Software Product Design and Development II
Metrics & Visualization

John Businge

John.businge@unlv.edu

“You can’t manage what you can’t measure.”
– Unknown author*

Introduction

- Most metrics tools include some visualization to show a large volume of data.
- Data is tedious to interpret, and visualizations help you to analyze different measurements simultaneously.
- When different metrics are combined properly one can deduce which artifacts represent potential issues

Source Code Metrics

- Some common metrics are associated with source code and/or object-oriented.

LOC	**Lines of Code**
NOA	Number of Attributes
NOM	Number Of Methods
FAN-IN	Number of classes referenced by this class
FAN-OUT	Number of classes referenced by this class

CK Metrics

- Proposed by Chidamber& Kemerer(1994)
- Used a lot in academia (although CBO/LCOM are criticized as being outdated)

WMC	Weight Methods per Class
DIT	Depth of Inheritance Tree
NOC	Number of Children
CBO	Coupling Between Object classes
RFC	Response For a Class
LCOM	Lack of Cohesion of Methods

Other Metrics

- MOOD – Six metrics (Method Hiding Factor, Attribute Hiding Factor, Method Inheritance Factor, Attribute Inheritance Factor, Polymorphism Factor, Coupling Factor)
- QMOOD – 11 properties mapped to 6 quality attributes
- Community Metrics – Number of Developers, Authorship…

- … and many others.

Metrics are Relative

- A value considered low for one project might be very high for another.
- It really depends on the domain, size, and complexity of the application.
- Researchers usually take a median value on a project as a reference to what is high/low.

UML Diagrams

- (Mostly) The simple and standard way to present an abstract visualization of a system
- UML defines 13 diagrams
- Useful to plan and design reengineer activities

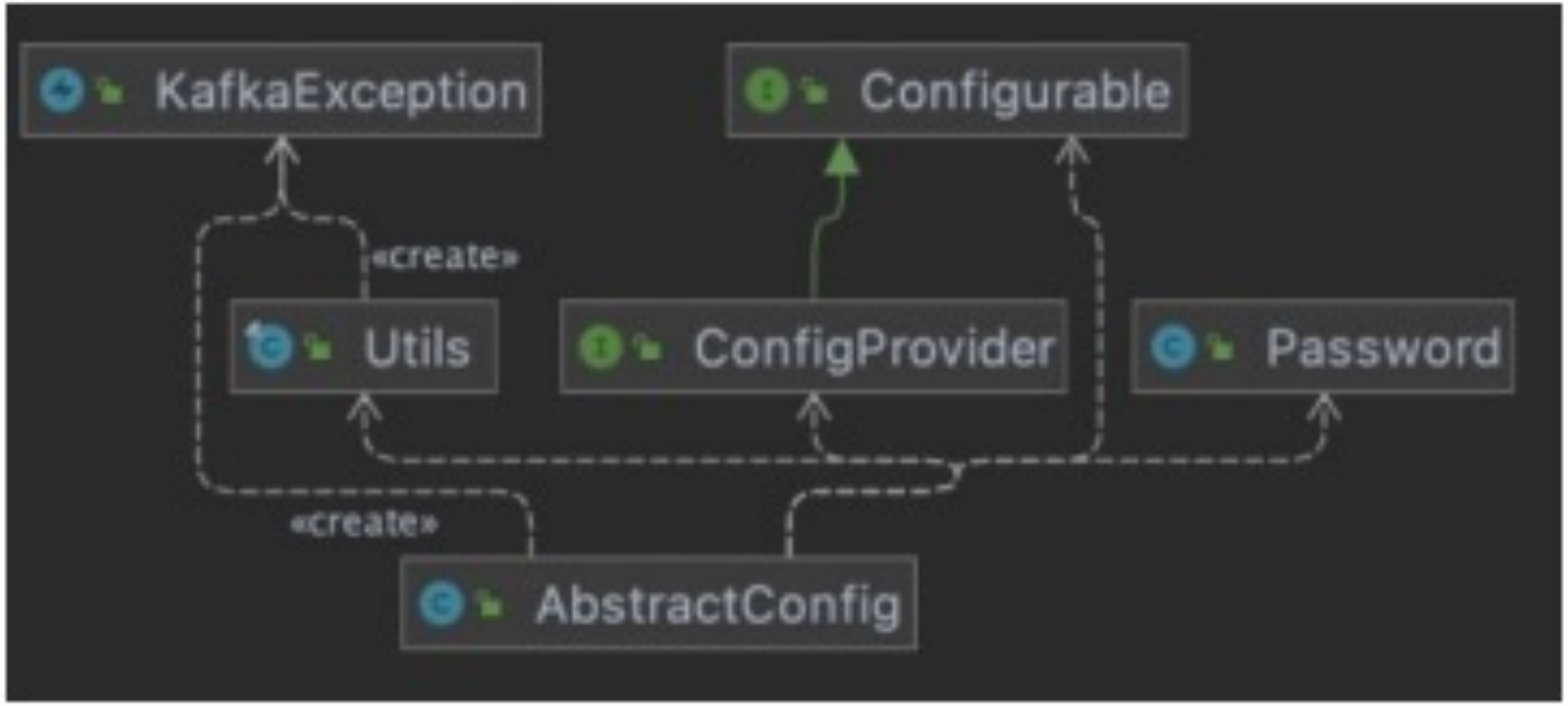
UML Diagrams: Attention

- In the FINAL Report, it is required for you to present a Class Diagram focusing on the reengineered parts
  - Both Before and After the reengineer activities
  - You may complement with other diagrams IF (and only if) you deem important to better explain/describe/reason your reengineer decisions/results.

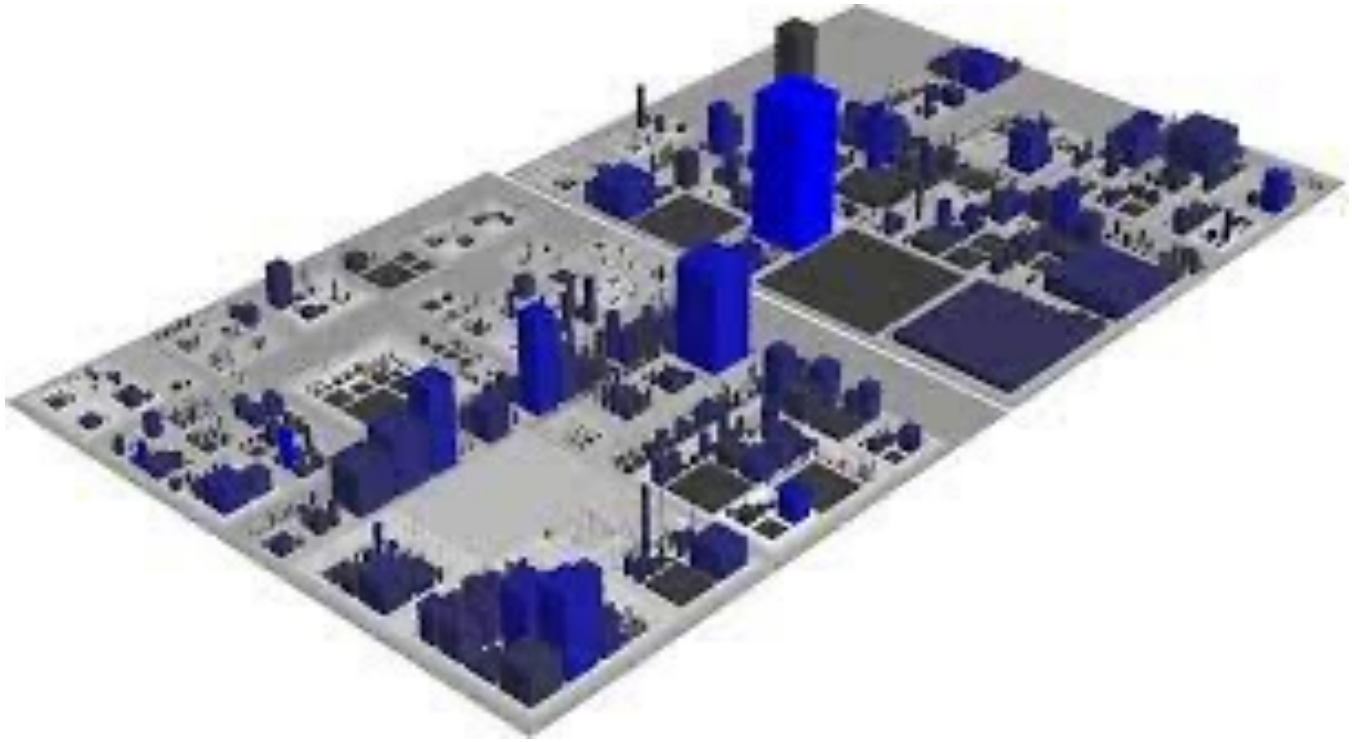
Sample Class Diagram – Patch in apache/kafka

https://github.com/apache/kafka/pull/10704

KAFKA-12791: ConcurrentModificationException in AbstractConfig use by KafkaProducer #10704



Code City



CodeCity is a visualization concept for source code.

The source code is shown as an interactive 3D city.

Code City

- Packages are “districts”, “neighborhoods,” or “city blocks”
- Each “building” represents a class \
- Width = Number of Attributes
- Height = Number of Methods
- Antennas => Classes with many methods and no attributes
- Parking lot => Classes with many attributes and no methods
- Skyscraper => Classes with a large number of methods and has many attributes

JSCity

- JSCity is a CodeCity implementation for JavaScript code
- Folders are districts, and files are sub-districts
- Functions are buildings; inner functions are represented as buildings on top of their nested function/building.
  - Width = Number of Variables (NOV)
  - Height = Lines of Code (LOC)
  - Blue Color = buildings are named functions
  - Green Color = buildings are anonymous functions.

(link on the course lesson page)

CodeScene

- CodeScene is a web application for software analytics and visualization.
- Based on the book “Your Code as a Crime Scene”.
- It fetches your project directly from GitHub.
- We are going to use the free version on the lecturers here (link on the course lesson page, or just type CodeScene in a search engine)

CodeScene: Project

- CodeScene is one of the official tools for this course.
- Because of its web interface, it is multiplatform.
- For the Intermediate Report, it is required to use CodeScene as a Visualization tool.
- Optionally, you can also use other visualization tools.
- For the Final Report, there is no requirement to use CodeScene, but it is encouraged.