1.– 4. September 2014
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Berühren verboten!

Berührungsfreie Interfaces in der industriellen Fertigung

Hariolf Betz / Stefan Partsch

eXXcellent solutions gmbh / Wieland-Werke AG

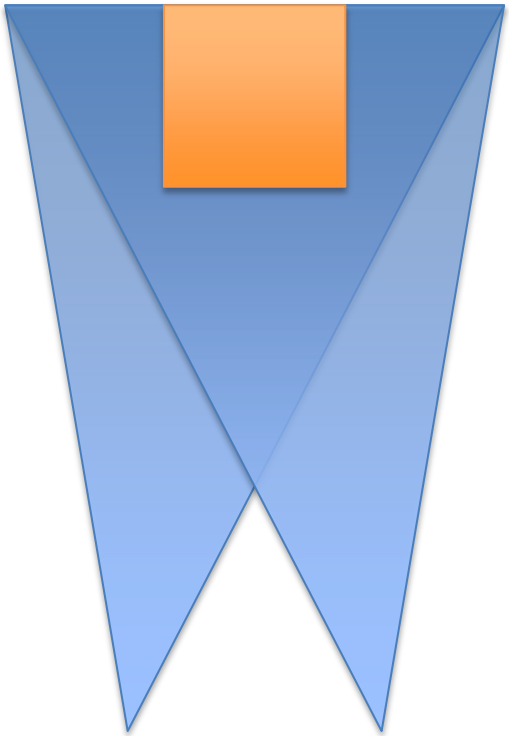
Agenda

1. Einführung
   - Was ist/kann die Kinect?
2. Gestenerkennung
   - Wie kann ich Gesten auf der Kinect erkennen?
3. Fallstudie: Kinect im industriellen Umfeld
   - Anforderung und Ergebnis
   - Evaluation und Vorführung

1. Einführung

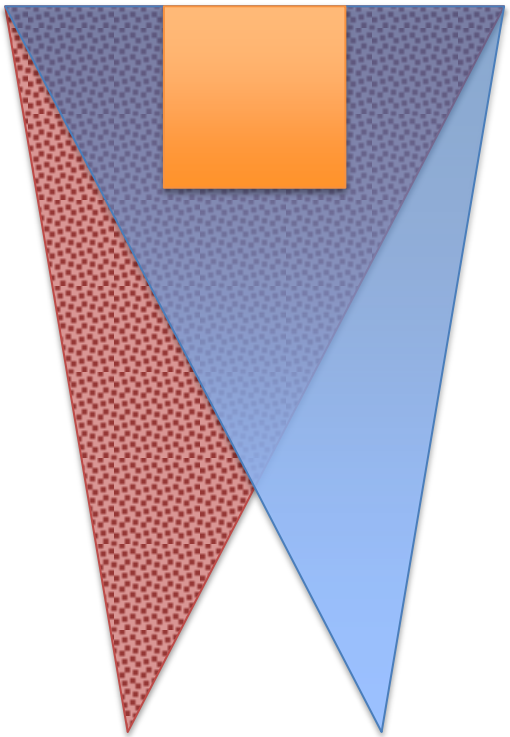
Tiefen-Sensoren: Technische Grundlagen

- Stereovision



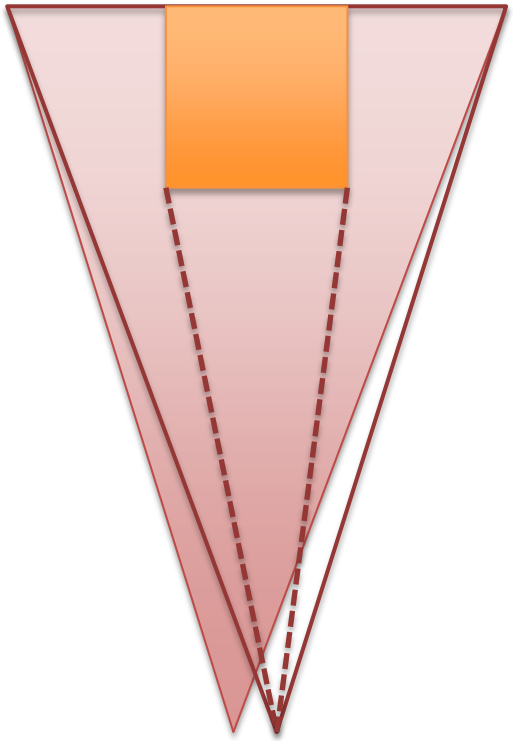
Tiefen-Sensoren: Technische Grundlagen

- Structured Light/Light Array

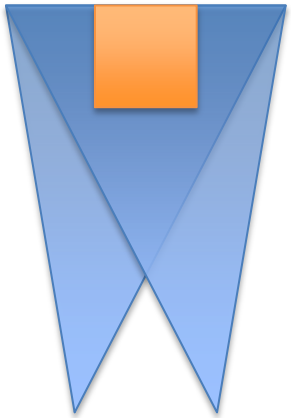


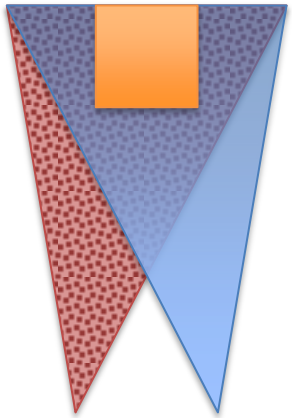
Tiefen-Sensoren: Technische Grundlagen

- Time-Of-Flight (TOF)

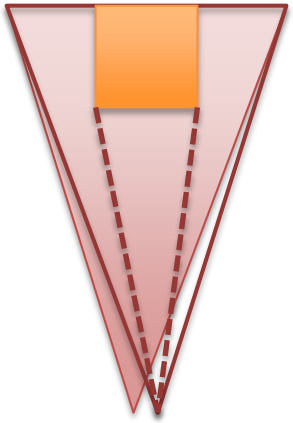


Tiefen-Sensoren: Vergleich





z.B. Kinect v1



z.B. Kinect v2

Geschichte der Kinect (SDKs)

- 06/2009: Ankündigung „Project Natal“
  - „Light Coding“-Sensoren von PrimeSense

- 11/2010: Kinect for Xbox 360
- 12/2010: Gründung OpenNI
  - PrimeSense veröffentlicht Treiber (NiTE)
- 06/2011: Microsoft Kinect SDK
- 02/2012: Kinect for Windows

- 11/2013: Xbox One (inkl. Kinect v2)
- 11/2013: Apple kauft PrimeSense

Frameworks

- OpenNI/NiTE
  - NiTE – Middleware (= natural interaction engine)
  - OpenNI – SDK (= open natural interaction)
  - C++ (C# und Java-Wrapper)
  - OpenNI.org seit 04/2014 offline

- Kinect for Windows SDK
  - C++, C#, Visual Basic
  - aktuelle Version: 2.0 (mit Unterstützung für Kinect v2)

2. Gesteinerkennung

Prototyping mit FAAST

- Middleware
  - Virtual Reality Peripheral Network (VRPN)
  - stellt Skelett-Daten über VRPN-Server zur Verfügung

- Input Emulator
  - … erlaubt das Erstellen eigener Gesten
  - … löst virtuelle Tastatur-/Mausevents aus.

Gestenerkennung: Kinect SDK

1. Aktiviere Stream
   - RGB-Bild
   - Tiefenbild
   - Skelett-Informationen
2. Registriere EventHandler an Datenquelle
3. Starte Sensor

```
mySensor.SkeletonStream.Enable();
mySensor.SkeletonFrameReady +=
     new EventHandler<SkeletonFrameReadyEventArgs>(
          myHandler);
mySensor.Start();
```

Gestenerkennung: Kinect SDK

- Event, sobald neues Frame vorliegt
- 20 Gelenke (Joint) pro Skelett
- pro Frame und Gelenk eine Position im Raum
  - x-, y-, z-Koordinate

```
JointCollection myJoints = mySkeleton.Joints;
float handX = myJoints[JointType.HandLeft].Position.X;
float elbowX = myJoints[JointType.ElbowLeft].Position.X;

if (handX - elbowX > 0.2)
{
    // do something
}
```

Gestenerkennung: Kinect SDK Tooling

- Hand Pointer Gestures
  - integriert in Windows Presentation Foundation (WPF)
  - „KinectRegion“
  - berührungsfreie Mauszeiger-Steuerung
- Gesture Builder (seit 2.0)
  - Gestendefinition durch maschinelles Lernen

Open Source: FUBI

- Full Body Interaction Framework
- Open Source
- EU-Projekt
- Definition von Gesten:
  - C++/C# …
  - … oder XML.

```
<JointRelationRecognizer name="LeftHandLeftOfElbow">
  <Joints main="leftHand" relative="leftElbow"/>
  <Relation type="left" min="200"/>
</JointRelationRecognizer>
```

Fazit: Gestenerkennung

1. „von Hand“
2. Konfiguration über Tooling
3. Maschinelles Lernen

➔ keine One-size-fits-all-Lösung

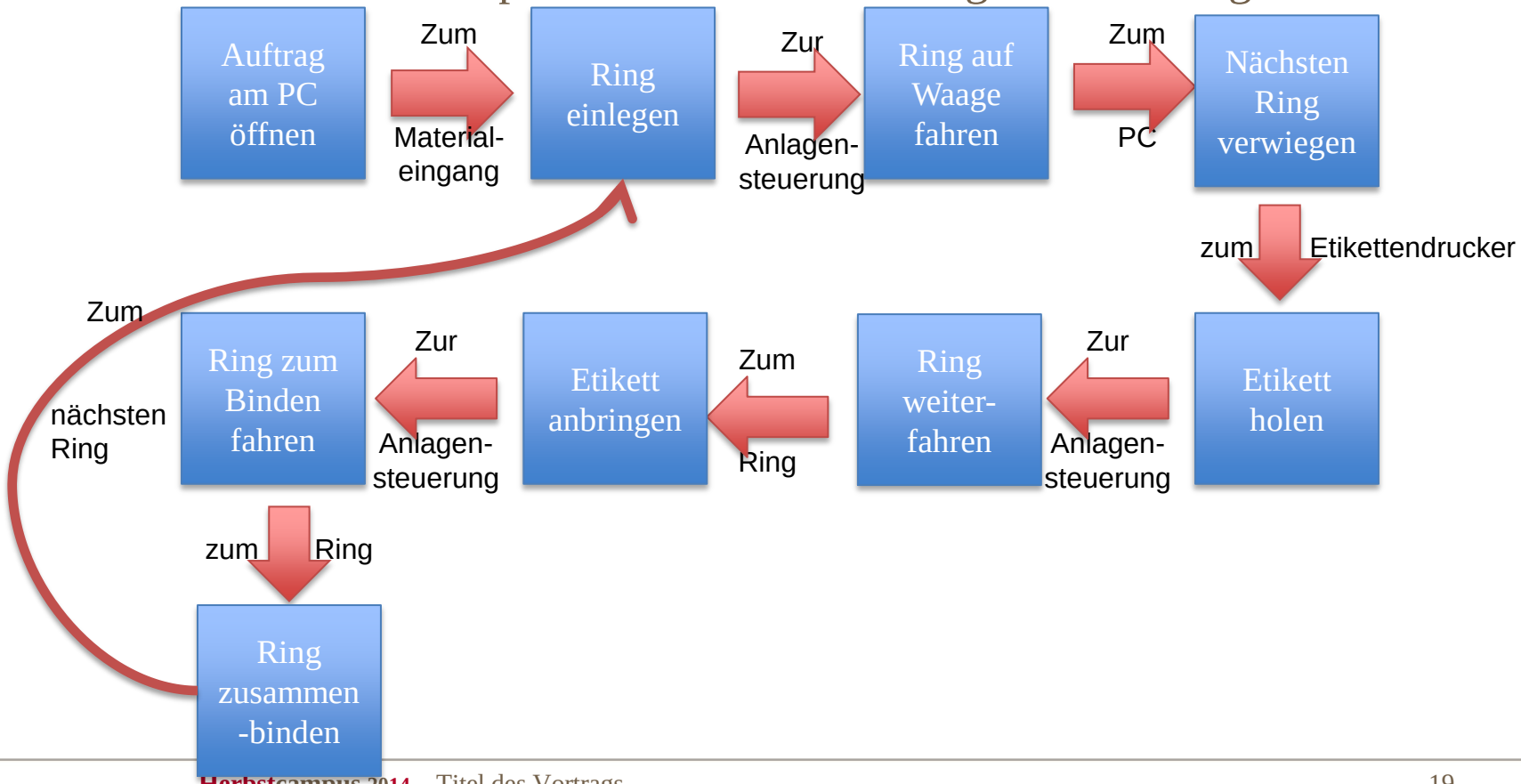
3. Fallstudie

Feldversuch

- Anwendungsfall
- Realisierung der Gestensteuerung
- Integration in die bestehende Anwendung
- Feldversuche
- Fazit

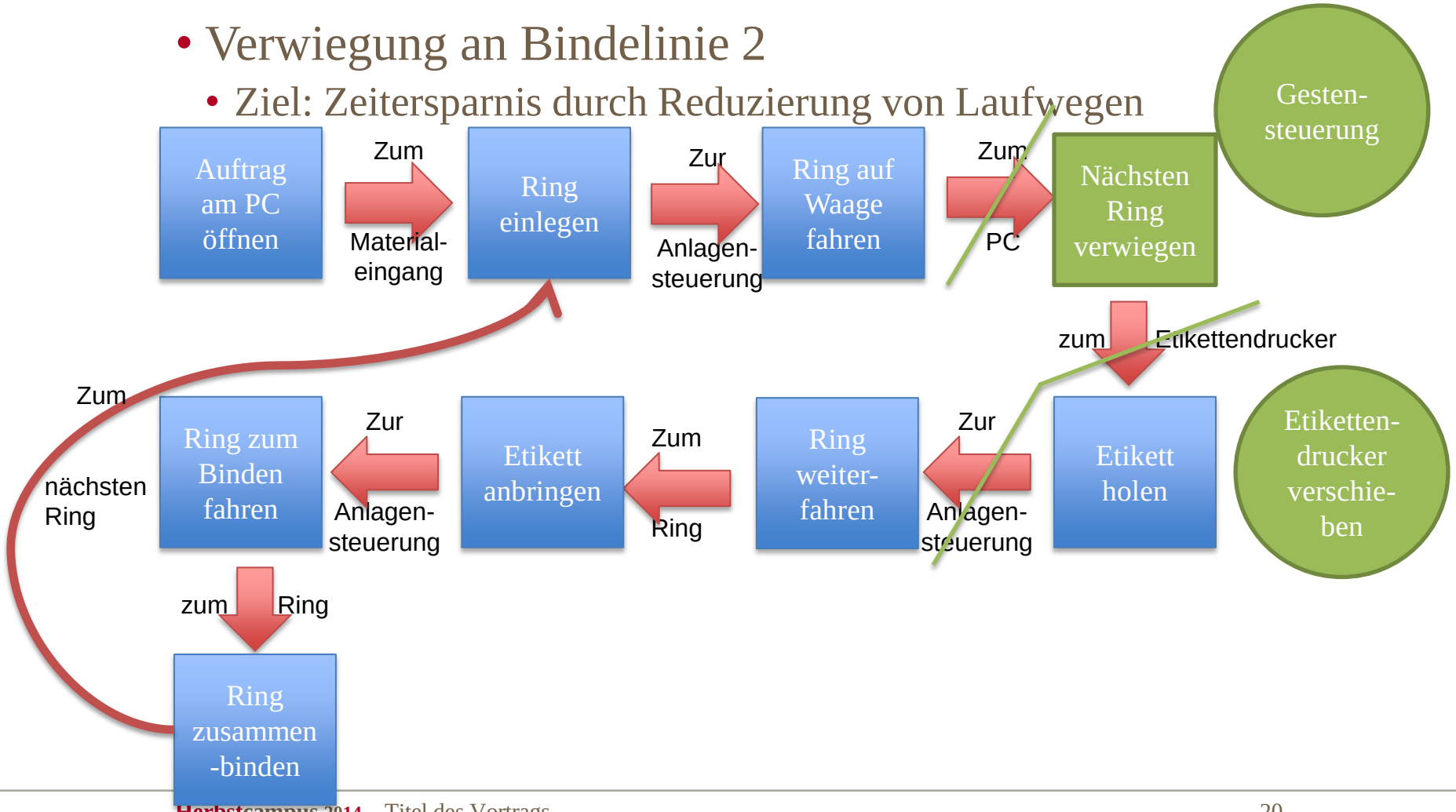
Feldversuch - Anwendungsfall

- Verwiegung an Bindelinie 2
  - Ziel: Zeitersparnis durch Reduzierung von Laufwegen



Feldversuch - Anwendungsfall

- Verwiegung an Bindelinie 2
  - Ziel: Zeitersparnis durch Reduzierung von Laufwegen



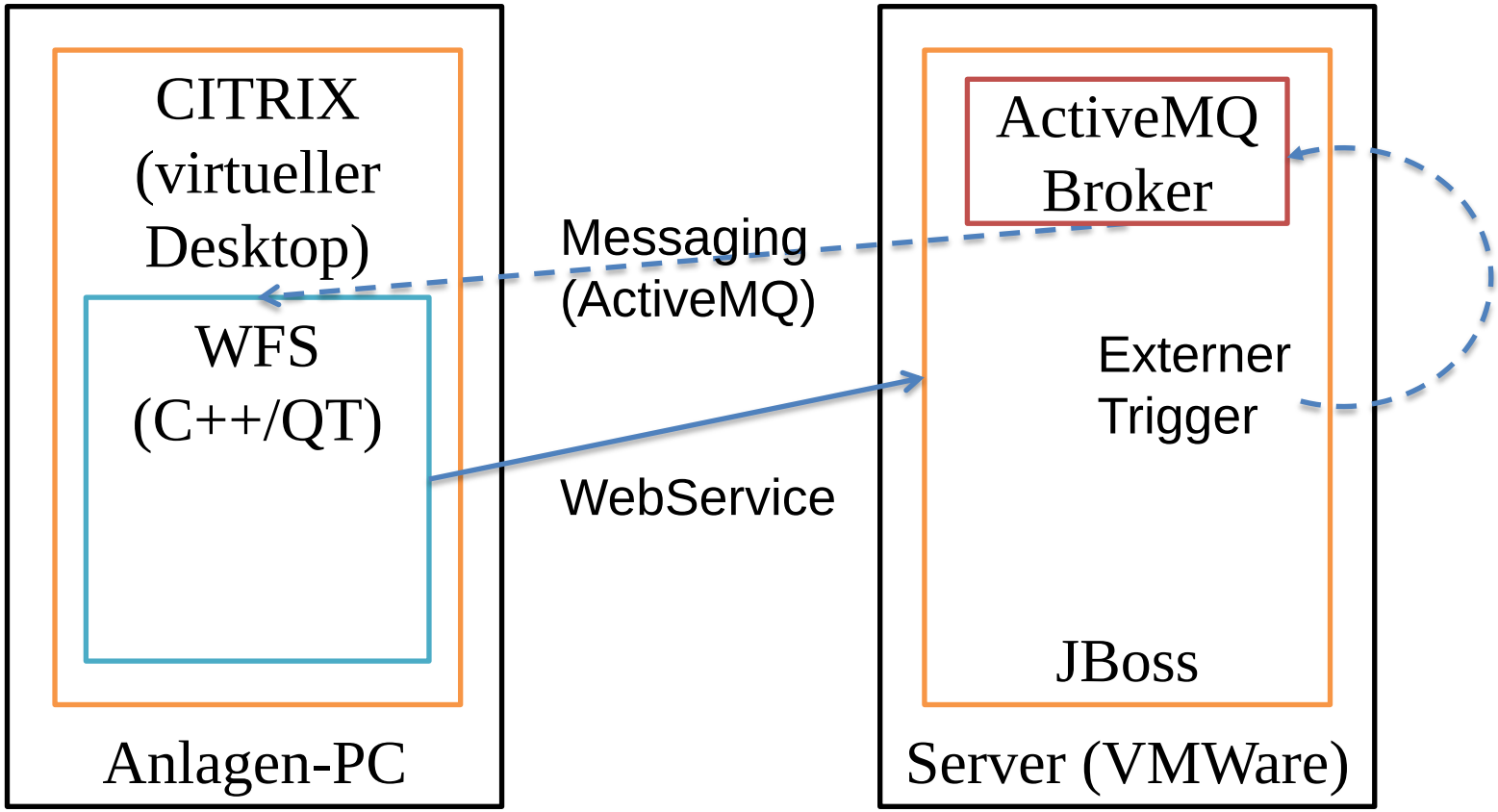
Feldversuch - Anwendungsfall

- Umgebungsbedingungen:
  - Wechselhafte Beleuchtung
  - Personen im Umfeld
  - Reflektionen (Umgebung, vorbeifahrende Stapler)
  - Staub/Schmutz

- Anforderungen
  - Erkennungsrate >= 96%
  - Geringe Zeitverzögerung (<300ms)
  - Sehr geringe Einlernvorgänge
  - Intuitive, ergonomisch durchführbare Gesten
  - Robustheit gegenüber Umgebungsbedingungen

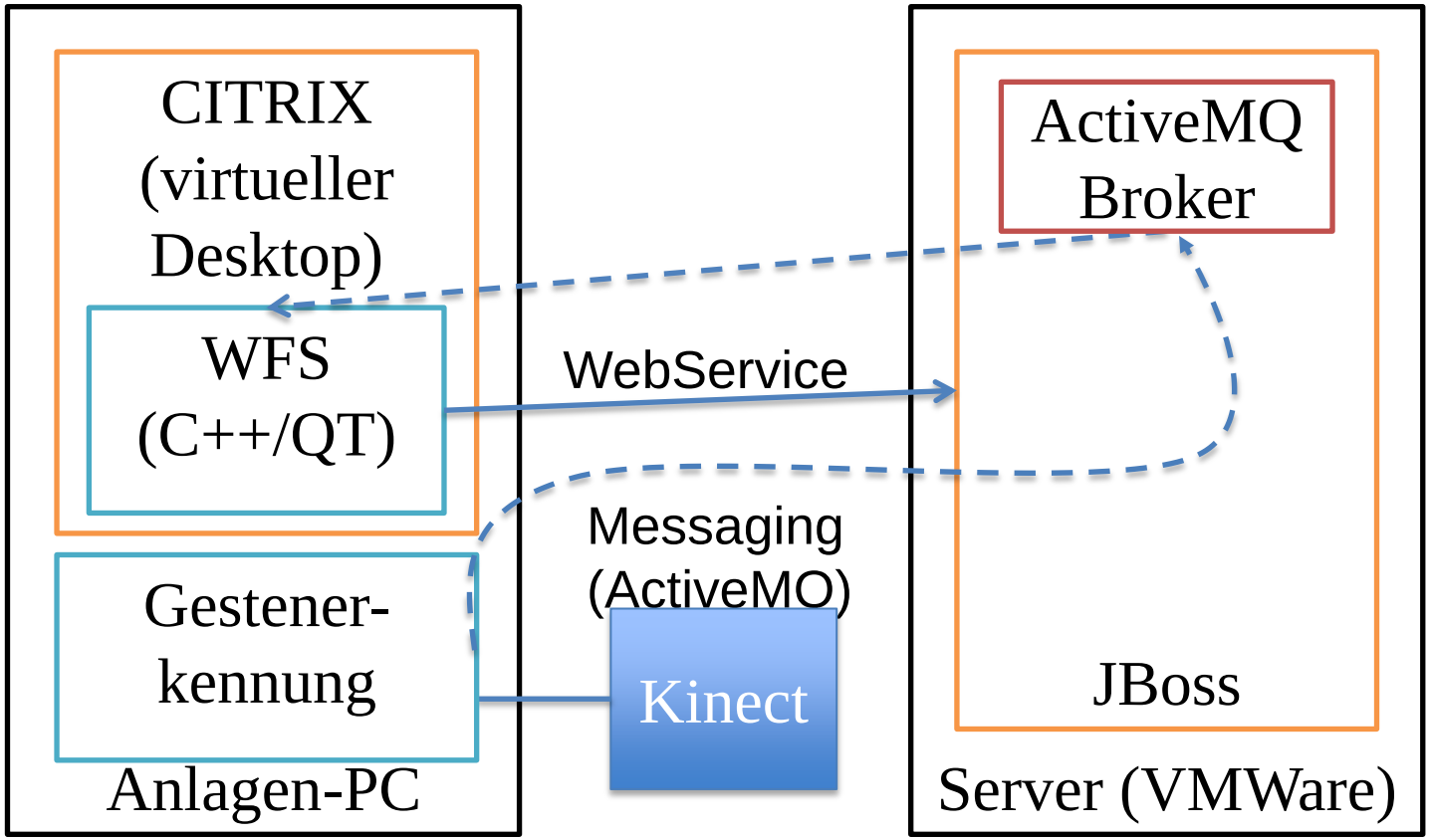
Feldversuch – Integration in Anwendung

Aktuelle Systemlandschaft



Feldversuch – Integration in Anwendung

Integration Kinect



Feldversuch – Integration in Anwendung

- Gestenerkennung als separater Prozess auf PC
- Kommunikation mit Backend und/oder Client per Messaging (ActiveMQ)
- Implementierung unabhängig von Client-Technologie (C++ / Web),
  z.B. C#-Bibliotheken einsetzbar
- Trennung Kinect-PC / Visualisierungs-PC möglich

Feldversuch - Ergebnisse

- Erster Feldversuch
  - Ziele:
    - Untersuchung der Störeinflüsse
    - Positionierung der Kinect prüfen
  - Beschreibung:
    - Visualisierung der Handflächen (erkannt / nicht erkannt)
    - Rudimentäre Erkennung einer Hakengeste
    - Wolkig → Keine Änderung der Lichtverhältnisse

Feldversuch - Ergebnisse

- Erster Feldversuch
  - Ergebnisse
    - Handerfassung zu 97,5 %, Interpolation gut
    - Gestenerkennungsrate 72%
    - Mehrere Personen erfasst
    - Anleitung des Benutzers mit 5 Minuten zu lang
    - Häufige irrtümliche Gestenerkennung
  - Maßnahmen
    - Beschränkung auf eine Person implementieren
    - Gestenerkennung verbessern
    - Diskrete Gesten

Feldversuch – Realisierung der Gestensteuerung

- Einschränkung auf erste Person im Sichtfeld
- Diskrete Gesten mit abgegrenztem Start/Ende
- $N-Recognizer
  - Bibliothek zur Erkennung von Touch-Gesten
  - Abgleich mit hinterlegten Templates
    - 2 Gesten: Wischen, Haken

Feldversuch - Ergebnisse

- Zweiter Feldversuch
  - Ziele:
    - Verbesserte Gestenerkennung untersuchen
  - Beschreibung:
    - Alternativer Ansatz der Gestenerkennung
    - Wechselnde Lichtverhältnisse

Feldversuch - Ergebnisse

- Zweiter Feldversuch
  - Ergebnisse
    - Gestenerkennungsrate
      - 65% bei direkter Sonneneinstrahlung
      - 84% ohne Sonneneinstrahlung
      - Weiterhin starke Beeinflussung durch weitere Person
    - Irrtümliche Gestenerkennung in 20% der Ruhepausen

      ➔ Verbessung nach Anleitung des Anwenders

  - Maßnahmen
    - Position des Sensors anpassen (Sonneneinstrahlung)
    - Mehrere Templates für Geste

Feldversuch - Ergebnisse

- Dritter Feldversuch
  - Ziele:
    - Integration in Wieland Factory Suite (WFS) untersuchen
    - Test unter realen Bedingungen
  - Beschreibung:
    - Gestenerkennung weiter verbessert
    - Integration in WFS mittels Messaging
    - Wechselnde Lichtverhältnisse
    - Sensor von der Sonneneinstrahlung weg gerichtet

Feldversuch - Ergebnisse

- Dritter Feldversuch
  - Ergebnisse
    - Gestenerkennungsrate 83%
    - Ersparnis je Arbeitsdurchgang
      - Laufweg: 17,5m → 9,5m (46%)
      - Zeit: 70s → 55s (21%)
    - Unterscheidung zweier Personen, wenn diese > 20 cm entfernt stehen
    - Integration in WFS ohne nennenswerte Verzögerung
    - Negative Einflüsse durch Sonneneinstrahlung lässt sich durch entsprechende Positionierung vermeiden

Fazit

- Gestenerkennung
  - Erkennungsrate nicht ausreichend
  - Wechselnde Lichtverhältnisse problematisch
  - Störeinflüsse durch Umgebung und Personen gering
  - Einfache Gesten intuitiv erlern- und ausführbar
    - Expliziter Start/Stop anfangs ungewohnt
  - Komplexere Interaktion mit Gesten schwierig
    - Listenauswahl beschränkt möglich
    - Eingaben nicht möglich

➔ Weitere Forschung und Verbesserung von Algorithmen erforderlich

Fazit

- Feldversuche
  - Positive Resonanz von Werksleitung und Anlagenbedienern
  - Einsparpotenzial gegeben, in anderen Anwendungsfällen noch mehr Potenzial vorhanden
  - Ggf. Umorganisation der Arbeitsplätze nötig
  - Alternativen zu Gestensteuerung in den Anwendungsfällen
    - Automatisierte Anlagenkommunikation
    - Mobile Devices
    - Funkmaus

➔ Interessantes Forschungsthema

Live-Demo

- Gestenerkennung
- Integration in Wieland Factory Suite

Vielen Dank!

Hariolf Betz / Stefan Partsch

eXXcellent solutions gmbh / Wieland-Werke AG