Entwicklung eines UI-Teilmodells für die Industrie 4.0 Komponente

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

Kurzfassung

Die Bereitstellung von Benutzungsschnittstellen (UI) per Industrie 4.0 Komponenten und ihrer Verwaltungsschalen (AAS) setzt den Entwurf eines neuen UI-Teilmodells voraus. In Abhängigkeit des geplanten Anwendungsfalls und des Nutzungskontexts muss das Teilmodell in der Lage sein mehrere UI-Fragmente (Varianten des UIs) bereitzustellen, die in UI-Lösungen eingefügt werden können. Dafür ist die Identifikation relevanter Merkmale und die Spezifikation einer Teilmodellstruktur nötig, um die Fragmente beschreiben und später das richtige auswählen zu können. Für den Einsatz des entworfenen Teilmodells in einem Plug-and-Produce-Szenario, müssen diese Merkmale formal spezifiziert sein, um durch automatische Werkzeuge korrekt interpretiert werden zu können. In diesem Beitrag werden der Entwurf eines UI-Teilmodells, ein Katalog von UI-Fragment-Merkmalen sowie eine erste Fallstudie zu deren Anwendung in einem industriellen Szenario vorgestellt.

1. Einleitung

Die Umsetzung des *Plug-and-Produce* Szenarios (PnP) ist ein wesentlicher Bestandteil der Initiative *Industrie 4.0* (I4.0). Das Szenario behandelt dabei eine möglichst automatische Anpassung der gesamten automatisierungstechnischen Einrichtungen auf unvorhersehbare Änderungen an einer Produktionsanlage[1]. Darunter können einfache Änderungen verstanden werden, wie der Austausch eines defekten Geräts, aber auch komplexe, wie das Hinzufügen oder Entfernen ganzer Teilanlagen oder der Komplettaufbau einer ganzen Anlage. Als Informationsquelle für die automatischen Anpassungen des Prozessführungs- bzw. Steuerungssystems sollen *I4.0 Komponenten* dienen[2], die aus einer *Verwaltungsschale* (AAS – Asset Administration Shell) und einem durch die AAS verwalteten *Asset* bestehen. Als Asset kann dabei jeder virtuelle oder reale Gegenstand von Wert für eine Organisation verstanden werden[3], z.B. die jeweiligen Prozesskomponenten. In der AAS werden in *Teilmodellen* alle zur Integration des Assets nötigen Informationen über das bereitgestellt, aber auch Dokumente, wie z.B. Quellcode oder CAD-Modelle. Weiterhin stellt die I4.0-Komponente Dienste zum Zugriff auf das Asset oder die Teilmodelle bereit.

Bestandteil des Automatisierungssystems einer Anlage sind neben der Steuerung an sich auch stets eine oder mehrere *Benutzungsschnittstellen* (UI – User Interface), mit deren Hilfe

menschliches Personal mit der Anlage oder Teilen davon interagieren kann. I.d.R. stellen solche Bediensysteme komplexe Teilsysteme der Anlage dar, die in Abhängigkeit der Anlagenkomplexität sowie der Arbeitsaufgaben des Personals aufwändig gestaltet werden müssen. Inhaltlich stellen sie die Anlage so dar, dass einzelne Repräsentationen an die Erfordernisse der Arbeitsaufgaben angepasst sind. Dabei stehen einzelne Komponenten der Anlage im Vordergrund oder auch ganze Strukturen, bestehend aus mehreren Komponenten. Die Anlagenstruktur (Hierarchie und Topologie) spiegelt sich darüber hinaus häufig in der Navigationsstruktur der *UI-Lösung* wieder. Neben dem Inhalt wird bei der Gestaltung des UIs aber auch der *Nutzungskontext* berücksichtigt[4], der die Beschreibung der das UI nutzenden Personen, der UI-Plattformen (die Interaktionsgeräte) sowie der Umgebung (physisch, technisch, organisationsbezogen, kulturell, gesellschaftlich[5], [6]) umfasst.

Im Rahmen eines PnP-Szenarios gehören die UI-Lösungen zu den anzupassenden Einrichtungen einer Anlage, da sich mit der Anlage auch Arbeitsabläufe, Umgebungsbedingungen, Organisationsstrukturen oder auch die UI-Plattformen ändern können. Wenn eine UI-Lösung automatisch an solche Änderungen angepasst werden kann, kann sie als PnP-fähig bezeichnet werden. Ein besonderer Fall ist die Inbetriebnahme neuer Anlagenbestandteile, da hierbei die korrespondierenden UI-Bestandteile – im Folgenden als *UI-Fragmente* bezeichnet – *mit* dem Einbau der Komponenten bereitgestellt werden müssen (z.B. mittels der I4.0-Komponenten der neuen Anlagenbestandteile), um sie in eine UI-Lösung integrieren zu können. Im Rahmen der AAS-basierten Distribution dieser UI-Fragmente ist somit die Entwicklung eines neuen *UI-Teilmodells* nötig, da in der derzeit verfügbaren Literatur zur Verwaltungsschale[7]–[9] noch keines vorgeschlagen wurde.

Beim Teilmodell-Entwurf sind hierbei die UI-spezifischen Besonderheiten zu berücksichtigen: So ist es nötig pro Komponente nicht nur ein UI-Fragment bereitzustellen, sondern entsprechend der hohen Variabilität an Anforderungen an verschiedene UI-Lösungen ggf. mehrere *Varianten* der UI-Repräsentation, d.h. mehrere UI-Fragmente. Um beim Erstellen einer UI-Lösung auf Basis der Fragmente ein geeignetes Fragment auswählen zu können, müssen diese im Teilmodell durch eindeutig interpretierbare Merkmale ausgezeichnet werden. Im Kontext PnP-fähiger UIs muss eine solche Auswahl durch automatische Werkzeuge erfolgen. Ziel dieses Beitrags ist daher die Entwicklung eines neuen UI-Teilmodells für die Verwaltungsschale, mit dessen Hilfe UI-Fragmente in Form ihrer Implementierung bereitgestellt und jeweils durch aussagefähige Merkmale beschrieben werden können. Das Teilmodell soll hierbei allerdings nicht die eigentliche Implementierung der UI-Fragmente unter Nutzung einer beliebigen UI-Technologie ersetzen, sondern die UI-Fragment-Implementierungen um auswahlrelevante Merkmale ergänzen.

Im Rahmen dieses Beitrags behandeln wir in Kapitel 2 die Anforderungen an und daraus hervorgehende Merkmale für UI-Fragmente. In Kapitel 3 wird kurz der aktuelle Stand der Technik dargestellt, um anschließend in Kapitel 4 eine geeignetes UI-Teilmodell zu entwerfen. In Kapitel 5 stellen wir eine erste Fallstudie vor.

2. Anforderungen und Identifikation relevanter Merkmale

2.1. Akteure und Anforderungen an UI-Fragment-Merkmale

An der Entwicklung und Verwendung von UI-Fragmenten sind mehrere Akteure beteiligt, siehe Abb. 1. Aus deren jeweiligen Aufgaben ergeben sich Anforderungen an das UI-Teilmodell und darin verwendete Merkmale, wobei im Rahmen dieses Beitrags lediglich explizit die Auswahl von UI-Fragmenten zur Erstellung einer UI-Lösung betrachtet wird.

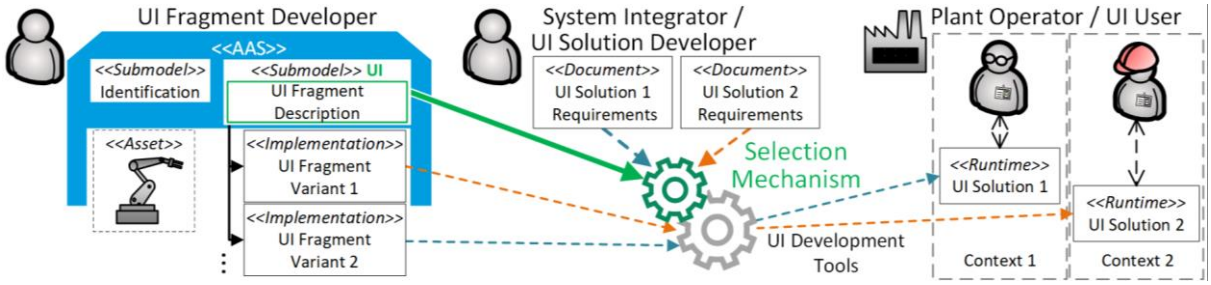


Abbildung 1: UI-Stakeholder und ihre Beziehungen zu einem Fragment-basierten UI-Entwurf

Das Personal eines *Anlagenbetreibers* interagiert durch die UI-Lösung mit der Anlage. Daher muss die UI-Lösung im Sinne der Erstellung gebrauchstauglicher Benutzungsschnittstellen[12] für den jeweiligen *Anwendungsfall* geeignet sein, was die konkreten *Arbeitsaufgaben*[4] (z.B. die Inbetriebnahme einer Komponente) sowie die auftretenden *Nutzungskontexte* (z.B. die Inbetriebnahme erfolgt im Feld unter Verwendung bestimmter UI-Plattformen) beinhaltet. Weiterhin müssen die individuellen Rollen der Nutzenden innerhalb der Organisation (bspw. die Berechtigungsstufe), individuelle Nutzereigenschaften, wie das Nutzerwissen (z.B. domänenspezifisch in Bezug auf den Aufgabengegenstand), die Erfahrung bei der Aufgabenbewältigung oder die Kenntnis der geänderten Anlagenkonfiguration berücksichtigt werden. Solche Faktoren beeinflussen die zu erzeugende UI-Lösung sowohl inhaltlich (Gegenstand und Umfang sind angemessen reduziert, ermöglichen aber die Aufgabenabarbeitung), als auch gestalterisch (Symbolik, Ein- und Ausgaben können verstanden werden). Zudem ist bei der Zusammensetzung einer UI-Lösung aus UI-Fragmenten auf gestalterische *Konsistenz* zu achten (Verwendung einheitlicher Symbolik, Farbgebung, Anordnung, Nutzerfeedback usw.), um die Gebrauchstauglichkeit sicherzustellen[13], [14].

UI-Fragment-Entwickler sind für Entwurf und Implementierung der UI-Fragmente verantwortlich, wobei Komponenten beliebiger Komplexität darzustellen sind. Festzulegen sind die nö-

tigen Arbeitsaufgaben der Nutzenden, der jeweils darzustellende Inhalt (insbesondere komplexe Komponenten erfordern häufig eine Reduktion der Betrachtungseinheit) usw. Auf Basis dessen werden die UI-Teilmodelle erstellt sowie die Implementierung der UI-Fragmente inkl. deren Beschreibung im Teilmodell unter Verwendung aussagefähiger Merkmale. Dafür werden geeignete Modellierungswerkzeuge benötigt, die die Verwendung der Merkmale im Teilmodell unterstützen. Auch muss die Spezifikation der Merkmale formal sein.

Entwickler der UI-Lösung haben die Aufgabe funktionsfähige UI-Lösungen unter Verwendung einer beliebigen Anzahl an UI Fragmenten zu erstellen. Dies beinhaltet die Auswahl geeigneter UI-Fragmente in Abhängigkeit der Anforderungen an die UI-Lösung (vgl. Abb.1 „UI Solution Requirements“) sowie anschließend die Verwendung von Entwurfswerkzeugen zur Erzeugung der Lösung. Hierbei müssen UI-Fragmente ggf. zu einer Lösung zusammengesetzt werden, was ggf. nur möglich ist, wenn das Entwurfswerkzeug mit der Implementierungstechnologie der UI-Fragmente kompatibel ist. Diese heute größtenteils manuell ausgeführten Aufgaben sind zukünftig mittels automatischer Werkzeuge auszuführen[1]. Daher soll sich die Aufgabe der Lösungsentwickler möglichst auf die Formulierung der UI-Anforderungen beschränken. Für eine automatisierte Auswahl der gewünschten Fragmente müssen deren Beschreibung und die Anforderungen dasselbe Vokabular verwenden – die UI-Fragment-Merkmale – das daher formal und semantisch eindeutig zu spezifizieren ist.

Aus diesen Betrachtungen ergeben sich die folgenden zu berücksichtigenden Merkmalskategorien:

a) *Anwendungsfallmerkmale* (Use-Case Properties) zur Beschreibung der Nutzeraufgaben und Fragment-Inhalte,
b) *Nutzungskontextmerkmale* (Context-of-Use Properties) zur Beschreibung der Entwurfsannahmen,
c) *Gestaltungsmerkmale (Design Properties)* zur Beschreibung der getroffenen Entwurfsentscheidungen und zur späteren Sicherstellung der UI-Konsistenz sowie
d) *Technische Merkmale (Technical Properties)* zur Sicherstellung der Kompatibilität der UI-Fragment-Implementierungen untereinander und mit dem UI-Entwurfswerkzeug.

2.2. UI-Fragment-Merkmale

Im Folgenden werden die in Abschn. 2.1. identifizierten Merkmalskategorien, aufgrund der hohen Anzahl mit Ausnahme der Kontextmerkmale, näher erläutert sowie aus gängigen Anforderungen an industrielle UI-Lösungen einige relevante Merkmale abgeleitet. Dabei werden die Merkmale in englischer Sprache benannt und zur besseren Lesbarkeit in `dicktengleicher` Schrift gekennzeichnet. Kategorien erhalten zusätzlich das Suffix `Properties`.

2.2.1. Anwendungsfallmerkmale (`Use-Case Properties`)

Die durch das Fragment unterstützte Arbeitsaufgabe ist das relevanteste Auswahlkriterium[12]. Um die in der Literatur geforderte feingranulare inhaltliche Unterscheidung[20], [21] von UI-Fragmenten zu ermöglichen, wird der Anwendungsfall wie folgend dekomponiert: Die *konkrete* Arbeitsaufgabe setzt sich zusammen aus a) dem Aufgabentyp (synonym für *abstrakte* Arbeitsaufgabe oder *Aufgabenklasse*[22]–[24]) und b) der *Betrachtungseinheit* des UI-Fragments (auch *Gegenstand* der Arbeitsaufgabe). Der Anwendungsfall ergibt sich dann aus der konkreten Arbeitsaufgabe (a+b) sowie c) weiteren Parametern zur näheren Bestimmung der konkreten Arbeitsaufgabe. Nach Abb. 2 könnte ein Anwendungsfall bspw. wie folgt dargestellt werden: Als Aufgabentyp wurde „Wartung“ definiert, als Gegenstand „Komponente X“. Ein weiterer Parameter bezeichnet das zugewiesene Gewerk „Mechanik“. Das UI-Fragment soll also speziell für die Wartung mechanischer Aspekte der Komponente ausgewählt werden und entsprechend nur Bestandteile des mechanischen Subsystems darstellen.

Abbildung 2: Dekomposition des UI-Anwendungsfalls

Somit können die nachfolgenden Merkmale spezifiziert werden:

`Supported User Task Type`: Es werden eine oder mehrere unterstützte Aufgabentypen der Benutzenden ausgezeichnet, die im Lebenszyklus einer Anlage auftreten können[3], [25]. Diese können einfache, aber auch zusammengesetzte Komponenten bis hin zur gesamten Anlage betreffen. Teilsysteme müssen konfiguriert und parametriert werden, vor während oder nach dem Einbau; sie müssen während der Inbetriebnahme getestet werden; im Betrieb überwacht, geführt, bedient, beschickt, entladen oder evaluiert und gewartet werden; im Fehlerfall außer Betrieb genommen sowie untersucht, repariert und wieder in Betrieb genommen werden etc. Jeder Aufgabentyp bedingt u.U. abhängig von der jeweiligen Betrachtungseinheit und entsprechend der jeweiligen Teilaufgaben unterschiedliche UI Gestaltungen. `Represented Entities`: Die Betrachtungseinheit des UI-Fragments wird beschrieben. Wenn ein UI-Fragment durch eine I4.0-Komponente ausgeliefert wird, kann i.d.R. implizit davon ausgegangen werden, dass das Asset (bspw. eine Prozesskomponente) der I4.0-Komponente auch der Betrachtungseinheit des UI-Fragments entspricht (vgl. etwa Abb. 1). Jedoch kann eine Komponente beliebig komplex aufgebaut sein und aus mehreren untergeordneten I4.0-Komponenten bestehen. Eine Arbeitsaufgabe mag jeweils eine Teilmenge dieser Unterkomponenten betreffen, sodass ein UI-Fragment für diese Aufgabe nicht einer Unterkomponente zuordenbar ist. Daher wird die Betrachtungseinheit des UI-Fragments ex-

plizit ausgezeichnet, sodass das UI-Fragment der übergeordneten I4.0-Komponente zugeordnet werden kann.

`View Domains`: Als eine von vielen möglichen zusätzlichen Parametern für den Anwendungsfall kennzeichnet dieses Merkmal die Darstellung des Inhalts und der etwaigen Struktur in einer domänenspezifischen Sicht, die spezialisierte Sichten auf die Anlage ermöglicht. Zusätzlich zu der bereits genannten Mechanik-Domäne können Elektrik, andere Energieformen, Automatisierungstechnik, Verfahrenstechnik, IT oder auch Produkt- oder Prozessspezifische Disziplinen relevant sein.

2.2.2. Gestaltungsmerkmale (`Design Properties`)

Im Rahmen der UI-Gestaltung werden häufig im Rahmen eines Projekts oder im Kontext eines Unternehmens *Gestaltungsrichtlinien* festgelegt, die u.a. die Farbgebung, die Symbolik, die Strukturierung etc. festlegen, mit dem Ziel konsistente UI-Lösungen zu erstellen[13], [14], [26]. Als Beispiele dafür können die VDI/VDE Richtlinie 3699-2[27] (Farbkonzept) oder die DIN EN ISO Norm 10628[28] (verfahrenstechnische Symbole) genannt werden, die als zu berücksichtigend festgelegt werden können. Daraus ergibt sich das folgende Merkmal:

`Applied Design Guideline`: Falls möglich, sollen angewendete Gestaltungsrichtlinien ausgezeichnet werden.

Jedoch besteht auch die Möglichkeit, dass keine Gestaltungsregeln anwendbar sind, diese für das konkret zu entwerfende UI-Fragment nicht aussagefähig sind oder bestimmte Entwurfsentscheidungen explizit entgegen der Richtlinien getroffen werden. In diesen Fällen muss das UI-Fragment zusätzlich durch weitere Merkmale beschrieben werden, wie bspw. dem `Representing Symbol` Merkmal, welches einem bestimmten UI-Inhalt (z.B. einer Entität, einem Medium, einem Zustand) eine bestimmte Kodierungsmetapher (z.B. eine Farbe, ein Symbol) zuordnet.

2.2.3. Technische Merkmale (`Technical Properties`)

In industriellen Anwendung wird eine Reihe unterschiedlicher Technologien für Benutzungsschnittstellen verwendet. Dazu zählen Hersteller-spezifische, meistens nicht standardisierte Technologien, aber auch standardisierte, wie das HMI des Module Type Package (MTP), welches per VDI/VDE/NAMUR Norm 2658-2[11] standardisiert ist, oder das UI des Field Device Integration (FDI) Standards, welcher per DIN EN Norm 62769-2[11] standardisiert ist. Weiterhin befinden sich allgemein verfügbare UI Frameworks in Verwendung, bspw. basierend auf Windows Presentation Foundation (WPF)[15], Qt[16] oder auch HTML5[17]. In der Forschungsgemeinschaft sind wiederum domänenspezifische Sprachen, wie MARIA[18] o-

der UsiXML[19] in Verwendung. Zur Abdeckung einer solchen Bandbreite wurden u.A. folgende Merkmale spezifiziert:

`UI Technology Identifier`: Das UI-Fragment verweist auf die Technologie, in der es implementiert wurde.

`UI Technology Version Supported`: Die Version der verwendeten UI-Technologie wird benannt.

2.3. Anmerkungen zu den UI-Fragment-Merkmalen

Als eine erste Version mögen die identifizierten Merkmale ausreichen, um ein gemeinsames Vokabular zwischen UI-Fragment- und UI-Lösungs-Entwicklern (vgl. Abb. 1) bereitzustellen. Darüber hinaus muss die Merkmalsspezifikation zur Verwendung in verschiedenen Werkzeugen (AAS, Anforderungsmodell, Editoren) allerdings rechentechnisch realisiert werden (vgl. Abschn. 2.1). Dieser Aufgabe widmen wir uns in Kap. 4.

Wie bereits angedeutet, sind nicht stets alle Merkmale für die Beschreibung eines UI-Fragments erforderlich. So kann bspw. das `Represented Entites` Merkmal entfallen, wenn das UI-Fragment eine elementare Darstellung (nur eine Entität als Betrachtungseinheit) der Komponente ist, die das UI-Fragment beinhaltet. Andere Merkmale können als redundant angesehen werden, wie zum Beispiel das `Applied Design Guidelines` Merkmal. Dieses ist redundant zu anderen Gestaltungsmerkmalen, falls eine angewendete Design Richtlinie bereits Aussagen zu diesen Merkmalen beinhaltet. Überflüssige Merkmale können ohne Kenntnis aller Designrichtlinien aber nicht ermittelt werden, nicht zum Zeitpunkt des Teilmodell-Entwurfs, aber auch nicht bei der Verwendung eines UI-Fragments. Daher erscheint es sinnvoll, Merkmale zu spezifizieren, obwohl sie redundant sein könnten, und zur Beschreibung von UI-Fragmenten heranzuziehen, sodass eine Auswahlentscheidung (vgl. Abb. 1) auch ohne Kenntnis der Designrichtlinie möglich ist.

Innerhalb der UI-Fragment-Beschreibung muss jedem verwendeten Merkmal ein gültiger oder geforderter Wert zugeordnet werden (vgl. Kap. 3). Dafür muss für jedes Merkmal wenigstens ein Datentyp definiert werden. Für einige Merkmale, wie z.B. das `UI Technology Identifier` Merkmal, reichen primitive Datentypen aus, wie in diesem Fall eine Zeichenkette. Für andere Merkmale wiederum eignen sich enumerierte Datentypen, die die Wertezuordnung vereinfachen können. Es können allerdings nicht alle Merkmale mit primitiven Datentypen beschrieben werden. Nötig sind komplexe Typen, wie Referenzen, Listen, Wertebereiche oder strukturierte Typen. So sollte das `Represented Entities` Merkmal als eine Liste bestehend aus Referenzen auf Entitäten (bspw. die AAS einer dargestellten Komponente) beschrieben werden können, da ein UI-Fragment nicht stets nur eine Entität beinhalten muss. Strukturierte Typen werden dann benötigt, wenn eine Wertzuordnung alleine

nicht ausreichend ist, sondern mehrere kombinierte Aussagen zur Beschreibung eines Merkmals nötig sind. Bspw. müsste das `Representing Symbol` Merkmal sowohl das verwendete Symbol beinhalten, aber auch die symbolisierte Wissenseinheit.

3. Stand der Technik

3.1. Komponentenbasierte Bereitstellung von UI-Fragmenten

Im Stand der Technik industrieller Lösungen für die Bereitstellung von UI-Fragmenten unter Verwendung von Prozesskomponenten sollten die folgenden zwei Ansätze erwähnt werden: Das Human Machine Interface (HMI) des Module Type Package (MTP) spezifiziert per VDI/VDE/NAMUR 2658-2[11] sowie Field Device Integration (FDI) Pakete spezifiziert per DIN EN 62769-2[10].

FDI stellt ein UI für eine einzelne Prozesskomponente basierend auf EDDL und WPF bereit, welches mehrere Gestaltungsvarianten beinhaltet. Diese werden unterschieden mittels eines Merkmals ähnlich des `Supported User Task Types`. Dieses spezifiziert allerdings lediglich drei verschiedene Aufgabentypen. Implizit gilt als `Represented Entity` stets die assoziierte Prozesskomponente. Zwei weitere Merkmale beschreiben den Nutzungskontext: die Plattform (Desktop/Mobil) und die Nutzerrolle (Administrator oder Operator). Wird der technologieneutrale Entwurf der AAS berücksichtigt, erscheint es nicht sinnvoll, die Spezifikation des UI-Teilmodells auf Basis von FDI zu erstellen, da FDI spezifische Plattformeigenschaften voraussetzt (Interpreter und WPF gefordert).

Im Gegensatz dazu ist die MTP-HMI-Spezifikation nicht in der Art auf eine UI-Technologie limitiert und basiert auf einer erweiterbaren UI-Beschreibung für R&I-Fließbilder für Prozessmodule. Deren Technologie (AutomationML) ist der Datenmodellierung der AAS sehr ähnlich (vgl. Abschn. 3.2). Allerdings unterstützt dieser Standard bis jetzt keine Bereitstellung mehrerer UI-Fragmente. Verschiedene UI Gestaltungen können nur während der nötigen Transformation erreicht werden, bei der unter Verwendung eines Merkmals ähnlich des `Represented Entities` Merkmal UI-Fragmente in das Fließbild eingesetzt werden. Dennoch könnte die MTP-HMI Beschreibung entsprechend des UI-Teilmodells (vgl. Kap. 4) erweitert werden.

3.2. Teilmodellspezifikation

In der Literatur zur Verwaltungsschale[8], [9], [29] sind bereits erste Vorschläge zur Teilmodellspezifikation verfügbar. In Abb. 3 ist ein relevanter Ausschnitt der Konzeption sinngemäß dargestellt. Wesentlich ist, dass in einem Teilmodell eine beliebig verschachtelte Datenstruktur mittels der Klasse `StructuralElement` erstellt werden kann. Das Metamodell der AAS ist dabei vollständig generisch und vom konkreten Teilmodell unabhängig. Eine domänen-

spezifische, d.h. Teilmodell-spezifische, Verwendung erfordert jedoch eine semantische Auszeichnung, um eine eindeutige Interpretierbarkeit der Strukturelemente sicherzustellen. Dafür wird das Konzept der *Semantikreferenz*[8], [30] (Attribut `semantics`) verwendet. Diese verweist auf eine extern spezifizierte Wissenseinheit (ein Konzept – zwischen Parteien vereinbarter Begriff[31]), wie z.B. eine Merkmalsspezifikation. Mittels der Klasse `PropertyStatement` können jedem Strukturelement Merkmalsaussagen per Attribut `value` zugeordnet werden, nach dem derzeitigen Stand allerdings ausschließlich primitiven Datentyps. Zur semantischen Auszeichnung der Merkmalsaussage kommt wiederum eine Semantikreferenz zum Einsatz.

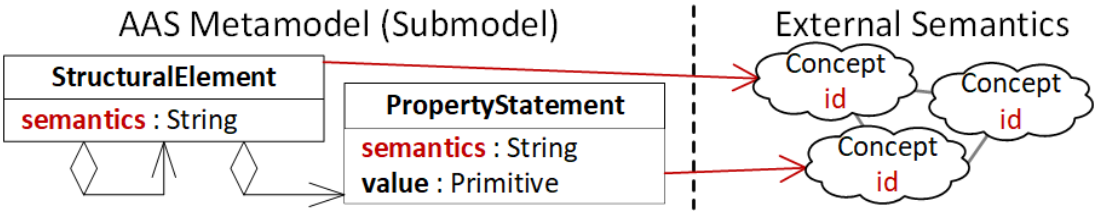


Abbildung 3: Ausschnitt (sinngemäß) des AAS Konzepts, Prinzip der Semantikreferenz

Für die Spezifikation externer Semantiken können verschiedene Werkzeuge verwendet werden, wie die in der Industrie bekannten Merkmalskataloge (auch als semantische Wörterbücher bezeichnet) eCl@ss[32], CDD[33], ETIM[34] usw. Darin werden bspw. Geräteklassen Merkmale zugeordnet (der Klasse „Pumpe“ wird bspw. das Merkmal „Nennleistung“ vom Typ `float` zugeordnet). Ein solcher Eintrag kann ein einfaches Teilmodell mit Bezug auf lediglich eine Entität in Inhalt und Struktur vollständig definieren (so beinhaltet ein Strukturelement zur Repräsentation einer Instanz einer Klasse im Merkmalskatalog lediglich dazugehörige Merkmalsaussagen). Komplexere Teilmodelle benötigen jedoch anstelle einer einfachen Liste von Merkmalen eine Strukturspezifikation (die Bedeutung verschachtelter Strukturelemente muss etabliert werden). Hierfür wurde bereits die Verwendung von UML-Klassendiagrammen vorgeschlagen[8], die auf das AAS-Metamodell abgebildet werden können (Klassen auf Strukturelemente, Attribute/Referenzen auf Merkmalsaussagen, Kompositionen/Aggregationen auf Unter-Strukturelemente). Somit sind auch ähnliche Werkzeuge wie EMF-basierte (Eclipse Modeling Framework[35]) Softwaremodelle, XML Schemata[36] oder Ontologien[31] verwendbar. Unter Nutzung der Semantikreferenzen kann somit die Semantik eines Teilmodells vollständig beschrieben werden.

4. Entwurf des UI-Teilmodells

In diesem Abschnitt präsentieren wir unseren Vorschlag für die komponentenbasierte Bereitstellung von UI-Fragmenten. Da mit dem UI-Teilmodell mehrere UI-Fragmente bzw. –Varian-

ten beschrieben werden sollen (vgl. Abb. 1), ist eine komplexe Strukturierung erforderlich. Im Folgenden wird daher ein UML-Klassendiagramm verwendet.

Aufgrund der bereits vorgeschlagenen Abbildung von UML auf das AAS-Metamodell (vgl. Abschn. 3.2) wären alle Merkmale als Klassenattribute zu beschreiben. Die identifizierten Merkmale müssen jedoch stets als unvollständig angesehen werden. Daher müsste bei jeder nachträglichen Spezifikation neuer Merkmale die Teilmodellspezifikation aktualisiert werden. Aus diesem Grund wird UML lediglich zur Strukturspezifikation verwendet. Darüber hinaus erfolgt die Spezifikation von Merkmalen in einem Katalogwerkzeug, das jederzeit erweitert werden kann, ohne die Teilmodellspezifikation anpassen zu müssen. Der nachfolgende Entwurf ist daher in zwei Teile gegliedert: Zunächst wird eine Teilmodellstruktur spezifiziert (entsprechend Abb. 3 links), anschließend ein Katalogwerkzeug (entsprechend Abb. 3 rechts).

4.1. Strukturentwurf des UI-Teilmodells

Abb. 4 zeigt unseren Vorschlag für eine Teilmodellspezifikation. Als Wurzelelement wird je nach Bedarf die Klasse `UIFragmentRepository` oder `UIFragment` verwendet.

Zunächst besteht jedes Fragment aus einer Implementierung (Klasse `UIFragmentImplementation`) und einem Beschreibungsobjekt (Klasse `UIFragmentDescription`), das die Merkmalsaussagen beinhaltet. Jedoch existieren UI-Technologien, die Adaptionsmechanismen beinhalten. Eine Implementierung kann daher mehrere Gestaltungsvarianten, d.h. mehrere UI-Fragmente, realisieren. Darüber hinaus, kann als UI-Fragment nicht dessen Implementierung verstanden werden, sondern die Repräsentation einer Komponente im UI. Daher kann ein UI-Fragment auch durch mehrere Implementierungen (jede für eine Fragment-Variante) realisiert werden. Daher kann ein UI-Fragment mehrere Implementierungen beinhalten, die jeweils durch mindestens eine Beschreibung charakterisiert werden. Das endgültige Verständnis, was als Fragment, -Variante oder –Implementierung aufgefasst wird, obliegt den Fragment-Entwicklern. Die einzige Anforderung ist jeweils, dass ein Auswahlwerkzeug (vgl. Abb. 1) eine geeignete Implementierung ermitteln und an angeschlossene Werkzeuge weiterreichen kann.

Jede Implementierungen kann per `URI` von externer Stelle zum Download angeboten werden oder per Direktzugriff unter Nutzung von AAS-Funktion als `File`. Das Beschreibungsobjekt dient als Träger der Merkmalsaussagen. Hierbei ist allerdings eine Einschränkung der derzeitigen AAS-Spezifikation zu berücksichtigen (vgl. Abb. 3), die bewirkt, dass Merkmalsaussagen keine komplexen Datentypen als Wert aufnehmen können. Für UI-Fragment-Merkmale werden jedoch auch komplexe Datentypen benötigt, wie Wertebereiche, Listen, strukturierte Werte usw., vgl. Abschn. 2.3. Daher kann im UI-Teilmodell die Klasse `PropertyStatement` nicht zur Darstellung von Merkmalsaussagen verwendet werden. Stattdessen

müsste bspw. für eine Liste ein Strukturelement verwendet werden, welches für jeden Listeneintrag eine Merkmalsaussage enthält. Im Sinne einer konsistenten Modellierung wird in diesem Entwurf daher eine verallgemeinerte Merkmalsaussage (Klasse `GeneralizedPropertyStatement`) verwendet, die ihren Wert in Form eines Objekts (Klasse `PropertyValue`) beinhaltet. Dieser kann in der AAS entweder als Strukturelement oder als Merkmalsaussage realisiert werden. Als `PropertyValue` können sowohl primitive Datentypen, als auch komplexe angelegt werden. Entsprechend den Anforderungen aus Kap. 2 können mittels der verallgemeinerten Merkmalsaussage auch Wertebereiche (Referenzen `minValue` und `maxValue`) oder Wertelisten (Referenz `value`) modelliert werden.

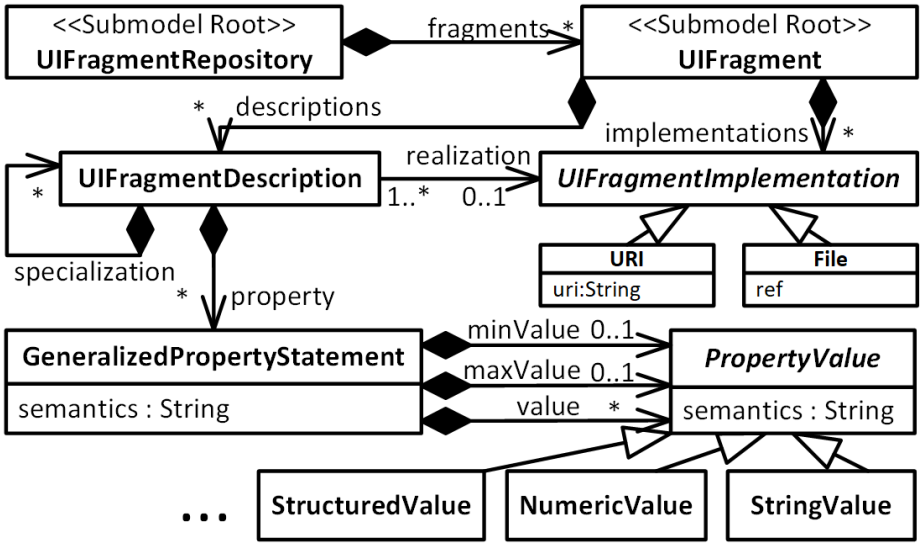


Abbildung 4: UI-Stakeholder

So wie die ursprüngliche Merkmalsaussage muss auch die verallgemeinerte durch eine semantische Referenz auf einen Merkmalskatalog ausgezeichnet werden. Eine weitere semantische Referenz am Merkmalswert ermöglicht die Verwendung polymorpher Datentypen. So können im Merkmalskatalog mehrere Datentypen für ein Merkmal spezifiziert werden.

In Fällen, in denen eine Komponente mit mehreren Fragment-Varianten ausgestattet werden soll, ist im Hinblick auf die große Anzahl an Merkmalen (vgl. Kap. 2) zu erwarten, dass mehrere Varianten eine Mehrzahl an identischen Merkmalsaussagen beinhalten werden und nur eine geringe Anzahl, durch die ein Unterschied der Varianten beschrieben wird. Daher wurde die `specialization`-Referenz eingeführt, die die Verschachtelung von `UIFragmentDescription`-Objekten ermöglicht, wodurch gemeinsame Merkmale in eine übergeordnete Beschreibung extrahiert werden können. Dadurch kann der Modellierungsaufwand in solchen Fällen minimiert werden. Eine abstrakte Fragment-Beschreibung liegt somit dann vor, wenn einer solchen Beschreibung keine Implementierung zugeordnet wurde.

4.2. Katalogspezifikation für Merkmale

Die wichtigsten Anforderungen an einen Merkmalskatalog sind die eindeutige Referenzierbarkeit der Merkmale mittels IDs (vgl. Abb. 3) sowie die Bereitstellung der Spezifikation für Modellierende und deren Werkzeuge. In diesem Fall betrifft das entsprechend Abb. 1 die Modellierungswerkzeuge für das UI-Teilmodell und die UI-Lösungsanforderungen sowie das Auswahlwerkzeug. Entwickelnde müssen auf eine textuelle Darstellung und Erklärungen zu den Merkmalen zugreifen, Entwurfswerkzeuge benötigen hingegen Zugang zu einer formalen Beschreibung der Datentypen je Merkmal, z.B. um Entwickelnde innerhalb eines Editors bei der Modellierung unterstützen zu können. Im Falle der UI-Fragment-Merkmale werden entsprechend der Abschnitte 2.3. und 4.1 primitive und komplexe Datentypen verwendet.

Zur rechentechnischen Realisierung eines Merkmalskatalogs stehen nach Kap. 3 verschiedene Werkzeuge zur Verfügung: So könnten existierende Kataloge, wie eCl@ss, um zusätzliche Merkmale erweitert werden. Dies ist allerdings typischerweise nicht möglich, da die zugrundeliegenden Werkzeuge (d.h. die Metamodelle der Kataloge) lediglich Merkmale primitiven Datentyps erlauben (im Falle von eCl@ss, siehe [37]). Andere Kataloge basieren auf dem durch IEC Norm 61360[33] spezifizierten Metamodell, welches trotz der Bereitstellung von Wertebereichen, Listen usw. allerdings keine weiteren komplexen Datentypen erlaubt.

Daher ist im Kontext des UI-Teilmodells die Entwicklung eines separaten Katalogwerkzeugs notwendig. Hierfür haben wir uns für eine Lösung auf Basis eines EMF-Metamodells entschieden, um effizient einen geeigneten Katalogeditor sowie einen Generator für eine Dokumentationswebseite erstellen zu können. Damit können beliebige Kataloginstanzen erstellt und veröffentlicht werden. Dies wird im Rahmen dieses Beitrags nicht weiter betrachtet.

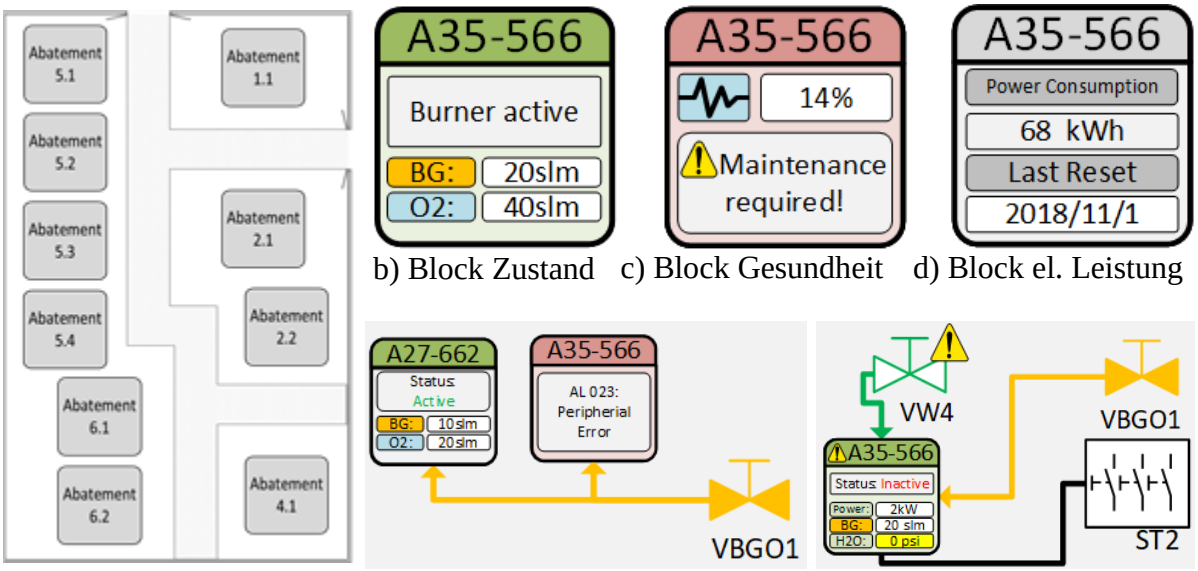
Um im UI-Teilmodell UI-Fragmente beschreiben zu können, wurde ein Katalog für UI-Fragment-Merkmale entworfen, der alle Merkmale aus Abschnitt 2.2. enthält. Da einige Merkmale u.U. auch in anderen Zusammenhängen verwendbar sind, wurden einige Aspekte in separaten Katalogen spezifiziert. Darunter eine Beschreibung typischer Aufgabenklassen, die im Merkmal `Supported User Task Type` referenziert werden können, oder Gewerke, die durch das Merkmal `View Domain` referenziert werden können. Diese könnten im Rahmen der Aufgabenmodellierung und –planung verwendet werden. Die entstandenen Kataloge können online unter https://agtele.eats.et.tu-dresden.de/uimdf eingesehen werden.

5. Fallstudie

In einer ersten Fallstudie wollen wir demonstrieren, dass UI-Fragmente verschiedener Prozesskomponenten dazu verwendet werden können komplexe Darstellungen entsprechend der Anlagenstruktur zu erstellen und dabei verschiedene Anwendungsfälle zu berücksichtigen, die jeweils die Auswahl unterschiedlicher Fragmente bewirken.

In Kooperation mit der Firma *DAS Environmental Experts GmbH* (https://www.das-ee.com) haben wir hierfür einen komponentenbasierten UI-Entwurf für ein Abgasreinigungssystem erstellt. Dabei ist die Aufgabe eines einzelnen Moduls die Entsorgung von Prozessgasen der Halbleiterindustrie. Das Ziel der Fallstudie war der Entwurf von Ansichten eines Monitoring-Systems, welches an Standorten eingesetzt werden kann, wo mehrere Anlagen der Firma installiert sind. Die Ansichten basieren dabei auf einem Etagenplan (siehe Abb. 5.a), in den verschiedene Block-Darstellungen der Anlagen eingefügt werden sollen. U.a. stellen die Anlagen hierfür UI-Fragmente für den Nominalbetrieb (Abb. 5.b), zur Wartungsplanung (Abb. 5.c) und zur Überwachung des Elektroenergieverbrauchs (Abb. 5.d) bereit. Für die Außer- und Inbetriebnahme einer einzelnen Anlage soll der Etagenplan die Verbindungen der Anlagen mit ihrer Peripherie aufzeigen. Hierbei wird berücksichtigt, dass nicht stets 1:1-Beziehungen zwischen Komponenten der Peripherie und den Reinigungsanlagen besteht, weshalb komplexe Strukturdarstellungen benötigt werden. Abb. 5.e) zeigt einen exemplarischen Fall, bei dem mehrere Anlagen an einen Strang der Gasversorgung angeschlossen sind. Falls die Versorgung zu unterbrechen ist, bspw. um eine einzelne Anlage auszutauschen, ist dies nicht möglich, ohne dass alle Anlagen am selben Strang abgeschaltet werden. Dieselbe Aufgabe kann unter Nutzung des View Domain Merkmals auch in anderen Teilsystemen realisiert werden. Für die Fehleranalyse einer einzelnen Anlage müssen weitere verschiedene Versorgungssysteme berücksichtigt werden, wie Wasser, Brenngas und Elektroenergie (vgl. Abb. 5.f). Zunächst werden die drei UI-Fragment-Versionen b), c) und d) der AAS einer Reinigungsanlage zugeordnet und durch die Merkmale aus Abschnitt 2.2 beschrieben, siehe Tab. 1. Die Übersichtsbilder 5.e) und 5.f), die auf Basis des Etagenplans 5.a) erstellt wurden sowie der leere Etagenplan selbst können ebenfalls als UI-Fragmente verstanden werden und werden daher in einer übergeordneten AAS beschrieben, siehe Tab. 2. Die ausgewählten Fragmentmerkmale nutzen Referenzen (Symbol →) auf Einträge in den zuvorgenannten zusätzlichen Katalogen. Diese Referenzen werden unter Verwendung von Zeichenketten realisiert (vgl. Abb. 4), die auflösbare IDs als Werte beinhalten. Aufgrund der Merkmalsspezifikation im Katalog können die Referenzen validiert werden, sodass korrekte Referenzziele sichergestellt werden können. Der Teilmodelleditor unterstützt Entwickelnde, durch Vorschläge für mögliche Referenzziele. Somit gelten die genannten Anforderungen (vgl. Abschn. 2.1) an Editoren und die Merkmalsspezifikation als erfüllt.

In diesem ersten Schritt einer umfangreicheren Fallstudie wurde die Anwendbarkeit der identifizierten Merkmale demonstriert. Allerdings ist die Demonstration einer automatischen Fragment-Auswahl sowie der detaillierten UI-Teilmodell- und Anforderungs-Modellierung noch ausstehend.



a) Lageplan b) Block Zustand c) Block Gesundheit d) Block el. Leistung

e) Strang Gas f) Peripherie einer Anlage

Abbildung 5: Lageplan und verschiedene UI-Fragmente der Reinigungsanlage

Tabelle 1: Merkmalsaussagen für UI-Fragmente 5.b), 5.c) u. 5.d). Symboldefinition: (1) `Supported User Task Type`; (2) `Represented Entities`; (3) `View Domain`; (4) `Applied Design Guideline`; → Referenz auf Objekt/ externe Definition

Merkmal	Aussagen 5.b)	Aussagen 5.c)	Aussagen 5.d)
(1)	→ supervision	→ monitoring	→ supervision
(3)	∅	∅	→ electrical energy
(4)	[DAS UI Design Guideline Identifier]		

Tabelle 2: Merkmalsaussagen für UI-Fragmente 5.a), 5.e) u. 5.f). Symboldefinition, s. Tab. 1

Merkmal	Aussagen 5.a)	Aussagen 5.e)	Aussagen 5.f)
(1)	→ navigation (in plant)	→ commissioning	→fault analysis
(2)	→ floor → DAS Abatement Type	→ A27-662 → A35-566 → VGB01	→ A35-566 → VW4 → VBG01 → ST2
(3)	∅	→ gas supply	→ gas supply → water supply → el. Power
(4)	∅	[DAS UI design Guideline Identifier] [DIN EN ISO 10628 Identifier]	

6. Zusammenfassung und Ausblick

In diesem Beitrag wurde die erste Version unseres Vorschlags für die Spezifikation eines UI-Teilmodells für die Industrie 4.0 Verwaltungsschale vorgestellt. Unter Verwendung dieses Modells können beliebig viele UI-Fragmente mittels der AAS bereitgestellt und durch formali-

sierte Merkmale semantisch ausgezeichnet werden. In Zukunft können die in diesem Beitrag eingeführten Merkmale dazu verwendet werden mittels eines automatischen Werkzeugs ein geeignetes Fragment zu identifizieren und zum Einfügen in eine UI-Lösung zu auszuwählen. Um die Merkmale in einem generischen UI-Teilmodell einheitlich und eindeutig verwenden zu können, wurden mehrere Merkmalskataloge erstellt. Deren Verwendung in Merkmalsaussagen wurde bereits in einer ersten Fallstudie an einem industriellen Anwendungsfall getestet, wodurch sich auch unser Ansatz als tragfähig erwiesen hat. Das Metamodell der UI-Teilmodells, die Kataloge und dazugehörige Werkzeuge wurden zum *UI Meta Description Framework* (UIMDF) zusammengefasst, welches zukünftig in den Repositories unseres Instituts (https://gitlab.com/tud-ifa) veröffentlicht werden.

Es verbleibt der Bedarf an weiterer Forschung: Wir erwarten keinen zusätzlichen Aufwand bei der Integration der hier vorgestellten Fragmentmerkmale in existierende UI-Gestaltungsprozesse, da die identifizierten Fragment-Merkmale lediglich als eine Formalisierung von Randbedingungen für die UI-Gestaltung angesehen werden können, die in einem fundierten UI-Entwurf ohnehin zu beschreiben sind. Dennoch würde die Erstellung und Wartung mehrerer UI-Fragmente für eine einzelne Komponente, die in einem PnP-Szenario benötigt werden, einen ressourcenintensiven Prozess darstellen. Aufgrund der großen Anzahl an Fragment-Merkmalen entsteht sehr schnell das Problem der Variantenexplosion. Für diesen Fall wurde bereits die Funktion der Abstraktion der UI-Fragment-Beschreibung eingeführt. Allerdings sind weitere Ansätze, etwa aus dem Gebiet der modellbasierten UI-Entwicklung[38] (MBUID), zu untersuchen, um auch den Aufwand der Erstellung der Fragment-Implementierungen zu reduzieren. Hier existieren bereits Ansätze, die die Adaption eines UI-Fragments an den gegenwärtigen Nutzungskontext per Parametrisierung erlauben[39]. Für diesen Zweck müssen auch Merkmale spezifiziert werden, die den intendierten bzw. angenommenen Nutzungskontext beschreiben. Leider existieren in diesem Gebiet nur wenige Beiträge, die Kontexteigenschaften formal beschreiben[40]. Daher müssen zukünftig geeignete Merkmale zur Beschreibung des Nutzungskontexts identifiziert und in Katalogen spezifiziert werden. Zusätzlich sind die Anforderungen an eine UI-Lösung und ein automatisches Auswahlwerkzeug zu entwerfen, was aufgrund von Merkmalen mit komplexen Datentypen, die im UI-Teilmodell verwendet werden und mit den UI-Anforderungen zu vergleichen sind, ein nicht-triviales Problem darstellt. Weiterhin muss der Erstellungsprozess einer UI-Lösung auf Basis dieses Auswahlwerkzeugs berücksichtigt werden, was zur Spezifikation weiterer Fragment-Merkmale führen kann, bspw. zur Parametrierung der UI-Fragmente. Schließlich wollen wir die identifizierten Merkmale in weiteren Fallstudien intensiver untersuchen und ausbauen.

7. Danksagung

Wir danken der Firma *DAS Environmental Experts GmbH* für die freundliche Unterstützung sowie *Maksym Lebedyk* für seine Arbeiten an der Fallstudie.

Literatur

[1] M. Schmidt, U. Löwen u. a., „WFF - Wandlungsfähige Fabrik - Langfassung“. Juni 29, 2016.
[2] H. Bedenbender *u. a.*, „Industrie 4.0 Plug-and-Produce for Adaptable Factories: Example Use Case Definition, Models, and Implementation“, Juni 2017. [Online] https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/Industrie-40-%20Plug-and-Produce.html.
[3] Deutsches Institut für Normen, *Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)*. Berlin: Beuth, 2016.
[4] D. Zühlke, *Nutzergerechte Entwicklung von Mensch-Maschine-Systemen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
[5] DIN-NAErg und DIN-NIA, Ergonomie der Mensch-System-Interaktion – Teil 11: Gebrauchstauglichkeit: Begriffe und Konzepte (ISO 9241-11.2:2016); Deutsche und Englische Fassung prEN ISO 9241-11:2016. Berlin: Beuth Verlag GmbH, 2017.
[6] DIN-NAErg, Ergonomie - Genereller Ansatz, Prinzipien und Konzepte (ISO 26800:2011); Deutsche Fassung EN ISO 26800:2011. Berlin: Beuth Verlag GmbH, 2011.
[7] Plattform Industrie 4.0, „Verwaltungsschale in der Praxis“, Plattform Industrie 4.0, VDI, VDE, Gesellschaft für Mess- und Automatisierungstechnik, Apr. 2019.
[8] Plattform Industrie 4.0, „Verwaltungsschale im Detail - Part 1 - The exchange of information between partners in the value chain“. Nov. 2018.
[9] AG Referenzarchitekturen, Standards und Normung, „Struktur der Verwaltungsschale“. Plattform Industrie 4.0, Apr. 20, 2016, Zugegriffen: Mai 15, 2019. [Online] https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/struktur-der-verwaltungsschale.html.
[10] DKE DIN und VDE, Feldgeräteintegration (FDI) – Teil 1: Überblick (IEC 62769-1:2015); Englische Fassung EN 62769-1:2015. Berlin: Beuth Verlag GmbH, 2016.
[11] VDI/VDE-GMA, Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie - Modellierung von Bedienbildern. Berlin: Beuth Verlag GmbH, 2018.
[12] DIN-NAErg und DIN-NIA, Ergonomie der Mensch-System-Interaktion - Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme. Berlin: Beuth, 2011.
[13] A. Butz und A. Krüger, *Mensch-Maschine-Interaktion*, 2. Auflage. Berlin/Boston: Walter de Gruyter GmbH, 2017.
[14] N. Bevan, „International standards for HCI and usability“, *Int. J. Hum.-Comput. Stud.*, Bd. 55, Nr. 4, S. 533–552, Okt. 2001.
[15] Wikipedia, „Windows Presentation Foundation“, Mai 14, 2019. https://de.wikipedia.org/wiki/Windows_Presentation_Foundation.
[16] The Qt Company, „Qt“, Juli 01, 2019. https://www.qt.io/.
[17] S. Faulkner, A. Eicholz u.A., *HTML 5.2*. W3C, 2017.
[18] F. Paternò, C. Santoro, und L. D. Spano, „MARIA: A Universal, Declarative, Multiple Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments“, *ACM Trans. Comput.-Hum. Interact.*, Bd. 16, Nr. 4, S. 1–30, Nov. 2009.
[19] Q. Limbourg, J. Vanderdonckt u. a., „Usixml: A user interface description language supporting multiple levels of independence“, *Eng. Adv. Web Appl.*, S. 325–338, 2004.
[20] T. Stiedl, J. Diemer, und Schulz, „AUP - Anwenderunterstützung in der Produktion - Langfassung“. März 31, 2016.

[21] D. Gorecky, M. Schmitt u. a., „Human-machine-interaction in the industry 4.0 era“, in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, Porto Alegre RS, Brazil, Juli 2014, S. 289–294.
[22] D. Diaper und N. A. Stanton, „Wishing on a sTAr: The Future of Task Analsysis“, in *The Handbook of Task Analysis for Human-Computer Interaction*, Mahwah, New Jersey: Lawrence Erlbaum Associates, Inc., 2004, S. 603–619.
[23] F. Paternò, „ConcurTaskTrees: an engineered approach to model-based design of interactive systems“, *Handb. Anal. Hum.-Comput. Interact. Lawrence Erlbaum Assoc.*, 2002.
[24] P. A. Akiki, A. K. Bandara, und Y. Yu, „Engineering Adaptive Model-Driven User Interfaces“, *IEEE Trans. Softw. Eng.*, Bd. 42, Nr. 12, S. 1118–1147, Dez. 2016.
[25] NAMUR-NA 35-AF1, *PLT-Planung und -Abwicklung in der Prozessindustrie*. Leverkusen: NAMUR e.V., 2019.
[26] C. Denis und L. Karsenty, „Inter-Usability of Multi-Device System - A Conceptual Framework“, in *Multiple user interfaces: cross-platform applications and context-aware interfaces*, Hoboken, NJ: J. Wiley, 2004.
[27] VDI/VDE-GMA, *Prozessführung mit Bildschirmen: Grundlagen*. Berlin: Beuth Verlag GmbH, 2014.
[28] Fließschemata für verfahrenstechnische Anlagen Allgemeine Regeln (ISO 10628:1997) Deutsche Fassung EN ISO 10628:2000. Berlin: Beuth Verlag GmbH, 2001.
[29] Platform Industrie 4.0, „Die Verwaltungsschale im Detail - von der Idee zum implementierbaren Konzept“. Platform Industrie 4.0, Juli 01, 2019, [Online] https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/verwaltungsschale-im-detail-pr%C3%A4sentation.html.
[30] C. Hildebrandt *u. a.*, „Semantische Allianz 4.0: Semantische Inhalte für Industrie 4.0“, gehalten auf der Automation 2017, Düsseldorf, 2017, Bd. VDI Berichte 2293, S. 262.
[31] T. R. Gruber, „Toward principles for the design of ontologies used for knowledge sharing?“, *Int. J. Hum.-Comput. Stud.*, Bd. 43, Nr. 5, S. 907–928, Nov. 1995.
[32] ecl@ss e.V., „The eCl@ss-Standard“, Okt. 16, 2019. https://www.eclass.eu/standard.html (zugegriffen Nov. 15, 2019).
[33] International Electrotechnical Commission, „IEC 61360 - Common Data Dictionary (CDD - V2.0014.0016)“, Nov. 11, 2019. https://cdd.iec.ch/cdd/iec61360/iec61360.nsf.
[34] ETIM International, „ETIM International“, Nov. 11, 2019. http://community.etim-international.com (zugegriffen Nov. 11, 2019).
[35] Eclipse, „Eclipse Modeling Framework (EMF)“, 2016. http://eclipse.org/modeling/emf/ (zugegriffen Mai 25, 2016).
[36] D. C. Fallside und P. Walmsley, Hrsg., *XML Schema Part 0: Primer Second Edition*. W3C, 2004.
[37] „Property - wiki.eclass.eu“, *eCl@ss Wiki*, Nov. 08, 2019. http://wiki.eclass.eu/wiki/Property (zugegriffen Nov. 08, 2019).
[38] G. Calvary, J. Coutaz u. a., „A Unifying Reference Framework for multi-target user interfaces“, *Interact. Comput.*, Bd. 15, Nr. 3, S. 289–308, 2003.
[39] M. Freund, C. Martin, und A. Braune, „A Library System to Support Model-Based User Interface Development in Industrial Automation“, in *Human-Computer Interaction. Theory, Design, Development and Practice. HCII 2016.*, Cham, 2016, Bd. 9731, S. 476–487.
[40] C. Loitsch, G. Weber u. a., „A knowledge-based approach to user interface adaptation from preferences and for special needs“, *User Model. User-Adapt. Interact.*, Bd. 27, Nr. 3–5, S. 445–491, Dez. 2017.