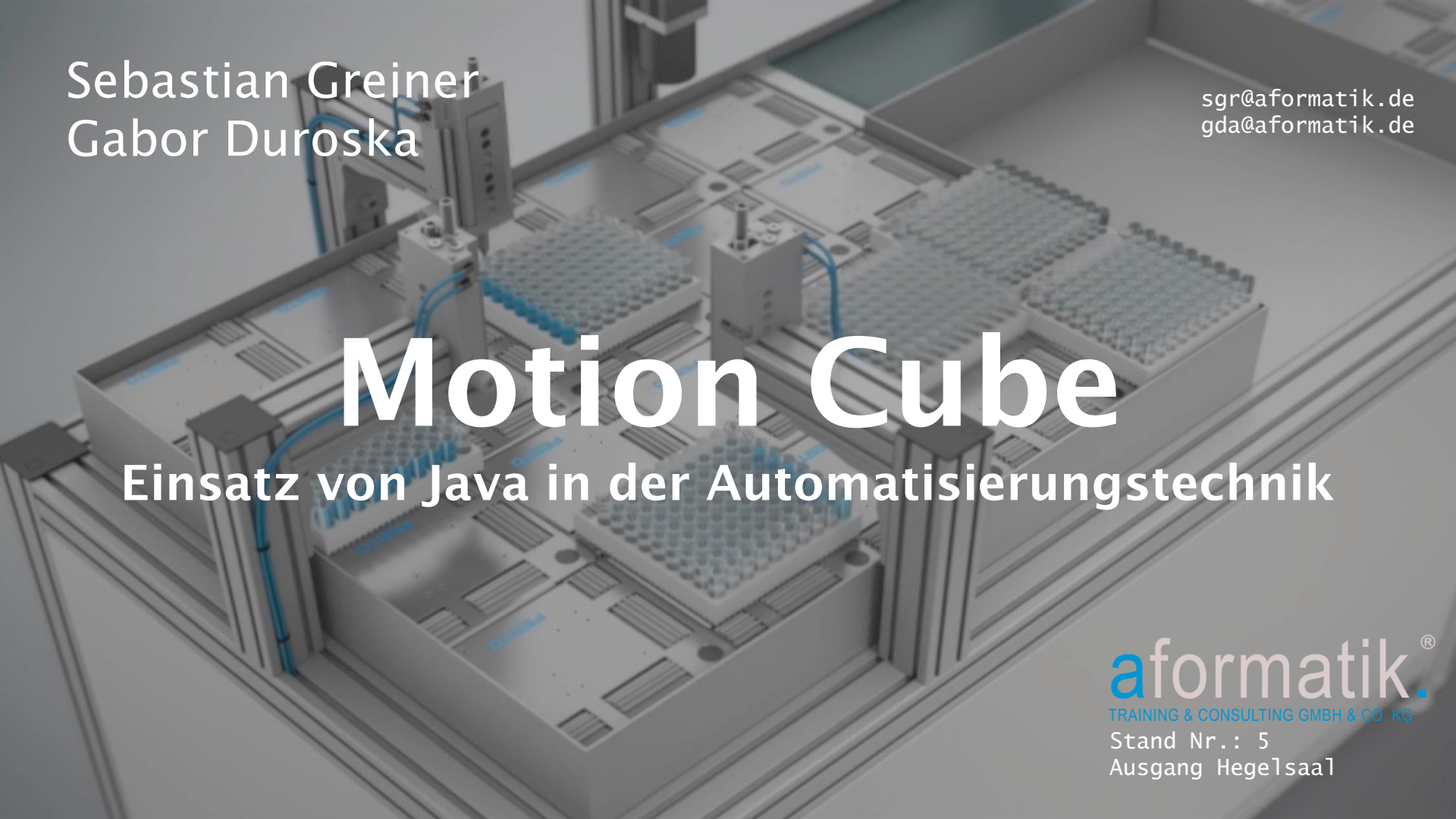


Sebastian Greiner
Gabor Duroska

sgr@aformatik.de
gda@aformatik.de

Motion Cube

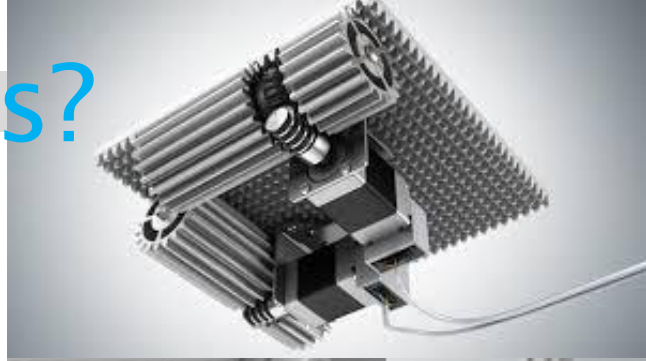
Einsatz von Java in der Automatisierungstechnik

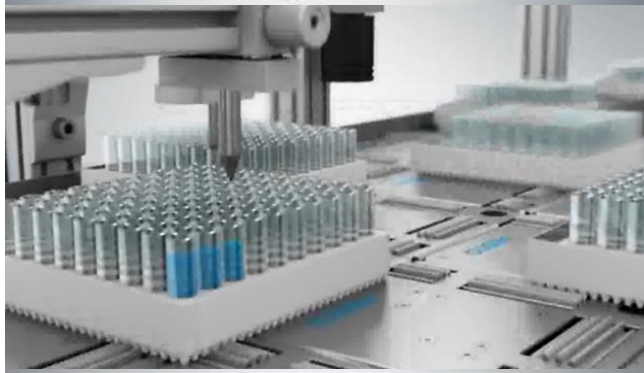
aformatik®
TRAINING & CONSULTING GMBH & CO. KG

Stand Nr.: 5
Ausgang Hegelsaal

Motion Cube. Was ist das?

„Motion Cube ist ein intelligentes und modular aufgebautes Transport-system, welches die Vorzüge der Parallelität und der Flexibilität in sich vereint.“





Use Cases

- Laborautomatisierung
- Elektronikfertigung
- Logistik
- …

Anforderungen

- Parallelität („Multi-Carrier-System“)
- Modularität
- Einfaches Handling
- Geringer Wartungsaufwand
- Anpassungsfähigkeit an sich verändernde Gegebenheiten

Tasks

- Entwicklung eines GUI zur Steuerung, Programmierung und Überwachung
- Entwicklung eines Konfigurationstools
- Entwicklung einer Hardwaresteuerung
- Entwicklung eines Modbus-Interface zur Übertragung von Fahrbefehlen durch ein Hostsystem

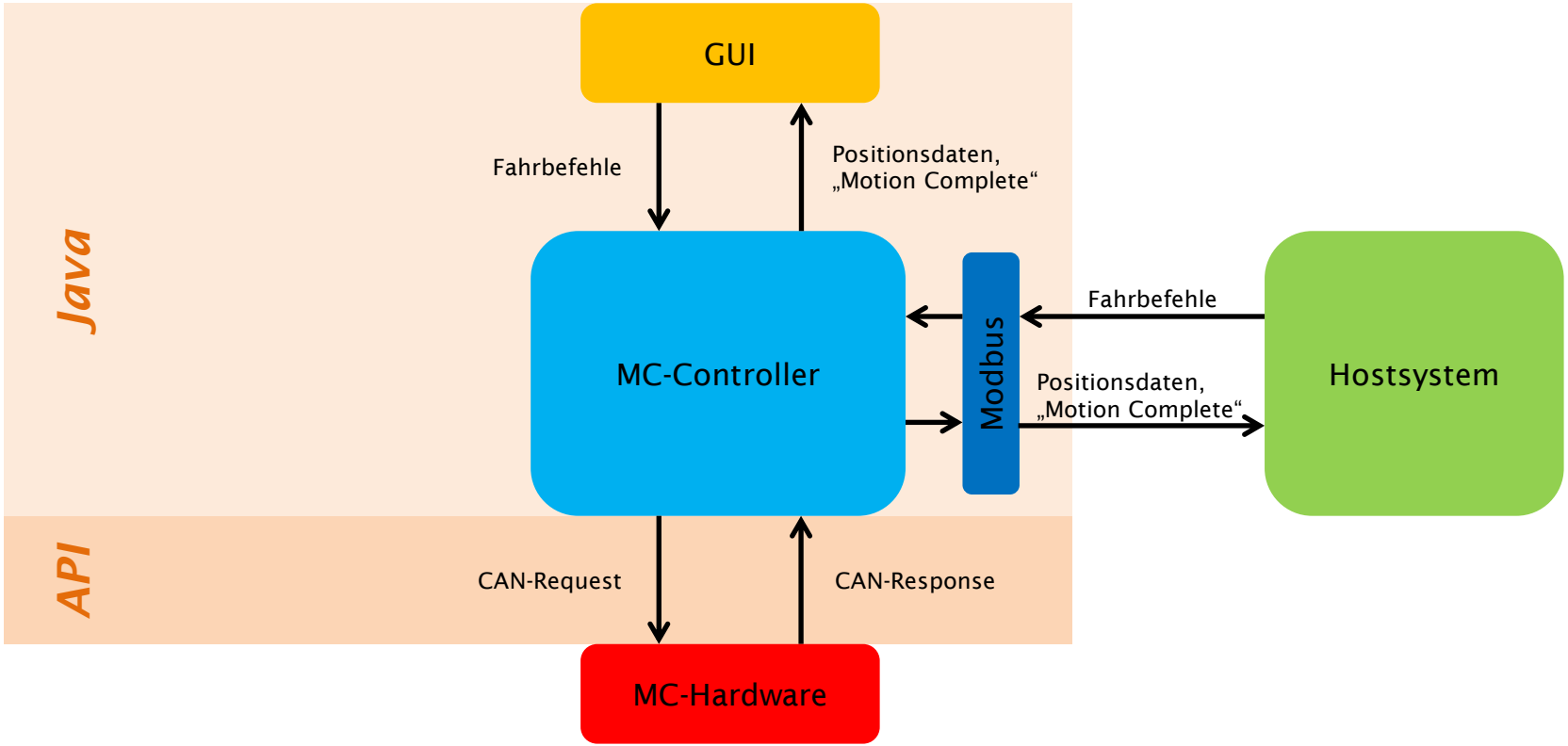
Keyfeatures

- Interaktives GUI zum Definieren und Ausführen von Fahrbefehlen
- Konfigurations- und Simulationsmodus
- Fahrsequenzen
- Berechnung des Fahrweges zwischen zwei Punkten
- Kollisionsdetektion
- Modbus-Interface

Auszeichnung

Motek 2015: 1. Preis in der Kategorie Automatisierung und Robotik

Architekturmodell



Funktionsweise (1)

- MC-Controller empfängt Fahr-sequenzen von dem GUI
- Berechnung eines Fahrweges
- Validierung der Fahrsequenzen und Zerlegung in einzelne Fahrbefehle
- Generierung von CAN-Steuerbefehlen für die am Fahrauftrag beteiligten Motoren

Funktionsweise (2)

- Ausführung der Steuerbefehle
- Falls Ziel erreicht: Rückmeldung „Motion Complete“
- Falls Ziel nicht erreicht: Speicherung des verbleibenden Fahrauftrags in der Queue

Ansteuerung

- CAN-Befehle werden als 8-Byte-Arrays übertragen
- Alle CAN-Befehle sind im Code als Konstanten hinterlegt

```
public static final byte[] START               = new byte[]{(byte) 0x2B, (byte) 0x40, (byte) 0x60,
       (byte) 0x00, (byte) 0x1F, (byte) 0x00, (byte) 0x00, (byte) 0x00};

public static final byte[] SWICHTED_ON_ENABLED = new byte[]{(byte) 0x2B, (byte) 0x40, (byte) 0x60,
       (byte) 0x00, (byte) 0x07, (byte) 0x00, (byte) 0x00, (byte) 0x00};
```

Fahrsequenzen (1)

- Fahrsequenzen sind Abfolgen von Fahrbefehlen und Delays
- Fahrsequenzen können geschachtelt werden
- Eine Fahrsequenz kann zeitgleich mehrere Carrier ansprechen

Fahrsequenzen (2)

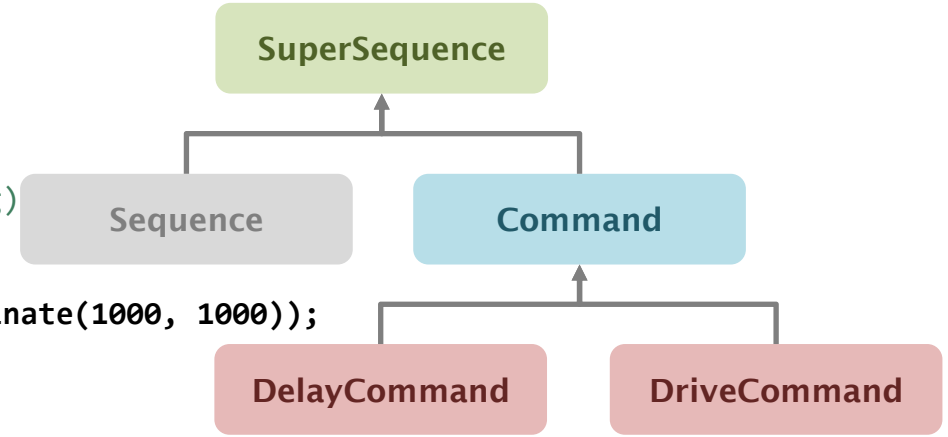
```
// Sequenz wird erstellt
Sequence mainSequence = new Sequence();
mainSequence.setName("Test Sequenz");
mainSequence.setRepeat(2);

// Fahrbefehl wird erstellt (Absolute Positionierung)
DriveCommand driveCommand1 = new DriveCommand();
driveCommand1.setCarrier(1);
driveCommand1.setAbsoluteDestinationCoord(new Coordinate(1000, 1000));

// Delay mit einer Sekunde
DelayCommand delayCommand = new DelayCommand();
delayCommand.setCarrier(2);
delayCommand.setMillis(1000);

// Fahrbefehl wird erstellt (Relative Positionierung)
DriveCommand driveCommand2 = new DriveCommand();
driveCommand2.setCarrier(2);
driveCommand2.setRelativeDestinationCoord(new Coordinate(50, 50));

// Fahrbefehle und Delays werden der Sequenz hinzugefügt
mainSequence.add(driveCommand1);
mainSequence.add(delayCommand);
mainSequence.add(driveCommand2);
```



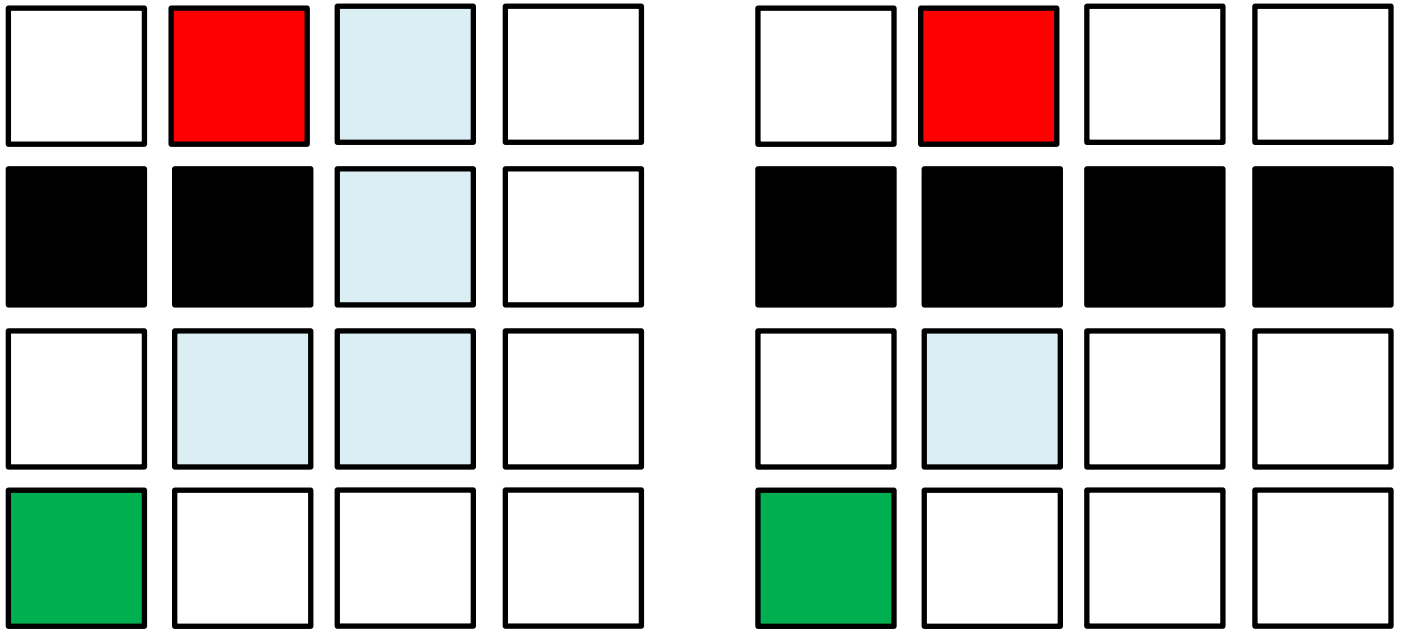
Pathfinder (1)

- Basiert auf dem A*-Algorithmus
- Jeder Motion Cube wird durch einen Eintrag in der Matrix repräsentiert
- Nicht vohandene Cubes bzw. durch andere Carrier belegte Cubes werden in der Matrix als Hindernisse gekennzeichnet

Pathfinder (2)

- Fahrweg wird komplett vor Beginn der Fahrt berechnet
- Es werden nur ganze Cubes berücksichtigt
- Teilfahrten von bzw. auf ganze Cubes werden in einem Pre- bzw. Postprocessingschritt erledigt

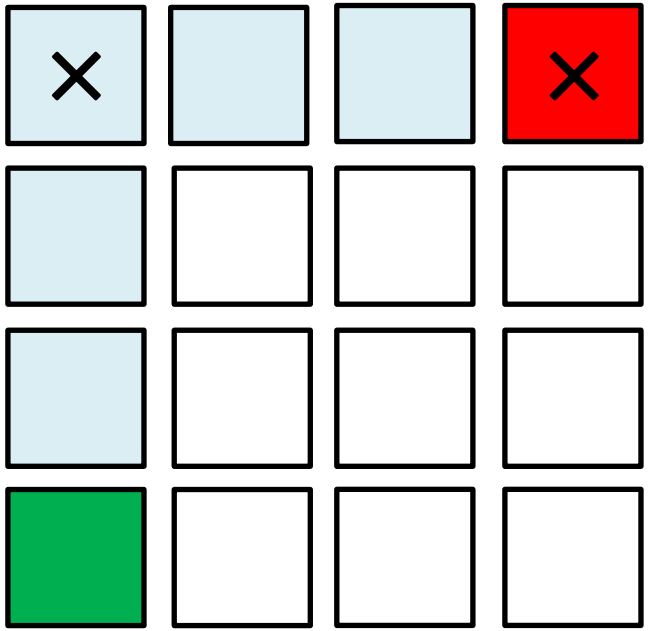
Pathfinder (3)



Live-Tracking (1)

- Zu hoher Traffic auf dem CAN-Bus bei stetiger Positionsrückmeldung aller Motion Cubes
- $\Rightarrow$ Position der Carrier muß simuliert werden
- Synchronisation mit der Hardware erfolgt immer bei Zielerreichung einer zusammenhängenden Teilbewegung

Live-Tracking (2)



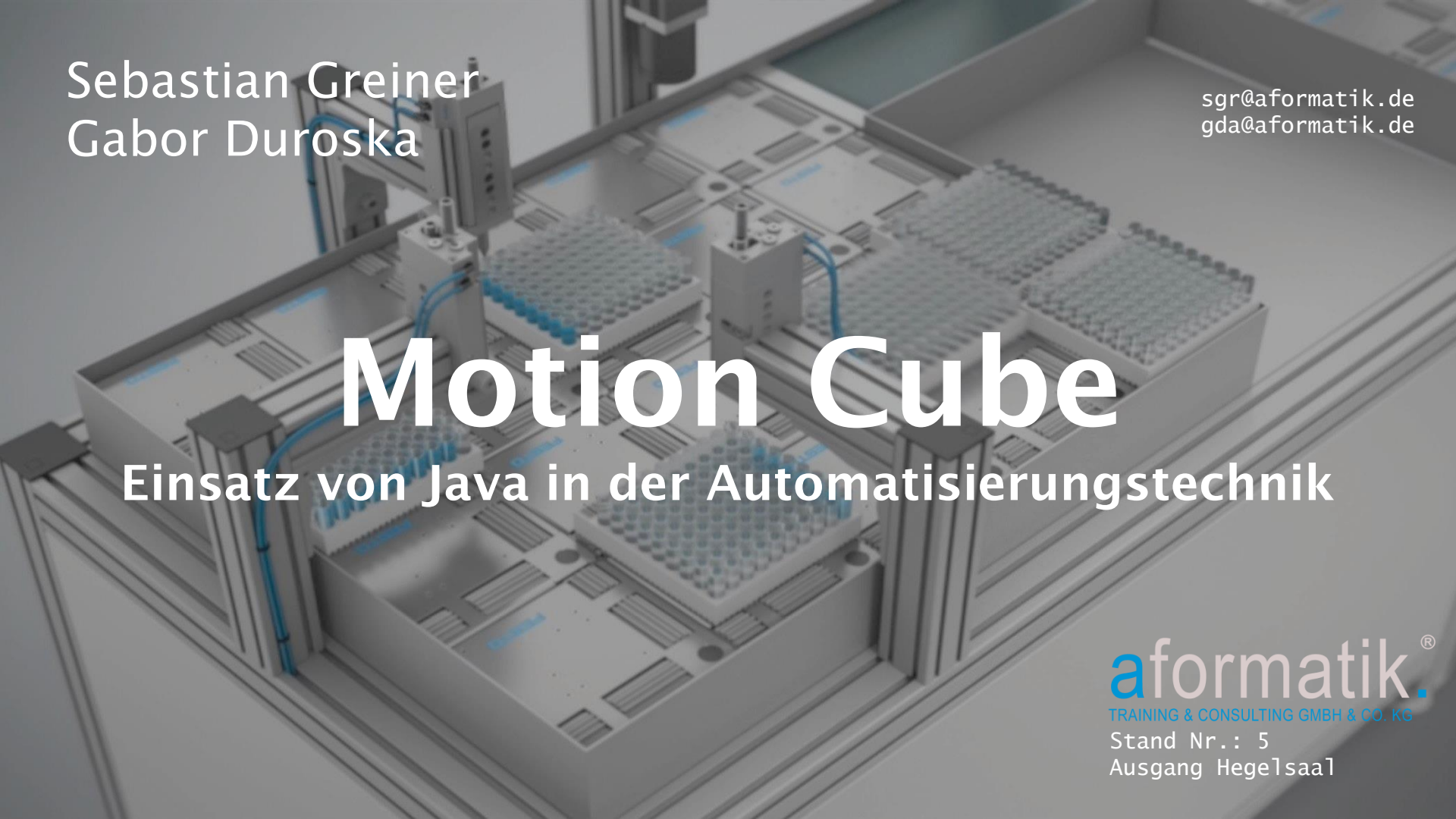
✕ Synchronisationspunkt

Warum Java?

- Java-Skills vorhanden
- Komplettes GUI-Framework enthalten
- Breite Palette an weiteren Frameworks
- Vorteile einer High-Level-Programmier-sprache
- Alle Java SE-Features verfügbar (Collections, Date & Time-Utils,…)
- Plattformunabhängigkeit

Warum nicht mit Java?

- Keine Echtzeitunterstützung vorhanden
- Kein aktuelles API verfügbar
- Code läßt sich auf Embedded-Systemen möglicher Weise nicht ausführen
- Höherer Ressourcenbedarf



Sebastian Greiner
Gabor Duroska

sgr@aformatik.de
gda@aformatik.de

Motion Cube

Einsatz von Java in der Automatisierungstechnik

aformatik®
TRAINING & CONSULTING GMBH & CO. KG

Stand Nr.: 5
Ausgang Hegelsaal