

IN5140

Smart processes and agile methods in Software Engineering

Group session

Agenda

First hour:

- Quick progress overview
- Exercise 1

Second hour:

Exam questions

Why measurement ?

Measurements are central in all kinds of improvement work – including software process improvement, both plan-driven and agile processes

Knowledge about measurements is useful whatever the discipline

- An enormous amount of data in our digital era. More important than ever to know
 - how data has been produced and
 - its quality
- Can you trust the data, how reliable is it?
 - fake news
 - conspiracy theories
 - basis for political decisions
 - basis for research

Measurements: Direct & Indirect

Direct

- Direct measurements of an attribute of an entity involves no other attribute or entity.

Eg. software length in terms of lines of code.

Indirect

- Indirect measurement is useful in making visible the interaction between direct measurements.

Eg. productivity, software quality

Measurements: Objective & Subjective

Objective

- Based on facts, not feelings, prejudice, opinions etc.

Subjective

- Based on feelings, not facts
- Related to the way people experience things

Measurements: Quantitative & Qualitative

Quantitative

- Data expresses quantity
- Data expressed as numbers
- Used in statistics

Temperature gauge measures 17°
Celsius

Qualitative

- Data expresses quality in some sense
- Data expressed as text, images and forms except numbers
- Not used in statistics

“I feel like it is 17° Celsius”

Exercise 1

Select three variables (measures) that you may use in your project to measure aspects of the process, system or outcome (success). For each of them, describe whether it:

- Can be measured directly (or only indirectly)
- Is objective or subjective
- Is quantitative or qualitative
- Is measured at the nominal, ordinal, interval or ratio scale

Measurements Scale

Scale Type	Characterization	Examples (generic)	Examples (Softw. Eng.)
Nominal	Divides the set of objects into categories, with no particular ordering among them	Labeling, classification	Name of process model Defect type
Ordinal	Divides the set of entities into categories that are ordered	Preference, ranking, difficulty, Likert scales	Failure severity Complexity of software
Interval	Comparing the differences between values is meaningful	Calendar time, temperature (Fahrenheit, Reaumur, Celsius)	Beginning and end date of activities
Ratio	There is a meaningful “zero” value, and ratios between values are meaningful	Length, weight, time intervals, absolute temperature (Kelvin)	Lines of code Lead time Number of errors Cost per function

Exercise 2

1. What does complexity mean?
2. How can we measure software complexity?
3. To what extent do you consider the measure Cyclomatic Complexity to represent the concept of complexity?
4. How could we find out whether Cyclomatic Complexity actually represents complexity in practice?
5. Would you consider software size as part (a subconcept) of maintainability?
6. Is size a part of complexity?
7. How can we measure software size?

Exercise 2

1. **What does complexity mean?**
2. How can we measure software complexity?
3. To what extent do you consider the measure Cyclomatic Complexity to represent the concept of complexity?
4. How could we find out whether Cyclomatic Complexity actually represents complexity in practice?
5. Would you consider software size as part (a subconcept) of maintainability?
6. Is size a part of complexity?
7. How can we measure software size?

1. What does complexity mean?

- The state or quality of being intricate or complicated
- Difficult to grasp and comprehend
 - Does not promote understanding
- Several factors combined contributes to high complexity

Exercise 2

1. What does complexity mean?
- 2. How can we measure software complexity?**
3. To what extent do you consider the measure Cyclomatic Complexity to represent the concept of complexity?
4. How could we find out whether Cyclomatic Complexity actually represents complexity in practice?
5. Would you consider software size as part (a subconcept) of maintainability?
6. Is size a part of complexity?
7. How can we measure software size?

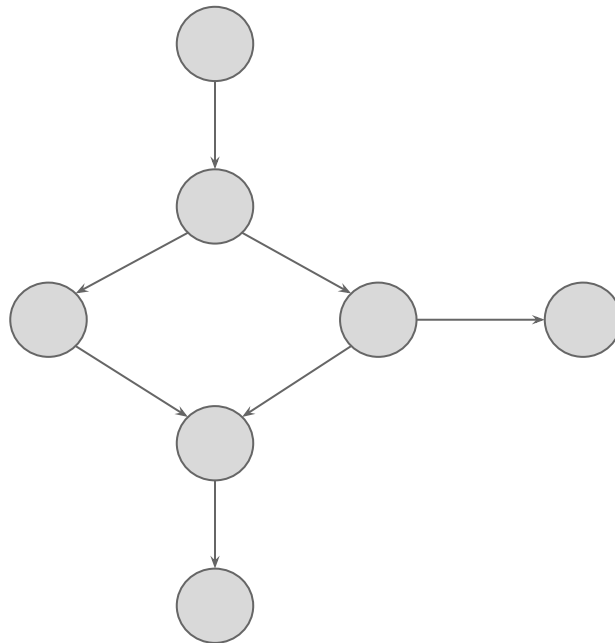
2. Can we measure software complexity?

- Cyclomatic complexity is often referred to as software complexity
- We can measure cyclomatic complexity with a formula:
 - **$M = E - N + 2P$**
 - M = cyclomatic complexity
 - E = number of edges
 - N = number of nodes
 - P = number of connected components
 - <https://docs.sonarqube.org/latest/user-guide/metric-definitions/>

2. Can we measure software complexity?

- Using this formula: $M = E - N + 2P$, what is the Cyclomatic Complexity of the following graph:

- M = cyclomatic complexity
- E = number of edges
- N = number of nodes
- P = number of connected components

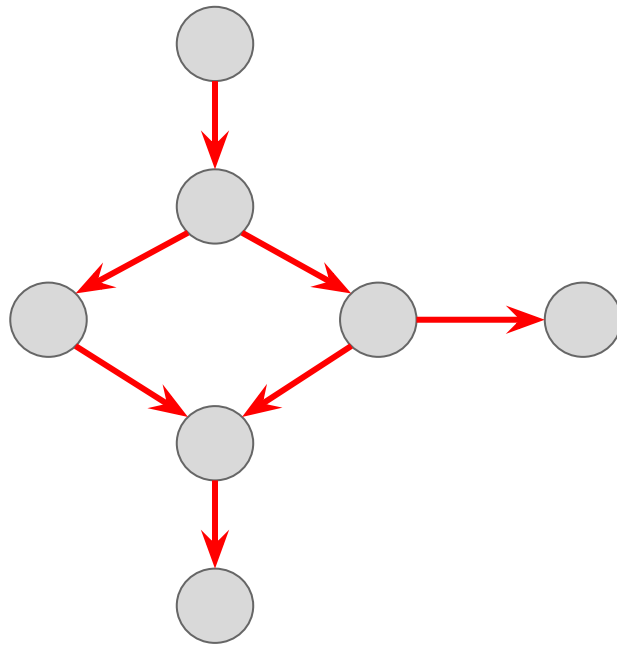


$$M = E - N + 2P$$

2. Can we measure software complexity?

- Using this formula: $M = E - N + 2P$, what is the Cyclomatic Complexity of the following graph:

- M = cyclomatic complexity
- E = number of edges
- N = number of nodes
- P = number of connected components

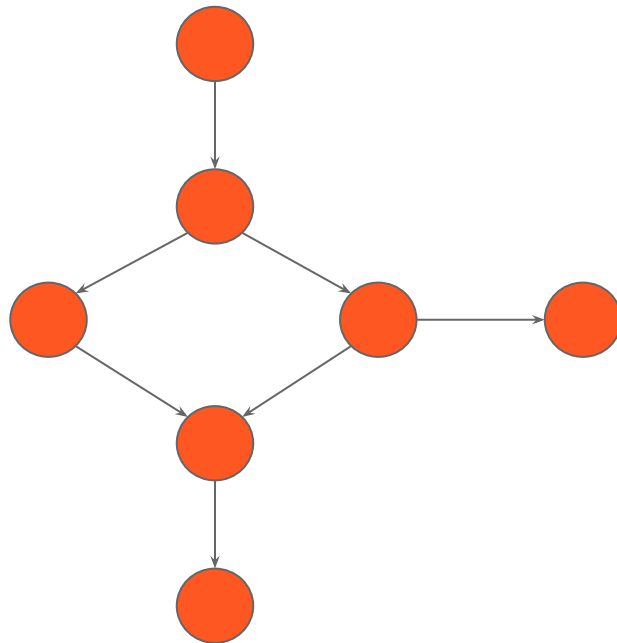


$$M = 7 - N + 2P$$

2. Can we measure software complexity?

- Using this formula: $M = E - N + 2P$, what is the Cyclomatic Complexity of the following graph:

- M = cyclomatic complexity
- E = number of edges
- N = number of nodes
- P = number of connected components

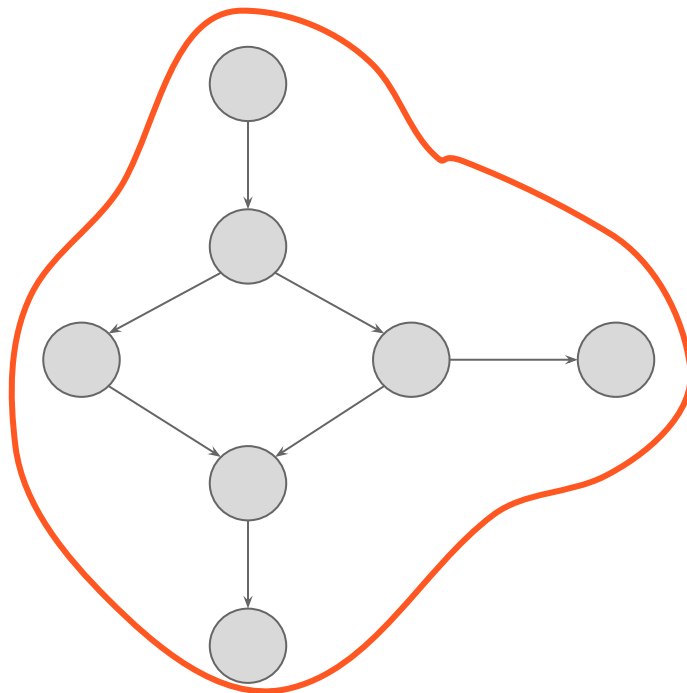


$$M = 7 - 7 + 2P$$

2. Can we measure software complexity?

- Using this formula: $M = E - N + 2P$, what is the Cyclomatic Complexity of the following graph:

- M = cyclomatic complexity
- E = number of edges
- N = number of nodes
- P = number of connected components

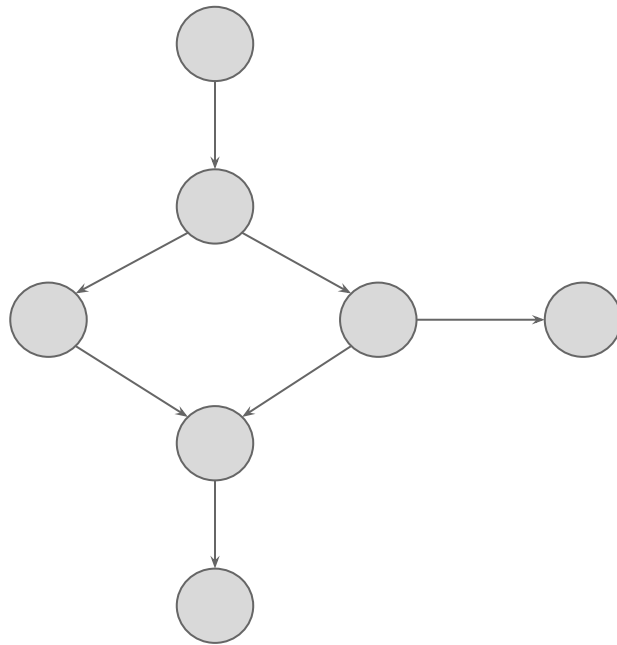


$$M = 8 - 7 + 2 \cdot 1$$

2. Can we measure software complexity?

- Using this formula: $M = E - N + 2P$, what is the Cyclomatic Complexity of the following graph:

- M = cyclomatic complexity
- E = number of edges
- N = number of nodes
- P = number of connected components



M = 2

Exercise 2

1. What does complexity mean?
2. How can we measure software complexity?
- 3. To what extent do you consider the measure Cyclomatic Complexity to represent the concept of complexity?**
4. How could we find out whether Cyclomatic Complexity actually represents complexity in practice?
5. Would you consider software size as part (a subconcept) of maintainability?
6. Is size a part of complexity?
7. How can we measure software size?

3. To what extent do you consider the measure cyclomatic complexity to represent the concept of complexity?

- Cyclomatic Complexity alone does not cover all aspects of complexity, although it proves proof that complexity exists
- Indicates software complexity (quantitative measure)
- Cyclomatic Complexity does not reveal underlying factors

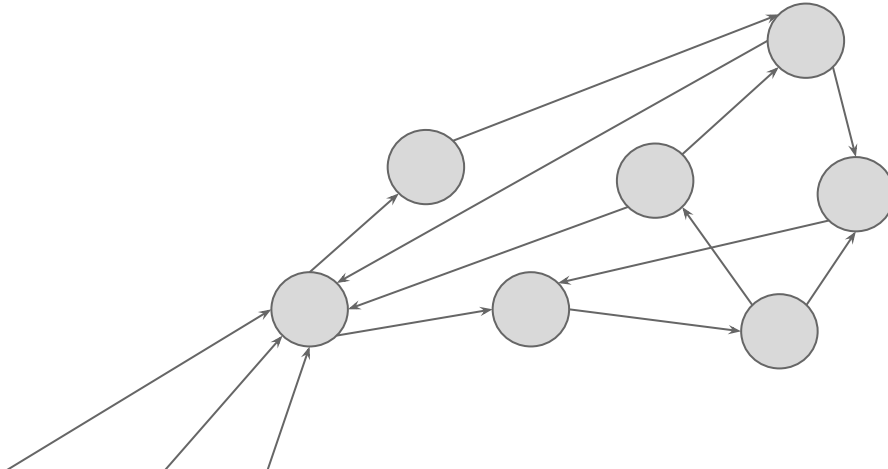
Exercise 2

1. What does complexity mean?
2. How can we measure software complexity?
3. To what extent do you consider the measure Cyclomatic Complexity to represent the concept of complexity?
4. **How could we find out whether Cyclomatic Complexity actually represents complexity in practice?**
5. Would you consider software size as part (a subconcept) of maintainability?
6. Is size a part of complexity?
7. How can we measure software size?

4. How could we find out whether cyclomatic complexity actually represent complexity in practice?

4. How could we find out whether cyclomatic complexity actually represent complexity in practice?

- $M = E - N + 2P$ ← a number
- How many connecting modules does the software have?



Exercise 2

1. What does complexity mean?
2. How can we measure software complexity?
3. To what extent do you consider the measure Cyclomatic Complexity to represent the concept of complexity?
4. How could we find out whether Cyclomatic Complexity actually represents complexity in practice?
- 5. Would you consider software size as part (a subconcept) of maintainability?**
6. Is size a part of complexity?
7. How can we measure software size?

5. Would you consider software size as a concept (a sub concept) of maintainability?

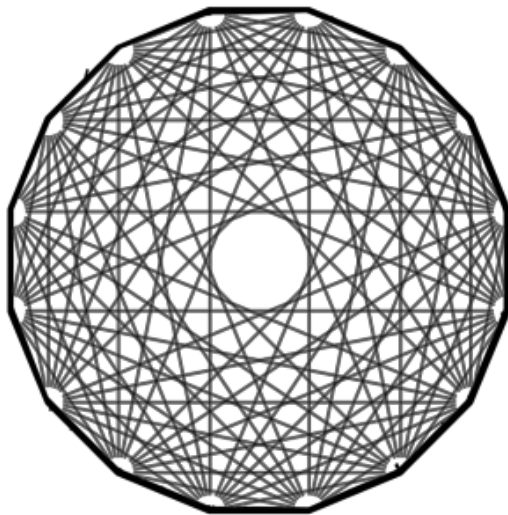
- Yes, greater software size often entails more maintenance
- Software size alone does not cover all levels of maintainability
 - Complexity
 - Cohesion and coupling
 - Technical debt
 - Bugs and errors
 - Time
 - Cost
 - Employee expertise and competence
 - etc.

Exercise 2

1. What does complexity mean?
2. How can we measure software complexity?
3. To what extent do you consider the measure Cyclomatic Complexity to represent the concept of complexity?
4. How could we find out whether Cyclomatic Complexity actually represents complexity in practice?
5. Would you consider software size as part (a subconcept) of maintainability?
- 6. Is size a part of complexity?**
7. How can we measure software size?

6. Is size a part of complexity?

- Often, but not always
- Greater size often entails higher complexity
- Number of connected components/dependencies in the software also matters



Exercise 2

1. What does complexity mean?
2. How can we measure software complexity?
3. To what extent do you consider the measure Cyclomatic Complexity to represent the concept of complexity?
4. How could we find out whether Cyclomatic Complexity actually represents complexity in practice?
5. Would you consider software size as part (a subconcept) of maintainability?
6. Is size a part of complexity?
- 7. How can we measure software size?**

7. How do we measure software size?

- Lines Of Code (LOC) for all components in the software
- Number of bytes for entire project
- Number of components in the software

15 minute  break



Next Week:

**Lean and agile
software engineering**