A PRACTICAL APPROACH

Increasing Productivity in a highly dynamic Software World

April 3, 2023

Dr. Johannes Bohnet, CEO & Founder Seerene GmbH
Alexander Hofmann, CTO MaibornWolff GmbH

```
package com.
import ...
public final class LocationUtils {

    /**
     * Parses Point from it's String representation.
     * @param locationString - String that represents location, as 2 double values split with coma. Accepts space after/before co
     * @return org.springframework.data.solr.core.geo.Point instance
     */
    public static Point parseLocation(String locationString) {
        Preconditions.checkNotNull(locationString,  errorMessage: "Location String should not be null");
        Preconditions.checkArgument(locationString.contains(","),  errorMessage: "Location must be split with coma");
        locationString = locationString.trim();

        if (locationString.contains(" ,")) {
            locationString = locationString.replaceAll( regex: " ,",  replacement: ",");
        }

        if (locationString.contains(", ")) {
            locationString = locationString.replaceAll( regex: ", ",  replacement: ",");
        }

        String[] location = locationString.split( regex: ",");
        Preconditions.checkArgument( expression: location.length >= 2,  errorMessage: "Location should constist at least 2 Double parameters");
        double lat = Double.parseDouble(location[0]);
        double lon = Double.parseDouble(location[1]);

        return new Point(lat, lon);
    }
}
```

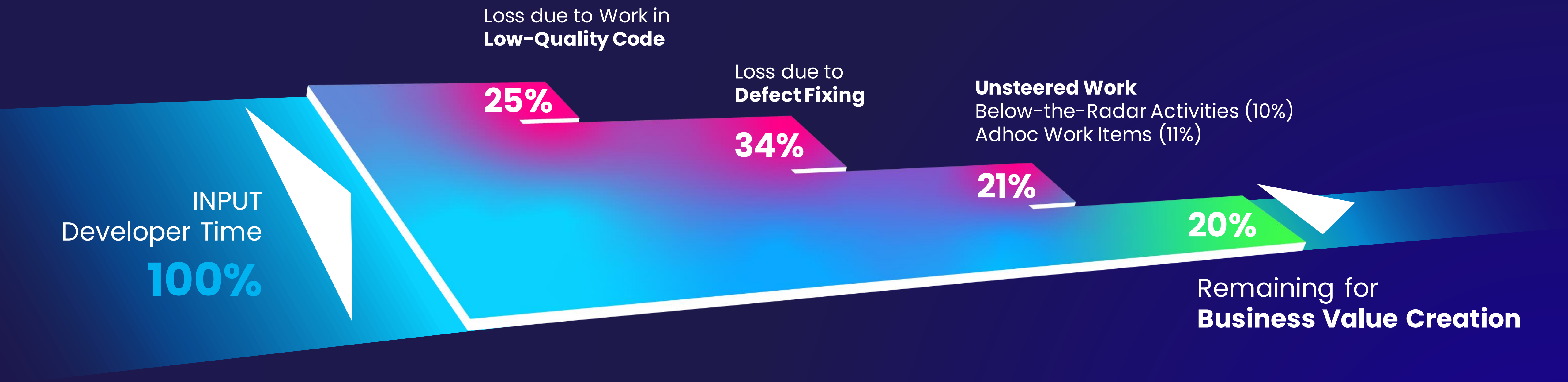
1 View inside the Software Factory

CAPACITY VERSUS NEEDS

The world wants more Software than the world can build

A LOT OF POTENTIAL FOR MORE PRODUCTIVITY

Why so little software for so much money?



2 3 Superpowers for more Productivity

BY 2030: INCREASE PRODUCTIVITY BY A FACTOR OF 5

Superpower 1: Near-/Offshore Talents

Demographic change

New talent markets (Africa)

Distributed work … new normal

Sourcing

Specialization

BY 2030: INCREASE PRODUCTIVITY BY A FACTOR OF 5

Superpower 2: Artificial Intelligence

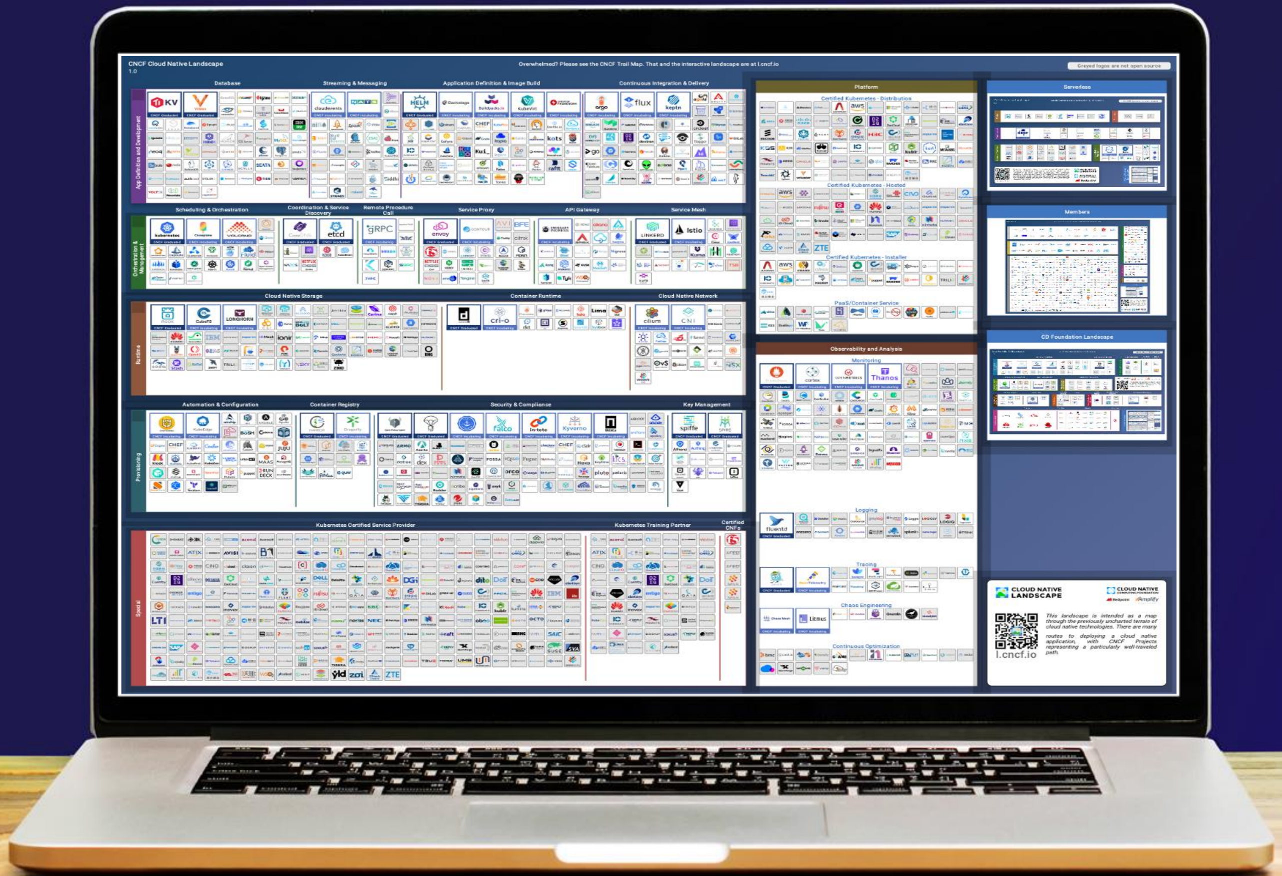
Software engineers today
factor 10 faster than in 1996

Similar productivity boost with open
source (start 1998) … **today 80:20**
from standard:individual

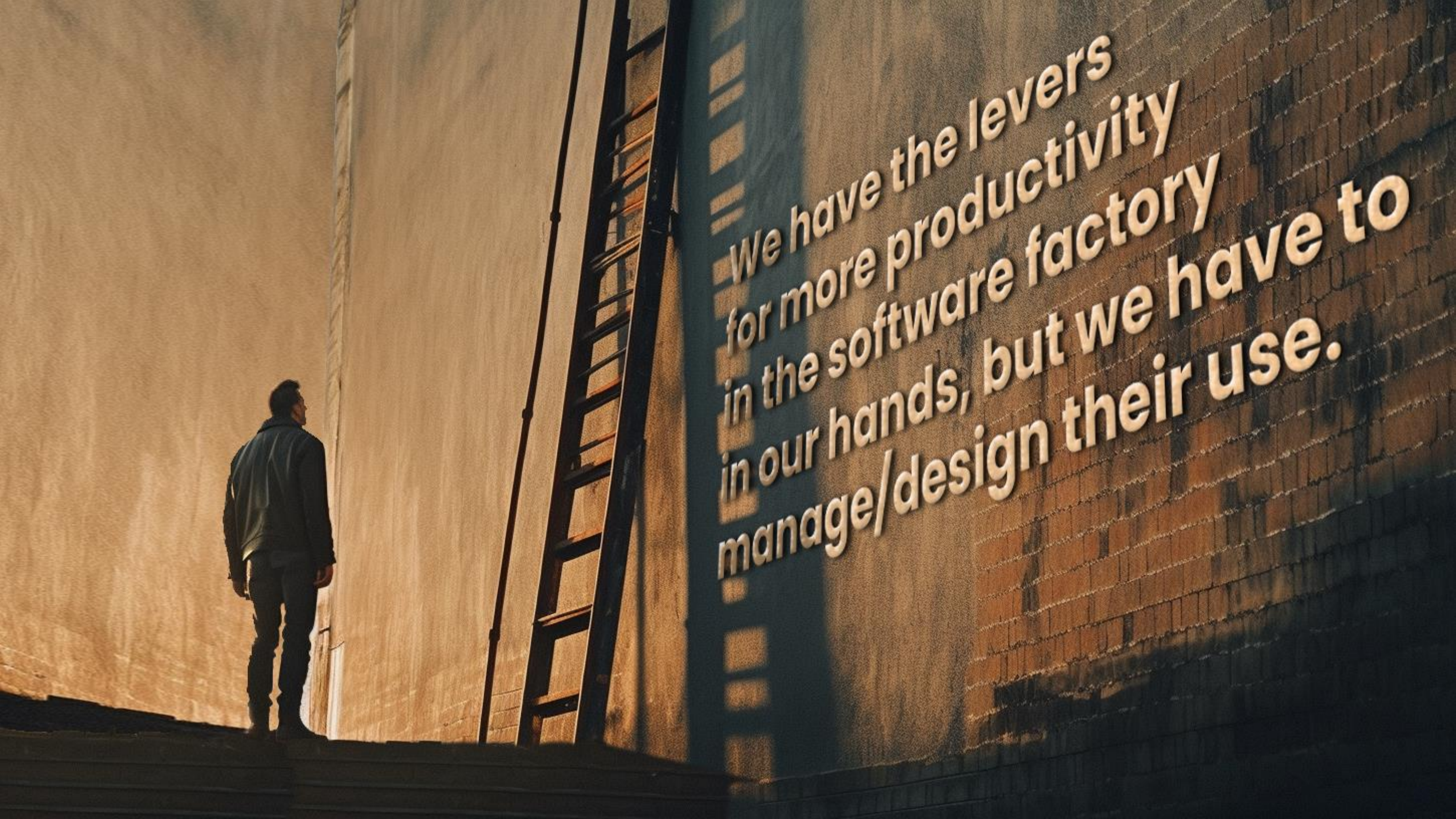
New kid in town … Large Language Models
(ChatGPT, Copilot X, etc.)
Revolution of Software development

BY 2030: INCREASE PRODUCTIVITY BY A FACTOR OF 5

Superpower 3: Technology



- CNCF map -> Open Source (Cloud Native Computing Foundation)
- Public cloud infrastructures
- Virtual Reality / Metaverse
- LowCode platforms
- Quantum Computing



We have the levers
for more productivity
in the software factory
in our hands, but we have to
manage/design their use.

WE URGENTLY
NEED A NEW
APPROACH

Next Level

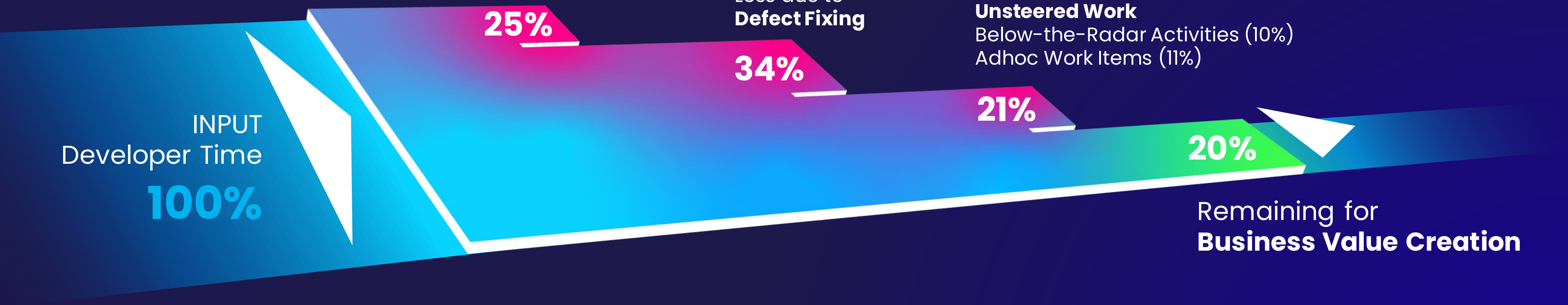
Software Engineering

```
package com.
import ...
public final class LocationUtils {

    /**
     * Parses Point from it's String representation.
     * @param locationString - String that represents location, as 2 double values split with coma. Accepts space after/before
     * @return org.springframework.data.solr.core.geo.Point instance
     */
    public static Point parseLocation(String locationString) {
        Preconditions.checkNotNull(locationString,  errorMessage: "Location String should not be null");
        Preconditions.checkArgument(locationString.contains(","),  errorMessage: "Location must be split with coma");
        locationString = locationString.trim();

        if (locationString.contains(" ,")) {
            locationString = locationString.replaceAll( regex: " ,",  replacement: ",");
        }

        if (locationString.contains(", ")) {
            locationString = locationString.replaceAll( regex: ", ",  replacement: ",");
        }

        String[] location = locationString.split( regex: ",");
        Preconditions.checkArgument( expression: location.length >= 2,  errorMessage: "Location should constist at least 2 Double parameters");
        double lat = Double.parseDouble(location[0]);
        double lon = Double.parseDouble(location[1]);

        return new Point(lat, lon);
    }
}
```

3 Next Level Software Engineering

MAKING USE OF DATA IN THE DEVELOPMENT INFRASTRUCTURE

Measuring Efficiency



INPUT
Developer Time
100%

Loss due to Work in
Low-Quality Code
25%

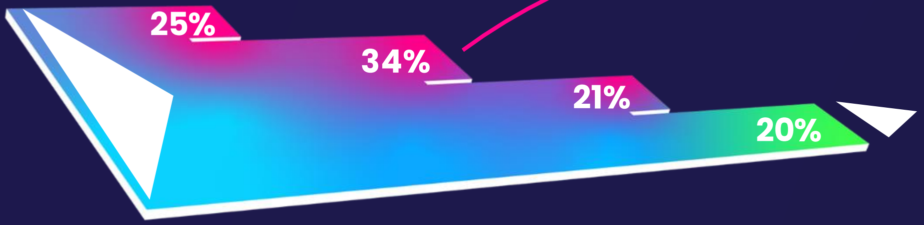
Loss due to
Defect Fixing
34%

Unsteered Work
Below-the-Radar Activities (10%)
Adhoc Work Items (11%)
21%

20%
Remaining for
Business Value Creation

INEFFICIENCY EATS UP BUDGETS

Finding and fixing Code that unnecessarily consumes Developer Time



Navigating to the
Root Cause of the Problem

Code Landscape visualized
as a “Code City”

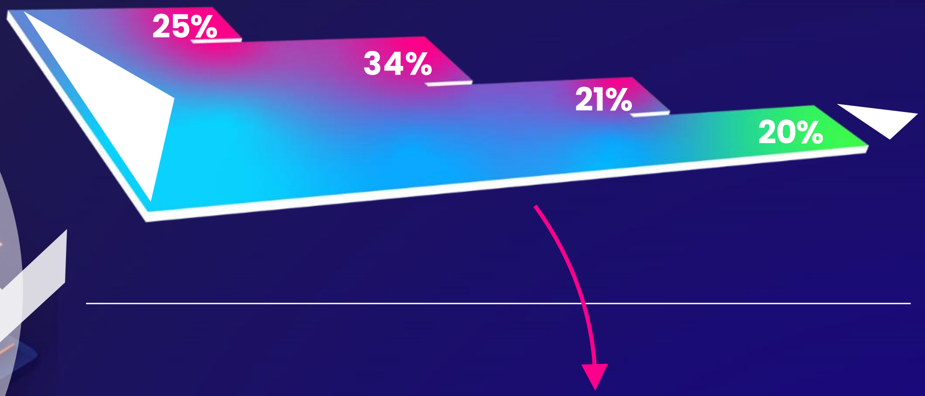
ANALYTICS-DRIVEN CONTINUOUS IMPROVEMENT CYCLE

Key is to close the loop and measure the impact of improvement activities

Strategic

Measure
Impact of
Improvements

Measure Efficiency



Operational

Execute
Improvements

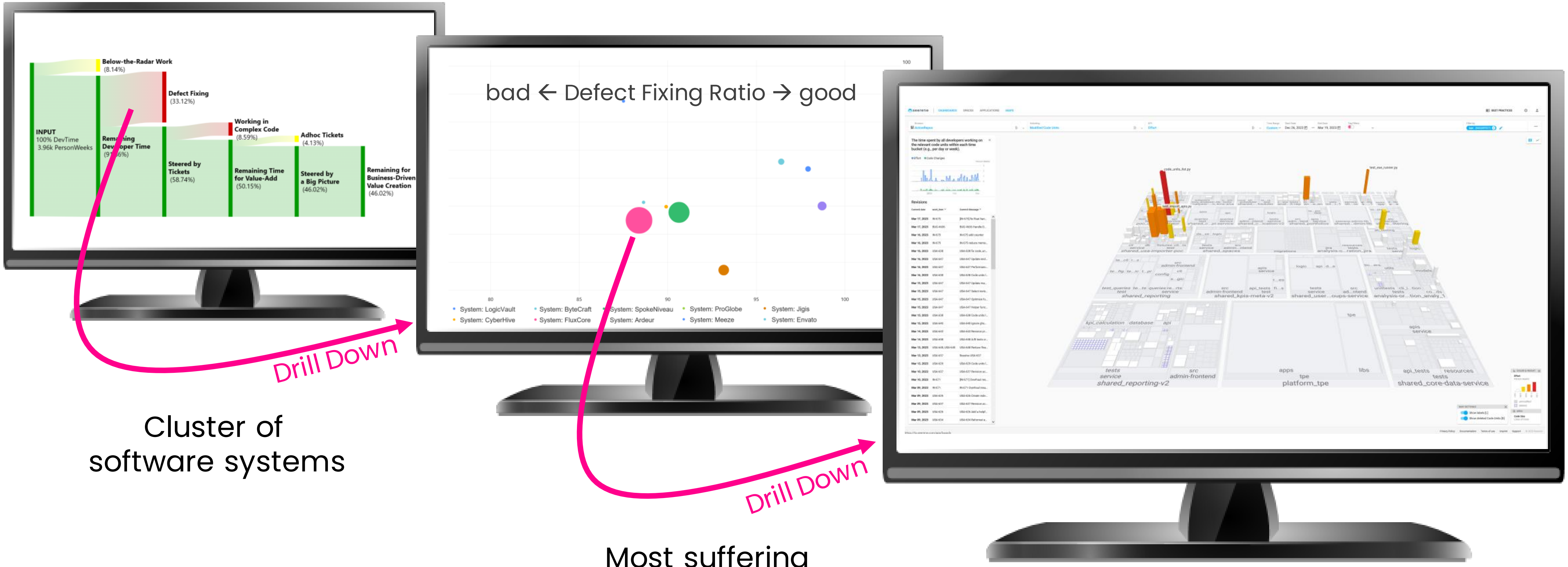
Define
(small and focused)
Improvements

Reveal
Root Causes

DIGITAL BOARDROOM – BUILDING TOP-NOTCH SOFTWARE ORGANIZATIONS

From high-level Executive Views to Details in the Code

Efficiency
Business Unit / Department



Below-the-Radar Work (8.14%)
Defect Fixing (33.12%)
INPUT 100% DevTime 3.96k PersonWeeks
Remaining Developer Time (91.86%)
Working in Complex Code (8.59%)
Adhoc Tickets (4.13%)
Steered by Tickets (58.74%)
Remaining Time for Value-Add (50.15%)
Steered by a Big Picture (46.02%)
Remaining for Business-Driven Value Creation (46.02%)
bad ← Defect Fixing Ratio → good
100
80
85
90
95
100
System: LogicVault
System: ByteCraft
System: SpokeNiveau
System: ProGlobe
System: Jigis
System: CyberHive
System: FluxCore
System: Ardeur
System: Meeze
System: Envato
shared_reporting-v2
admin-frontend
platform_tpe
shared_core-data-service
Drill Down
Drill Down

Cluster of
software systems

Most suffering
software system

Hotspots in the
code architecture

From high-level Executive Views to Details in the Code

Efficiency – Department Level

How efficient is your software factory?

Cluster of
software systems
in department

Most suffering
system: FluxCore

**Defect fixing hotspots
in code architecture**

```
package com.
import ...
public final class LocationUtils {

    /**
     * Parses Point from it's String representation.
     * @param locationString - String that represents location, as 2 double values split with coma. Accepts space after/before
     * @return org.springframework.data.solr.core.geo.Point instance
     */
    public static Point parseLocation(String locationString) {
        Preconditions.checkNotNull(locationString,  errorMessage: "Location String should not be null");
        Preconditions.checkArgument(locationString.contains(","),  errorMessage: "Location must be split with coma");
        locationString = locationString.trim();

        if (locationString.contains(" ,")) {
            locationString = locationString.replaceAll( regex: " ,",  replacement: ",");
        }

        if (locationString.contains(", ")) {
            locationString = locationString.replaceAll( regex: ", ",  replacement: ",");
        }

        String[] location = locationString.split( regex: ",");
        Preconditions.checkArgument( expression: location.length >= 2,  errorMessage: "Location should constist at least 2 Double parameters");
        double lat = Double.parseDouble(location[0]);
        double lon = Double.parseDouble(location[1]);

        return new Point(lat, lon);
    }
}
```

Thanks for Your Attention