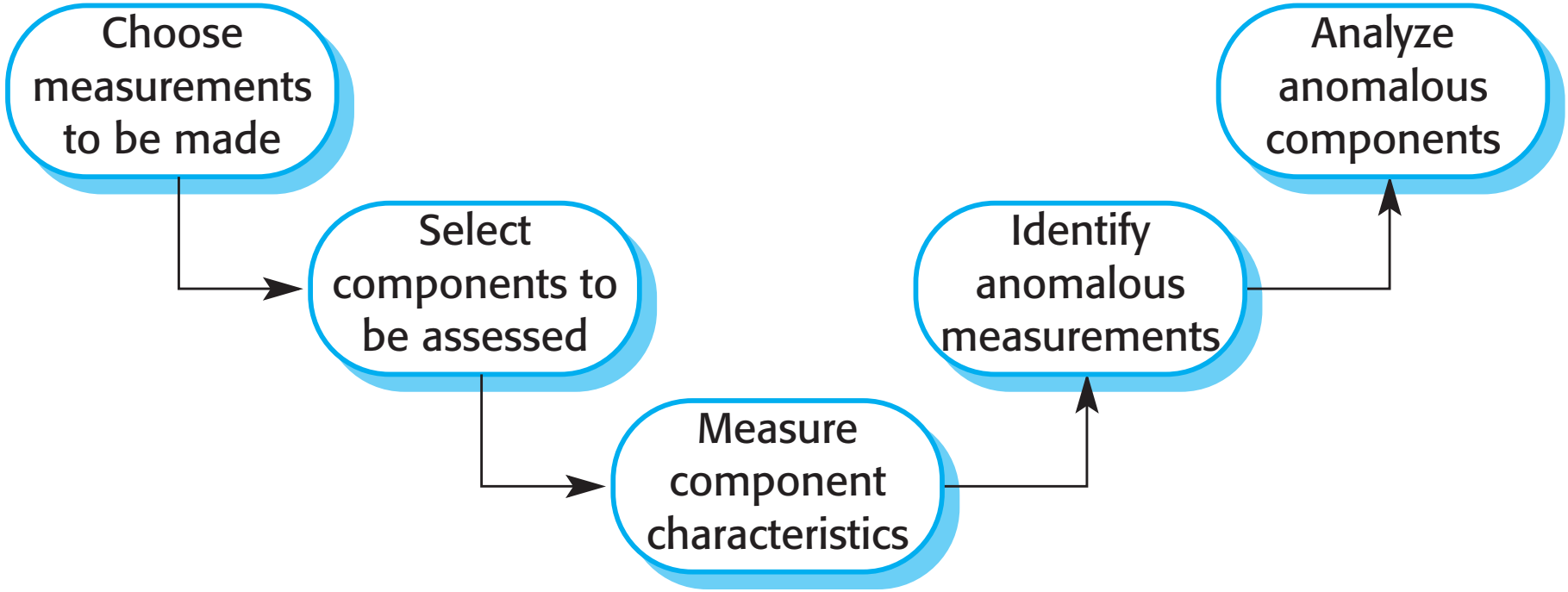
Approach

- **Deterime branches**,
  which result in the required coverage
- **Determine Conditions** for each path
- **Determine data** which fulfil the conditions

Profiling tools measure the covered pathes, e.g., JaCoCo.

Conclusions

The process of product measurement



Measurement surprises

- Reducing the number of faults in a program leads to an increased number of help desk calls
  - The program is now thought of as more reliable and so has a wider more diverse market. The percentage of users who call the help desk may have decreased but the total may increase;
  - A more reliable system is used in a different way from a system where users work around the faults. This leads to more help desk calls.

Key points

- Reviews of the software process deliverables involve a team of people who check that quality standards are being followed.
- In a program inspection or peer review, a small team systematically checks the code. They read the code in detail and look for possible errors and omissions
- Software measurement can be used to gather data about software and software processes.
- Product quality metrics are particularly useful for highlighting anomalous components that may have quality problems.