# Konzeption migrierbarer Benutzungsschnittstellen in der industriellen Automatisierungstechnik

# Application of Migratory User Interfaces in Industrial Automation

Dipl.-Ing. L**. Baron;** PD Dr.-Ing. A. **Braune**,
TU-Dresden, Institut für Automatisierungstechnik

**Kurzfassung**

Die zunehmende Gewöhnung von Benutzern an neue Interaktionskonzepte und Endgeräte ermöglicht deren Einführung in industriellen Umgebungen. Daraus folgen Anwendungsszenarien, in denen es, selbst während der Bearbeitung einer einzelnen Arbeitsaufgabe, zu häufigen Änderungen in der Zusammensetzung der verwendeten Geräte kommt. Dies motiviert die Entwicklung migrierbarer Benutzungsschnittstellen (MUI). In diesem Beitrag stellen wir zunächst die anerkannte Theorie der MUIs vor, inklusive verschiedener Klassifikationsmerkmale und spiegeln diese an den Anforderungen der Automatisierungstechnik. Anhand dessen diskutieren wir anschließend zwei Anwendungsszenarien. Die Analyse verwandter Arbeiten zeigt auf, dass existierende Ansätze nur eingeschränkt in diesen Szenarien eingesetzt werden können. Am Ende stellen wir eine Fallstudie vor, die die Anwendbarkeit von MUIs in industriellen Prozessvisualisierungen demonstriert.

**Abstract**

Due to familiarization of users with modern interaction concepts and devices, they become interesting for industrial environments as well. These devices enable use cases where users change the set of applied devices, even during handling one single task. This fosters the design of migratory user interfaces (MUI) which can be transferred freely between devices, in order to follow according to a user's device changes. Hence, in this paper the generally accepted theory, including a set of identified classifiers for MUIs, is being analyzed with respect to the demands of the domain of industrial process visualizations. Moreover, we discuss two use cases. Our review of the related work revealed only a limited applicability in those use cases. In order to demonstrate an MUI's usefulness in industrial process visualizations, we finally present our own case study

## 1. Einleitung

Die zunehmende Gewöhnung von Benutzern an neue Konzepte der Mensch-Maschine-Interaktion, wie z.B. Gestensteuerung auf Touch-fähigen Geräten, erleichtert deren Einführung in industrielle Umgebungen. Das herkömmliche Design industrieller Visualisierungslösungen sieht üblicherweise eine feste Zusammensetzung und Zuordnung von Aufgaben zu den verwendeten Bediengeräten vor. Mobile Endgeräte bringen jedoch ein deutlich breiteres Spektrum an Anwendungsszenarien mit sich, wie zum Beispiel der Vor-Ort-Unterstützung von Fehleranalysen, Reparaturen oder Inbetriebnahmen, in denen ein flexiblerer technischer Entwurf für verschiedene Anlagenteile und Aufgaben erforderlich ist. Darüber hinaus sind mobile Geräte durch eine große Vielfalt bezüglich der Eigenschaften (z.B. Bildschirmgröße), Bedienphilosophien und der Firmware gekennzeichnet. Dadurch ist die Entwicklung von Benutzungsschnittstellen (UI) motiviert, die frei zwischen Bediengeräten transferiert werden können, um so den Gerätewechseln der Benutzer ohne Informationsverlust zu folgen, d.h. der Transfer des UIs sollte die Arbeit des Benutzers so wenig wie möglich unterbrechen. Solche UIs werden in der Literatur als migratorische oder nomadische Benutzungsschnittstellen (MUI) bezeichnet [1]. Diese sind im Bereich multimedialer Anwendungen [9], [10] und Heimautomatisierung [12] oder kollaborativer Lernsoftware [11] bereits erforscht, im Bereich industrieller Prozessvisualisierungen aber noch nicht etabliert.

In diesem Beitrag stellen wir zunächst in Abschnitt 2 die anerkannten Grundlagen migratorischer UIs vor, die vor allem verschiedene Klassifikationsmerkmale umfassen, und spiegeln diese in Abschnitt 3 an den Anforderungen der Automatisierungstechnik. Anschließend erläutern wir in Abschnitt 4 zwei Szenarien, in denen Migrationsmechanismen in einem industriellen Umfeld angewendet werden könnten. Schließlich präsentieren wir dazu in Abschnitt 5 eine Fallstudie unter Verwendung einer bereits existierenden Visualisierungslösung.

## 2. Migratorische Benutzungsschnittstellen

### 2.1 Einführung

Nach [2] werde ein User Interface als migratorisch bezeichnet, wenn es sich von einem Quellgerät, auf dem ein Benutzer mit der Interaktion begonnen hat, auf ein oder mehrere bestimmte Zielgeräte transferieren lässt, auf denen die Interaktion sofort ohne Informationsverlust fortgesetzt werden kann. Das wesentliche dabei sei, dass der Transfer nicht nur die Artefakte umfasst, die zur Darstellung des UIs auf dem Zielgerät benötigt werden, sondern auch den UI Zustand. Dieser bestehe aus der gesamten Historie der Benutzung des UIs. Damit umfasse dies aufgerufene Dialoge, ein- und ausgegebene Daten, aufgerufene Funktionen, etc. Der Zweck des Zustandstransfers sei es eine Kontinuität bei der Benutzung des UIs nach einem

Migrationsvorgang zu erreichen. Wenn sich Quell- und Zielgeräte in Hard- oder Software unterscheiden, erfordere dies gegebenenfalls Adaptionsmechanismen.
Aspekte verteilter UIs müssen nach [1] dann berücksichtigt werden, wenn ein UI zur Darstellung eines bestimmten technischen Prozesses auf mehrere Bediengeräte aufgeteilt wurde. Eine Änderung dieser Verteilung (dynamisch – zur Laufzeit, statisch – in einer Entwurfsphase) verlange dann nach einem Migrationsmechanismus, wenn die Benutzungskontinuität sichergestellt werden soll. Die Autoren führen weiter aus, dass die Verteilung und damit auch ein potentieller Migrationsmechanismus auf verschiedenen Ebenen der softwaretechnischen Implementierung realisiert sein könnten.

1) Auf **Anwendungsebene** würden sich mehrere Geräte an der Verwaltung eines User Interfaces beteiligen.
2) Auf **Workspace-Ebene** sei das UI verteilt, wenn auf mehrere unabhängige Sammlungen an Ressourcen (Dateisysteme, Prozessanbindungen, Hintergrundwerkzeuge, etc.) zugegriffen werde.
3) Auf **Interaktorenebene** könne ein UI verteilt sein, wenn mehrere Geräte zur Ein- und Ausgabe verwendet würden. Solche Geräte sind beispielsweise Monitore, Tastaturen, Handscanner, etc.

Damit ergeben sich die drei wesentlichen Aspekte eines migratorischen User Interfaces:

1) **Verteilung**: Der Transfer statischer Artefakte zur Darstellung des UIs (bei einer Verteilung auf Anwendungsebene) oder auch das Einbinden oder Umleiten der Nutzer Ein- und Ausgaben (Verteilung auf Ebene der Interaktoren) ändern die UI-Verteilung.
2) **Zustandstransfer:** Wird der UI-Zustand vor einer Änderung der Verteilung bestimmt und entsprechend der neuen Verteilung auf die Zielgeräte transferiert kann die Nutzungskontinuität hergestellt werden. In [1] wird dies als der migratorische Aspekt eines UIs bezeichnet.
3) **Adaption:** Eine UI Transfer in eine andere Laufzeitumgebung kann eine Anpassung des UIs an das Zielgerät erforderlich machen. Angepasst werden kann sowohl die Darstellung, aber auch der transferierte Zustand. Der Aspekt der Adaption wird auch als UI Plastizität bezeichnet [1].

### 2.2 Klassifikationsmerkmale

Die von einem MUI bereitgestellten Funktionen können nach [1]-[3] anhand von 13 Merkmalen klassifiziert werden. Einige, für die Automatisierungstechnik relevante, Merkmale werden nachfolgend vorgestellt:

1) **Initiierung** Eine Migration kann entweder durch Benutzer oder automatisch durch das UI ausgelöst werden. Wird der Vorgang vom Zielgerät aus initiiert so muss das UI vom Quellgerät oder einem anderen Host geholt werden. Im Gegensatz dazu wird das UI verschickt, wenn die Initiierung auf Quellseite stattfindet. Mischformen sind ebenfalls möglich, zum Beispiel indem Inhalte in beiden Richtungen ausgetauscht werden.
2) **Anwendungsbereich** Das UI kann total – als Ganzes migriert werden, aber auch in Teilen – partiell, wobei letzteres stets zu einer Änderung der UI Verteilung führt (siehe Abschnitt 2.1). In Fällen verteilter UIs kann sich dabei die Anzahl der Geräte ändern, was entweder zu einer Verteilung auf eine größere oder einer Aggregation auf weniger Geräte führt. Zu Mischformen kommt es dann, wenn Aggregation und Verteilung gleichzeitig auf verschiedene Inhalte angewendet werden.
3) **Adaptionstyp** Der Adaptionstyp beschreibt, wie die Adaption des UIs an das Zielgerät erfolgt und das UI somit im Sinne der kontinuierlichen Benutzbarkeit funktionsfähig wird. Dies kann dynamisch zur Laufzeit erfolgen, indem eine Anpassung an das Zielgerät automatisch erfolgt. Üblicherweise sind dafür mehr oder weniger aufwändige Algorithmen erforderlich. Eine andere Variante ist, UIs für potentielle Zielgeräte bereits in der Entwurfsphase zu erstellen und zum Aufruf bzw. Download zu speichern. In [2] wird dies als ein voreingestelltes Verfahren bzw. statische Adaption bezeichnet. Durch die Nutzung sogenannter Templates, die die Struktur, aber nicht den Inhalt vordefinieren, können Zwischenformen realisiert werden.
4) **Simultanbenutzbarkeit** In Szenarien mit mehreren Geräten oder Benutzern können mehrere Benutzer ein Gerät verwenden (N-1), ein Benutzer mehrere Geräte (1-N) oder mehrere Benutzer mehrere Geräte verwenden (N-N). Zur Simultanbenutzung eines (verteilten) UIs durch mehrere Benutzer kommt es dann, wenn durch die Benutzung nicht nur zeitlich, sondern auch inhaltlich gleiche Aufgaben bearbeitet werden (bspw. in einem kollaborativen Szenario). Im Kontext migratorischer und verteilter UIs beinhaltet der Begriff der Simultanbenutzbarkeit auch die Art und Weise des Zustandstransfers. Dieser kann kontinuierlich mehrere UIs miteinander synchronisieren oder nur zu einem bestimmten Zeitpunkt (z.B. zum Zeitpunkt der Initiierung einer Migration).

## 3. Anforderungen industrieller Visualisierungen an ein migratorisches UI

### 3.1 Anforderungen an Software und Engineering

Industrielle Visualisierungslösungen müssen in der Regel mit komplexen und teuren technischen Systemen interagieren, was besondere Anforderungen an deren Sicherheit und Zuverlässigkeit mit sich bringt, insbesondere für die Benutzungsschnittstellen. Auch aufgrund der

Bindung des Engineerings an die Anforderungsdefinition in einem Lastenheft, bis hin zur Haftbarmachung [15], wird in der Entwurfsphase bereits eine strenge Vorhersagbarkeit des Verhaltens und der Qualität der Darstellung zur Laufzeit des UIs verlangt. Eigenschaften der Darstellung und der Funktionalität werden in Standards und Richtlinien der jeweiligen Branche und Normen festgelegt. Im Sinne der Gebrauchstauglichkeit ist die Erwartungskonformität ein wichtiges Qualitätsmerkmal. Dies äußert sich unter anderem dadurch, dass Benutzer durch das UI nicht an der Ausführung Ihrer Aufgabe gehindert oder durch nicht erwartungskonformes Verhalten davon abgelenkt werden dürfen [4].

### 3.2 UI Struktur und Funktionalität

Industrielle Visualisierungen bestehen gewöhnlich aus einer Reihe von Bedienpaneelen, die über Navigatorelemente miteinander verbunden werden. Auf den Paneelen sind Elemente zur Darstellung und Manipulation des zu steuernden Prozesses enthalten [5]. Solche Elemente sind durch korrespondierende Elemente des zugrunde liegenden Prozesses funktional miteinander assoziiert. Das hat zur Folge, dass solche Elemente, beispielsweise bei einer partiellen Migration, nicht beliebig voneinander getrennt werden dürfen. Informationen über solche Zusammenhänge sind aber in der Regel nicht Bestandteil der ursprünglichen Visualisierung auf dem Bediengerät, sondern werden im Entwurfsprozess von Domänenexperten informal bereitgestellt und fließen so in deren Entwurf ein. Das erforderliche Wissen liegt somit weder innerhalb noch außerhalb der Visualisierungslösung in einer maschineninterpretierbaren und somit verwendbaren Form vor. Nur wenn dieses Wissen formalisiert bereitgestellt wird, kann sichergestellt werden, dass alle für eine Arbeitsaufgabe notwendigen Bestandteile des UIs zusammen migriert werden. Dies ist bspw. in [6] der Fall, wo aus den migrierbaren Elementen eines UIs automatisch diejenigen ermittelt werden, die für die aktuelle Aufgabe gebraucht werden. Anhand dessen wird ebenso automatisch das passende Ziel-UI aus einer Reihe vorgefertigter UIs ausgewählt. In jedem Fall werden in der Automatisierungstechnik jedoch Eingriffsmöglichkeiten zur Anpassung der Inhalte, Verteilung und Adaption bei der Durchführung einer Migration benötigt, um die Anforderung der Vorhersagbarkeit nach Abschnitt 3.1 zu erfüllen.

Der Entwurf eines UIs ist meist streng an einen bestimmten Anwendungsfall, an die vorgesehene Rolle der Benutzerklasse, das jeweilige Bediengerät und dessen Umgebung gebunden [7] – mit anderen Worten an den Benutzungskontext. Bei der Nutzung verschiedenartiger, z.B. stationärer oder mobiler Endgeräte zur Anzeige eines UIs ist es daher nicht ungewöhnlich, dass für jedes Endgerät eigene, speziell entwickelte Lösungen erstellt werden. Nach Abschnitt 2.2 entspricht dies der Nutzung eines voreingestellten Adaptionstyps.

Wesentlich ist, dass Visualisierungslösungen über Schnittstellen mit Prozessdatenservern und/oder historischen Datenbanken kommunizieren, um stets eine aktualisierte Ansicht des Prozesses zu realisieren. Allerdings hat nicht jeder Benutzer gleiche Bedien- oder Beobachtungsrechte. Autorisierungsunktionen können Zugriffsrechte gegenüber den Servern oder auf spezielle Bedienfunktionen einschränken. Die Prozessdaten- oder Datenbankserver definieren einen wesentlichen Teil des UI Zustands. Da sich jedoch aufgrund verschiedener Benutzer Schreib- und Leserechte auf unterschiedlichen Geräten unterscheiden können, darf dieser Anteil am UI Zustand im Falle einer Migration nicht automatisch übernommen werden. Häufig werden aus Sicherheitsgründen bei verteilten UIs Bedienrechte zu einem Zeitpunkt nur genau einem Benutzer erlaubt.

### 3.3 Diskussion

Zieht man die Eigenschaften und Anforderungen industrieller UIs in Betracht können mögliche Eigenschaften der Migrationsfunktion diskutiert werden:

1) **Initiierung** In Fällen von automatisch initiierten Migrationen können Benutzer der jeweiligen Quell- oder Zielgeräte massiv beeinträchtigt werden, sei es durch wiederholt eingehende Migrationen, Anforderungen Dritter zu ausgehenden Migrationen oder aber auch durch Änderungen der Berechtigungen bei verteilten UIs. Die in Abschnitt 3.1 genannten Anforderungen an die Erwartungskonformität erfordern weiterhin ein Regelwerk bzgl. zwingend zusammengehöriger Elemente. Diese Nachteile entfallen bei einer manuell initiierten Migration per Ausnutzung der ohnehin bei den initiierenden Benutzern verfügbaren situativen Wahrnehmung.

   Um die Vorteile einer manuell initiierten Migration ausnutzen zu können, dabei aber die Arbeitsaufgabe der Zielbenutzer nicht zu unterbrechen, erscheint ein hybrider Ansatz bei der Vermittlung einer Migration zwischen Quell- und Zielseite sinnvoll. Diese sähe vor, dass ein Quellbenutzer zunächst einen Zielbenutzer über eine Migration informiert und sich dieser die Migration anschließend aktiv holen muss. Im entgegengesetzten Fall sollte ein Quellbenutzer zunächst – abhängig von den jeweiligen Berechtigungsstufen – nach seinem Einverständnis gefragt werden, wenn die Initiierung durch den Zielbenutzer erfolgte.
2) **Anwendungsbereich** Ob ein UI Element migriert werden darf, hängt von den Berechtigungen der beteiligten Benutzer ab. Falls nicht der Zugang zu Bestandteilen des UIs beschränkt wird, sondern lediglich die Sicht- und Schreibbarkeit der Datenpunkte auf dem Prozessdatenserver im Hintergrund, resultiert dies in Visualisierungen, die nur unvollständig durch Prozesswerte aktualisiert werden oder mit denen bestimmte Funktionen nicht

aufgerufen werden können. Wenn dies ein Sicherheitsrisiko darstellt oder gegen die Gestaltungsregeln verstoßen (Teile des UIs sind nicht funktional, ohne dass der Benutzer erkennt, dass dies an fehlenden Zugriffsrechten liegt), dürfen Migrationen nicht total ausgeführt werden, sondern müssen sich von vornherein auf die Elemente beschränken, die auch auf dem Zielgerät funktionieren.

3) **Adaptionstyp** Automatische Adaptionen zur Laufzeit erfordern ein getestetes und zuverlässig funktionierendes Regelwerk, da fehlerhafte oder nicht erwartungskonforme Darstellungen ein erhebliches Sicherheitsrisiko darstellen. Eine Formalisierung des erforderlichen Anlagen- und Prozessverständnisses (vgl. Abschnitt 3.2) ist dafür Voraussetzung. Der jeweilige Aufwand und die Risiken sind im Einzelfall zu überprüfen.
   Verglichen mit den Risiken automatischer Adaptionen, erscheint die Nutzung statisch angepasster UIs sinnvoll. Jedoch ist der Aufwand passende UIs für alle in Frage kommenden Zielgeräte und Benutzerrollen durch Domänenexperten zu entwickeln unter Umständen erheblich. Die Entwicklungskosten hängen in diesem Fall aber von der Anzahl der zu unterstützenden Bediengeräte und damit von der Anzahl der zu entwickelnden UIs ab. Im Gegensatz zur Laufzeitadaption werden bei der statischen Adaption keine Inhalte transferiert (vgl. Abschnitt 2.1), sondern lediglich auf Zielseite aufgerufen. Da allerdings bei individuell entwickelten UIs die Inhalte nicht zwingend identisch sind (Struktur, Inhalt und Darstellung können sich unterscheiden), müssen Elemente im Quell- und im Ziel-UI als korrespondierend gekennzeichnet (verlinkt) werden, um zu einer Anzahl an migrierten Elementen die im Ziel-UI aufzurufenden Inhalte zu definieren und die Ziele des Zustandstransfers festzulegen.
4) **Simultanbenutzbarkeit** Wenn die Zustände von Quell- und Ziel-UI fortlaufend synchronisiert werden, führt das im Sinne der kontinuierlichen Benutzbarkeit zu den besten Ergebnissen, da Benutzer ihre Geräte ohne Zeitverzug wechseln können, solange mindestens einmal eine Migration initiiert wurde. In Bezug auf die Sicherheit und Zuverlässigkeit einer Anlage kann eine solche Synchronisationsfunktion Vorteile bringen, wenn dadurch ein Vier-Augen-Prinzip oder mindestens verbesserte Kommunikationsmöglichkeiten realisiert werden, auch ohne dass sich das beteiligte Personal am selben Standort befinden muss. Allerdings muss die Synchronisation sowohl im Entwurfsprozess, als auch zur Laufzeit durch ihre Benutzer konfigurierbar sein. Dies bedeutet, dass beispielsweise die Richtung der Synchronisation steuerbar sein muss, sodass ein Benutzer mit Bedienrechten anderen Benutzern das Beobachten des eigenen UIs ermöglichen kann, aber nicht das Bedienen.

Zusammenfassend ergibt sich daraus, dass eine Ergänzung einer industriellen Visualisierungslösung durch eine Migrationsfunktion, folgende zusätzliche Informationen benötigt: 1) funktionale Abhängigkeiten von UI Elementen, 2) Kennzeichnung des für eine Migration zu berücksichtigenden UI Zustands und 3) Konfiguration der Synchronisationsfunktion. Die Vorhersagbarkeit des UI Verhaltens erfordert ein zuverlässiges und überschaubares Regelwerk um diese Informationen zur Laufzeit zu ermitteln oder eine geeignete Datenstruktur, um die Konfiguration zur Entwurfszeit vorausentwickeln zu können.

## 4. Beispielhafte Anwendungsszenarien

Zur Durchführung einer Fallstudie wurden folgende mögliche Szenarien identifiziert:

1) **Dynamischer Wechsel zwischen Ein- und Ausgabegeräten**
   Bei Wartungs- und Inspektionsrundgängen werden häufig Bar- oder QR-Codes verwendet, um die Sichtung eines Automatisierungsgeräts per Einscannen des Codes z.B. auf einem mobilen Gerät zu dokumentieren. Aus speziellen baulichen Gegebenheiten der Anlage kann ein unabhängiger Handscanner erforderlich sein. Ist dies nicht möglich oder nicht erwünscht, so kann anstelle des Handscanners auch die integrierte Kamera des mobilen Geräts genutzt werden. Änderungen während des Wartungsrundgangs entsprechen einer Migration der Scanfunktion vom Handscanner auf die Kamera des mobilen Geräts. Hinsichtlich der Einordnung (vgl. Abschnitt 2.1) erfolgt diese Migration auf Interaktorenebene. Ein ähnlicher Anwendungsfall liegt vor, wenn eine Maschine bei Ausfall eines Bedienpanels vor Ort automatisch die Aus- und Eingabe auf ein externes Gerät umleitet, wie z. B. einem Tablet-PC, damit sie arbeitsfähig bleibt.
2) **Migration zwischen Bediengeräten zur mobilen Inbetriebnahme und Wartung**
   Bei Wartungs- und Reparaturarbeiten an einer räumlich weit verteilten technischen Anlage kann ein Operator in der Leitwarte dem eintreffenden Wartungspersonal per Mausklick einen Ausschnitt aus dem UI schicken. Auf dem Zielgerät kann sich ein von der Leitwarte verschiedenes und an die Wartungsaufgabe angepasstes UI öffnen. Mit der Migration geht eine automatische Bedienfreigabe für den betreffenden Teil der Anlage einher, sodass Wartungstechniker vor Ort zunächst über die alleinigen Bedienrechte verfügen. Über eine Synchronisationsfunktion kann der Operator der Leitwarte den Fortschritt der Reparatur überwachen und bei einem sich einstellenden Sicherheitsrisiko oder einer Fehlbedienung reagieren. Die beiden UIs können jedoch auch unabhängig voneinander als Teilnehmer im Leitsystem agieren, sodass das Unterbrechen oder die Wiederaufnahme der gegenseitigen Synchronisation jederzeit möglich ist oder bei der Wartung vor Ort Bedienbilder aufrufbar sind, die in der Leitwarte nicht installiert sind.

Die Migrationsfunktion in diesem Szenario könnte durchaus ebenfalls auf Interaktorenebene realisiert werden, jedoch ist es üblich, dass die Visualisierungslösungen auf den jeweiligen Endgeräten (PC in der Leitwarte oder mobiles Gerät im Feld) installiert werden und sich unabhängig von anderen Endgeräten selbst verwalten. Damit muss eine solche Migrations- und Synchronisationsfunktion auf Anwendungsebene funktionieren.

## 5. Fallstudie

### 5.1 Existierende Prototypen

In der Literatur ist bereits eine Anzahl an Prototypen beschrieben, die jeweils einen Teil der in Abschnitt 2 genannten Eigenschaften realisieren. So zum Beispiel die „Multiaccess Service Platform“ (MASP) aus [12], die sich in einem Umfeld der Heimautomatisierung damit beschäftigt die Ein- und Ausgaben einer Anwendung dynamisch auf Ebene der Interaktoren zwischen verschiedenen Geräten zu verteilen. Damit ließen sich Teilaspekte des ersten Szenarios in Abschnitt 4 realisieren. Allerdings wird in der Literatur nicht erläutert, wie die Adaptionsmechanismen funktionieren [13], oder ob sich auch eine Verteilung auf Anwendungsebene realisieren lässt. Auf Anwendungsebene sind andere Konzepte anzusiedeln, wie die „Dynamische Migration beliebiger Webinhalte“ [14], „TERESA-basierte Anwendungen“ [6] oder das „DireWolf-Framework“ [11]. In diesen Frameworks kommen u. A. verschiedene Algorithmen zum Einsatz, um die Migration und Adaption zu steuern, jedoch sind diese nicht formalisiert oder in der Entwurfsphase anpassbar, wie in Abschnitt 3.3 gefordert. Außerdem werden entweder unterschiedliche Zugriffsrechte der Benutzer nicht berücksichtigt oder es wird keine Unterscheidung bezüglich der relevanten und nicht relevanten Teile des UI Zustands vorgenommen, wie sie durch die Einbeziehung von Prozessdatenservern notwendig wird (vgl. Abschnitt 3.2).

### 5.2 Konzeptionierung einer Migrationslösung

Unserer Fallstudie setzt Szenario zwei aus Abschnitt 4 um. Genutzt werden dabei mehrere vorgefertigte sowie an Gerät und Arbeitsaufgabe angepasste UIs (statische Adaption – vgl. Abschnitt 3.3-3). Damit soll die Erwartungskonformität jeder einzelnen Benutzungsschnittstelle sichergestellt werden. Ein Benutzer in der Leitwarte soll in der Lage sein, manuell ausgewählte Bestandteile seines UIs (nutzerinitiierte, partielle Migration – vgl. Abschnitt 2.2-1) an das Wartungspersonal im Feld zu senden. Damit stehen vor Ort, aufgrund des Funktionsumfangs oder der Anpassung an das Gerät, mehrere UIs für die Durchführung der Wartungsaufgabe zur Verfügung. Somit werden bei einer Migration nach Abschnitt 3.3-3 explizite Informationen benötigt, wo in den möglichen Ziel-UIs die migrierten Elemente zu finden sind. Für diese Verlin-

kungen, aber auch zur Konfiguration einer Zustandssynchronisation (vgl. Abschnitt 3.3-4) haben wir ein Migrationsmodell entwickelt, welches im Entwurfsstadium zu erstellen ist. Damit wird Anforderung der Vorhersagbarkeit nach Abschnitt 3.1 erfüllt. Da nicht sichergestellt ist, dass in jedem UI zwingend alle migrierten Elemente enthalten sind, wird ein Auswahlmechanismus auf dem Zielgerät integriert, mit dem der Zielbenutzer das gewünschte UI aufrufen kann. Unter Nutzung des Migrationsmodells wird gekennzeichnet, welches UI wie viele der migrierten Elemente enthält.

Abbildung 1 zeigt die Bestandteile der Laufzeitumgebung, die im Wesentlichen aus einem Migrationsserver und den auf den Endgeräten dargestellten UIs besteht. Diese existierenden UIs wurden zur Realisierung der Migrationsfunktion durch ein Plug-In erweitert, um beispielsweise Zugang zum UI-Zustand zu erlangen. Die Client-Komponente, stellt dabei technologieunabhängige Funktionen bereit, wie das Initiieren, aber auch das Empfangen und Öffnen einer Migration und kommuniziert dazu mit Server und Plug-In. Der Migrationsserver hält das Migrationsmodell bereit und implementiert eine einfache Zugriffsbeschränkung um die Anforderungen an die Benutzerautorisation (vgl. Abschnitt 3.2) umzusetzen.

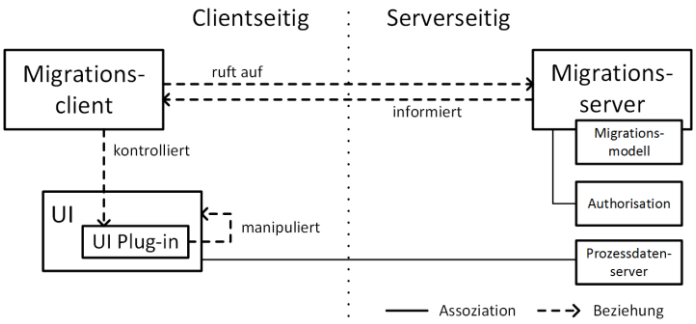


Bild 1: Aufbau unserer Laufzeitumgebung für die Erweiterung von Visualisierungslösungen mit einer Migrationsfunktion

In Abbildung 2 ist ein Bildschirmfoto unserer Migrationsprototypen zu sehen, der im oberen Bildteil die Steuerelemente für die Migration bereitstellt (Auswahl der Zielbenutzer und –Geräte, Migrationsbutton – „B“) sowie im Hauptteil unten rechts die migrierbare Prozessvisualisierung. Das Plug-In ermöglicht das grafische Hervorheben der migrierbaren Elemente, die dann durch den Benutzer zur Migration ausgewählt werden können – hier die mit „A“ gekennzeichneten Elemente. Bild 3 zeigt, dass sich nun auf dem Zielgerät eine eingehende Migration in Form eines Dialogs öffnen lässt. Dieser Dialog ermöglicht es, unter allen verfügbaren UIs das gewünschte auszuwählen. Die Anzeige der verfügbaren migrierten Elemente erleichtert die Auswahl. Bild 4 zeigt schließlich ein geöffnetes Ziel-UI, in diesem Fall eine modifizierte Version des Quell-UIs. Im Navigationsteil links können die verfügbaren Bedienseiten dieses UIs aufgerufen werden, um die migrierten Elemente innerhalb des geöffneten UIs anzuzeigen.

Eine Synchronisationsfunktion lässt sich in diesem Beispiel so konfigurieren, dass auf Quell- und Ziel-UI stets die miteinander korrespondierenden Bedienseiten geöffnet werden oder bspw. der in einem Trendgraphen gewählte Ausschnitt. Die Zugriffsrechte wurden hier auf ganze UIs beschränkt, sodass einzelne im Auswahldialog nicht zur Verfügung stehen, wenn der angemeldete Benutzer kein Zugriffsrecht hat.

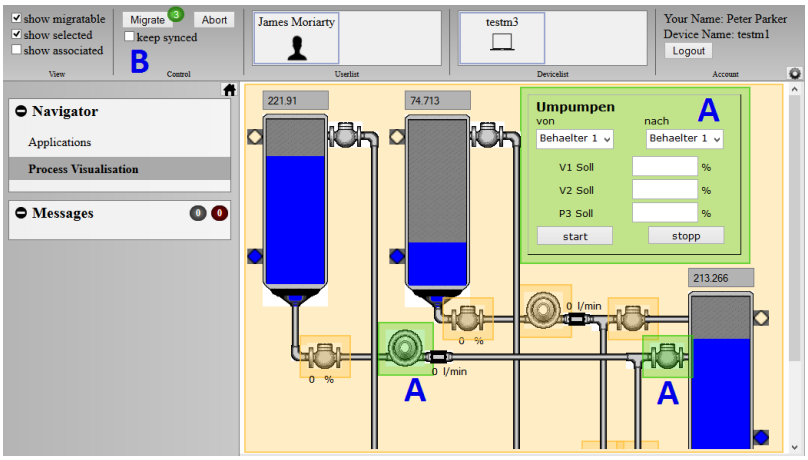


Bild 2 (links): Prozessvisualisierung mit Migrationsclient

Bild 3 (rechts): Geöffneter Dialog für eingehende Migrationen zur Auswahl des Ziel-UIs

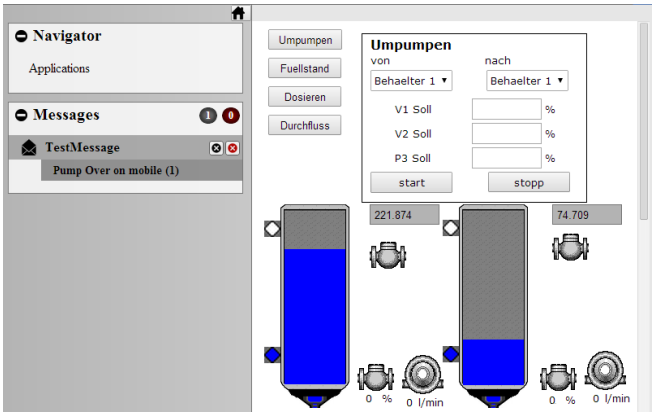


Bild 4: Geöffnetes Ziel UI mit modifizierter Anordnung grafischer Elemente

## 6. Zusammenfassung und Ausblick

Dieser Beitrag erläutert die wesentlichen Grundlagen migratorischer Benutzungsschnittstellen und diskutiert diese anhand der Anforderungen industrieller Prozessvisualisierungen. Anhand unserer Fallstudie konnten wir zeigen, dass existierende Visualisierungen mit einer Migrationsfunktion erweitert werden können. Jedoch werden dafür geeignete Schnittstellen der UI Laufzeitumgebung benötigt. Als sinnvoll hat sich die Systemarchitektur mit einem separaten Migrationsserver erwiesen, da so die Echtzeitanwendungen auf den Prozessdatenservern

nicht beeinträchtigt werden. Des Weiteren kann der Migrationsserver sicherstellen, dass Migrationen auch dann noch auf das Zielgerät verschickt werden können, wenn Quell- und Zielgerät nicht gleichzeitig aktiv sind. Ein Benutzer auf der Zielseite wird durch den Migrationsmechanismus nicht beeinträchtigt, da eingehende Migrationen nicht sofort aktiv werden, sondern zunächst in Form von Nachricht erscheinen. Diese müssen erst aktiv geöffnet werden. Migriert ein Benutzer hingegen das UI auf ein eigenes Gerät, öffnet sich der Auswahldialog sofort, was die Unterbrechung der Benutzung minimiert. Das Migrationsmodell kann Informationen formalisiert aufnehmen, die im Entwurfsprozess der UIs gewöhnlich lediglich als Expertenwissen vorliegen und zur Laufzeit nicht mehr zur Verfügung stehen. So kann zum Beispiel selbst die manuelle Auswahl an zu migrierenden Elementen optimiert werden, indem im Migrationsmodell gemeinsam zu migrierende Elemente gekennzeichnet werden. Jedoch bedeutet das Erstellen und die Pflege dieser Modelle gerade bei größeren Projekten einen erheblichen Mehraufwand, sodass dafür in Zukunft eine Werkzeugunterstützung erforderlich ist. Weitere praktische Erfahrungen mit Migrationslösungen in industriellen Umgebungen liegen derzeit noch nicht vor, da der Prototyp noch weiterentwickelt und ausführlicher getestet werden muss.
Obgleich in diesem Beitrag nur Laufzeitmigrationen (vgl. [1]) betrachtet und in der Fallstudie realisiert wurden, lassen sich ähnliche Mechanismen auf Migrationen zur Entwurfszeit anwenden. So werden beispielsweise bei der Realisierung von dynamischen Adaptionen Mechanismen benötigt, die aus den migrierten Elementen Bedienseiten innerhalb des Ziel-UIs erzeugen. Dies ähnelt einem Anwendungsfall, in dem eine Komponente eines Plug & Produce-Szenarios sein eigenes Bedienbild mitliefert, welches nach Einbau der Komponente automatisch in die bestehende Visualisierungslösung integriert wird (zur Entwurfszeit, aber auch zur Laufzeit). Wenn die Migrations- und Adaptionsfunktion in der Lage ist, sich auch an die auf den jeweiligen Geräten verwendete Visualisierungstechnologie anzupassen, dann lassen sich damit unter Umständen auch komplette Visualisierungen migrieren. Dies könnte beispielsweise beim Wechsel des Prozessleitsystems verwendet werden.

### Literaturstellen


[1] L. Balme, A. Demeure, N. Barralon, J. Coutaz, and G. Calvary, “Cameleon-rt: A software architecture reference model for distributed, migratable, and plastic user interfaces,” in Ambient Intelligence, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3295.

[2] S. Berti, F. Patern ` o, and C. Santoro, “A taxonomy for migratory user interfaces,” in Interactive Systems. Design, Specification, and Verification, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 3941.

[3] F. Paternò and C. Santoro, “A logical framework for multi-device user interfaces,” in Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems, ser. EICS ’12. ACM, 2012.

[4] Ergonomics of human-system interaction - Part 110: Dialogue principles, DIN EN ISO Std. 9241-110, September 2008.

[5] Process control using display screens - Part 3: Mimics, VDI/VDE Std. 3699-3, January 2014.

[6] R. Bandelloni and F. Paternò, “Flexible interface migration,” in Proceedings of the 9th international conference on Intelligent user interfaces, ser. IUI ’04. ACM, 2004.

[7] Development of usable user interfaces for technical plants - Part 1: Concepts, principles and fundamental recommendations, VDI/VDE Std. 3850-1, April 2014.

[8] Building automation and control systems - Part 7: Design of user interfaces, VDI Std. 3814-7, Mai 2012.

[9] Blumendorf, Marco, Dirk Roscher und Sahin Albayrak (2010). „Dynamic user interface distribution for flexible multimodal interaction“. In: International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction.

[10] Lachenal, Christophe und Joëlle Coutaz (2003). „A reference framework for multisurface interaction“. In: Proc. HCI International

[11] Kovachev, Dejan u. a. (2013). „DireWolf - Distributing and Migrating User Interfaces for Widget-Based Web Applications“. In: Web Engineering. Bd. 7977. LectureNotes in Computer Science. Springer Berlin Heidelberg.

[12] Blumendorf, Marco, Dirk Roscher und Sahin Albayrak (2010). „Dynamic user interface distribution for flexible multimodal interaction“. In: International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction. ICMI-MLMI ’10. ACM.

[13] Paternò, Fabio, Carmen Santoro und Lucio Davide Spano (2009). „MARIA: A Universal, Declarative, Multiple Abstraction-level Language for Service-oriented Applications in Ubiquitous Environments“. In: ACM Trans. Comput.-Hum. Interact. 16.

[14] G. Ghiani, F. Patern ` o, and C. Santoro, “On-demand cross-device interface components migration,” in Proceedings of the 12th international conference on Human computer interaction with mobile devices and services, ser. MobileHCI ’10. ACM, 2010.

[15] IHK, „Muster eines Softwareerstellungsvertrages“ Jan. 2014, http://www.frankfurt-main.ihk.de/recht/mustervertrag/software_erstellung/