Interner Projektbericht zum Themenfeld mimesis.model

Ansatz zur datenzentrierten Modellierung von Anwendungsvarianten für dialogbasierte Anwendungen in einem Multikanalumfeld

Stand 01.2014 (draft)

Dipl.-Inform. Michael Hitz
Duale Hochschule Baden-Württemberg - Stuttgart
Fachbereich Wirtschaftsinfomratik
Paulinenstraße 50
70178 Stuttgart
michael.hitz@dhbw-stuttgart.de

Inhaltsverzeichnis

1 Änderungshistorie

Version — Datum — Änderungen
1.0 — 1.2014 — Initiale Ausgabe

2 Vorbemerkung

Der vorliegende Projektbericht stellt den aktuellen Stand (1.2014) der Untersuchungen hinsichtlich der datenzentrierten Modellierung von Anwendungsvarianten im Projekt *mimesis* dar (Teilprojekt *mimesis.model*).

Die hier dargestellten Teilergebnisse werden im weiteren Projektverlauf vertieft und stellen lediglich den *aktuellen Zustand* der laufenden Arbeiten dar.

3 Motivation und Ziele

Das übergeordnete Themenfeld *mimesis* betrachtet ein Konzept zur Erstellung multikanalfähiger Anwendungen durch Bildung von Varianten in einer hierzu geeigneten Architektur.

In [Hit13b] wurde als Motivation für das Vorhaben angeführt, dass der Zuschnitt der webbasierten Frontend-Systeme auf bestimmte Nutzergruppen (*fachliche Kanäle*) in den vergangenen Jahre häufig dazu führte, dass parallel unterschiedliche Portale gebaut wurden, die inhaltlich sehr ähnliche bis gleiche Anwendungen enthielten, die aber speziell für jede Nutzergruppe neu entwickelt wurden. Oberflächen und Anwendungsabläufe wurden mehrfach entwickelt, obwohl sie in anderen Portalen bereits als Variante verfügbar waren. Kam ein neuer Kanal hinzu, der Zugang zu den Systemen des Unternehmens benötigte, wurde ein weiteres Portal erstellt.

Die wachsende Notwendigkeit, Anwendungen auch für unterschiedliche Endgräte (*technische Kanäle*) zur Verfügung zu stellen, stellt ebenfalls Anforderungen an die Anwendungen. So müssen Dialogschnittstellen speziell für die Eigenschaften des Geräts bereitgestellt - aber auch die unterschiedlichen Darstellungsmöglichkeiten der Geräte (z.B. kleine Displays) berücksichtigt werden.

Um die wiederkehrende Erstellung von ähnlichen Anwendungen zu vermeiden, ist ein Ansatz erforderlich, in welchem die Grundanwendung beschrieben werden und dann Varianten daraus durch Bildung von Erweiterungen erstellt werden können. Das Ziel ist, eine Anwendung multikanalfähig zu erstellen - eine einzige Anwendung, welche zur Verwendung durch mehrere Kanäle geeignet ist und deren Spezifika berücksichtigt.

Soll eine Anwendung für mehrere Kanäle eingesetzt werden können, hat dies Auswirkung auf zwei zentrale, nicht voneinander unabhängige Bereiche:

- **den Anwendungsfluß**: abhängig von der Nutzergruppe existieren ggf. Varianten in den Abläufen einer Anwendung.
- **die Dialogschnittstellen** : abhängig von der Nutzergruppe und verwendeten Ausgabegeräten werden unterschiedliche Dialogschnittstellen benötigt.

Für beide Bereiche müssen Varianten der Grundanwendung definiert werden können, mit denen ein Nutzer der Anwendung arbeitet.

Das in diesem Papier beschriebene Themenfeld *mimesis.model* befasst sich mit einer Lösung zur Beschreibung von fachlichen Anwendungunsvarianten, aus welchen in einem Folgeschritt automatisiert technologische Varianten der Benutzerschnittstellen einer multikanalfähigen Anwendung erzeugt werden können. Die Beschreibung der Anwendung erfolgt dabei auf einem Modell der in der Anwendung verarbeiteten Daten unter Hinzunahme von Metainformationen, welche die Semantik der Daten näher beschreiben.

Um eine Einordnung dieses Papers in den Gesamtkontext von *mimesis* geben zu können, folgt eine kurze Zusammenfassung der Grundidee des mimesis-Ansatzes:

Als Ausgangspunkt zur Lösung des Problems dient als Grundannahme, dass *datensammelnde Anwendungen* (s. [Hit13b][1] auf einer Menge von Daten operieren, die durch den Anwendungsfluss bestimmt ist. Anwendungen haben einen Anwendungsfluss, der i.S. eines *Workflows* aus einzelnen Schritten (*Tasks*) besteht, in denen Daten gesammelt oder Daten konsumiert werden. Die Menge der Daten, die in der Anwendung verarbeitet wird, ist über die Tasks bestimmt (Vereinigungsmenge aller verarbeiteten Daten der Tasks). Diese Menge stellt das *Datenmodell* der Anwendung dar.

Da die Anwendungsflüsse sich in einem Multikanalkontext i.S. von Varianten unterscheiden, besteht damit das Datenmodell einer *multikanalfähigen Anwendung* aus der Vereinigung der Datenmodelle (Gesamtdatenmodell), die von den kanalspezifischen Varianten in den Abläufen abgeleitet werden können. Eine kanalspezifische Variante arbeitet im Umkehrschluss auf einer Untermenge des Gesamtdatenmodells.

Die Lösungsidee zur Beschreibung von Varianten für Dialogschnittstellen besteht darin, dass sich das Gesamtdatenmodell (in einfacher Weise) um Metainformationen anreichern lässt, sodass daraus automatisiert eine kanalspezifische Dialogschnittstellenbeschreibung hergeleitet werden kann.

Die zu beweisende Annahme ist dabei, dass für den Fall der *datensammelnden Anwendungen* die Metainformationen datenbezogen angegeben werden können. Im Gegensatz zu bestehenden Ansätzen wird davon ausgegangen, dass für diese Anwendungskategorie keine Dialogschnittstellenspezifika modelliert werden müssen, sondern die kanalspezifischen Varianten der Dialogschnittstelle aus einer qualitativen Beschreibung des Datenmodells hergeleitet werden können.

Die Hauptfragestellung besteht darin, wie die Metainformationen beschaffen sein müssen, dass daraus Dialogbeschreibungen hergeleitet werden können, die hinsichtlich der bearbeiteten Daten den Anforderungen der einzelnen Kanäle entsprechen.

4 Forschungsfragen, Beitrag der Untersuchungen und Abgrenzung

Zur Lösung der Problemstellung wird folgenden Forschungsfragen im Projekt nachgegangen:

- kann basierend auf der Gesamtmenge der über alle Kanalvarianten verarbeiteten Daten unter Hinzunahme von Metainformationen ein Modell beschrieben werden, das die automatisierte Herleitung von kanalspezifischen Dialogschnittstellen gestattet?
- Welche Kontextinformationen für einem Kanal sind notwendig zur Ableitung einer kanalspezifischen Teilmenge des Gesamtdatenmodells (Variante) und wie kann dies in einem Metamodell beschrieben werden?
- Ist es möglich, Metainformationen im Datenmodell bereitzustellen, die eine vollständig automatisierte Herleitung von Dialogbeschreibungen ermöglicht? Können alle relevanten Anforderungen erfüllt werden (siehe Themenfeld *mimesis.ui*)?
- Wo liegen die Einschränkungen hinsichtlich der Flexibilität bei einer vollständigen automatisierten Herleitung der Dialogschnittstellen?

[1] Der Begriff bezieht sich auf die Anwendungskategorie der datensammelnden Anwendungen, auf die sich die Untersuchungen des Promotionsvorhabens fokussieren. Eine Erläuterung befindet sich neben den bereits veröffentlichten Papieren im Papier zu *mimesis.ui*.

- Wie weit können die Metainformationen datenbezogen (qualitativ) formuliert werden und dennoch zur Herleitung einer vollständigen Dialogschnittstelle dienen?
- Ist dies in einer *einfachen Beschreibung* formulierbar, die einerseits verständlich und wartbar bleibt?

Im Bereich der modellgetriebenen Entwicklung (insbesondere Bereich des Web Engineering) existieren in der wissenschaftlichen Literatur bereits Ansätze, die basierend auf Modellen Anwendungen herleiten. Diese haben meist die explizite Modellierung aller Anwendungsaspekte zum Inhalt, mit dem Ziel, ganze Portale mit komplexen Abläufen und Navigationsmöglichkeiten abbilden zu können. Diese Anforderung besteht jedoch in vielen aktuellen Portalen nicht mehr, da diese häufig Content-getrieben sind und Anwendungen im Sinne von kleinen Bausteinen in die Inhalte integriert werden.

Durch die thematische Nähe zur Webentwicklung sind die Ansätze und deren Beschreibungsform häufig an den Anforderungen von Webtechnologien ausgerichtet. Zudem ist in den meisten Fällen die Problemstellung der Erzeugung von **Varianten und die Multikanalfähigkeit** der modellierten Anwendungen nicht im Fokus der Konzepte. Die Umsetzung einer solchen Anforderung resultiert dort meist in der *expliziten Modellierung* einer weiteren Variante - mit der damit verbundenen Gefahr von inkonsistenten Varianten bei der Wartung.

Für die im Projekt vorgenommene Konzentration auf den Anwendungsfall der *datensammelnden Anwendungen* besteht durch den im Projekt verfolgten Ansatz ein Optimierungspotential. Die Konzentration gestattet eine Vereinfachung der Anforderungen und damit der zu modellierenden Aspekte, wirkt sich aber auf die Universalität der Lösung aus. Dennoch ist die Einschränkung der Universalität zu rechtfertigen, da ein Großteil der Anwendungen - nicht nur im Portalbereich - auf diesem Prinzip beruhen.

Der Fokus der im Projekt angestrebten Lösung liegt auf der automatisierten Herleitung möglichst vieler/aller Aspekte einer Dialogschnittstelle und in einer weitestgehenden Technologieneutralität. Das entstehende Modell soll ein Minimum an zu bearbeitenden Artefakten aufweisen und per se die Ableitung von aus einem zentralen Modell gestatten. Hierzu soll so wenig wie möglich explizit modelliert werden müssen um Redundanzen zu vermeiden und zu einer konsistent wartbaren, multikanalfähigen Dialogschnittstelle zu gelangen.

Die Arbeiten in *mimesis.model* liefern eine Beitrag im Umfeld der modellbasierten Erzeugung von Anwendungen. Der Beitrag besteht in einem Konzept zur Generierung von Dialogschnittstellenvarianten aus Modellbeschreibungen in einem Multikanalkontext. Hierbei wird eine technologieneutrale Beschreibung des Datenmodells mit Meatinformationen erarbeitet, welche die Basis zur Herleitung von Varianten gestattet. Damit wird ein neuer Beitrag zum Aufbau von multikanalfähigen Anwendungen basierend auf einer qualitativen Beschreibung des Modells einer Anwendung über Metainformationen geschaffen.

5 Arbeiten und Technologien im Umfeld - Ansatzpunkte

Work in progress: Die hier aufgeführtenReferenzen sind aktuell lediglich als Merker angegeben. Eine detailliertere Beschreibung der Inhalte erfolgt in weiteren Versionen des Dokuments. Sie dienen vorerst als *Pool* potentieller Referenzen. Zudem finden sich hier *Stichworte* unter welchen eine größere Anzahl an Literatur existiert, die künftig noch auf Eignung ausgewertet wird.

- WebML: Modellierung datenzentrierter Webanwendungen (CRUD)
- MBUI, IFML (OMG),UIML,

- UWE: genereller Ansatz zur Modellierung von Webanwendungen aus UML-Modellen
- weitere (s. Literaturrecherche)
- "Janus (Balzert 1997)",
- CAMELEON-Projekt: [CCT+03, CCT+02].Geht in die Richtung, in der auch wir arbeiten. Hier werden viele Themen ebenfalls angeschnitten. http://giove.isti.cnr.it/projects/cameleon.html + http://giove.isti.cnr.it/proje
- Chameleon: http://www.di.univaq.it/chameleon/
- Softwareproduktlinien: hier nicht sicher, ob ein "zu großes Fass" aufgemacht würde. Unterschiede aber auch im Ansatz. SPL: Beschreibe Features, wähle Features für Varianten.. Hier: Beschreibe Grundablauf, erweitere an geeigneten Stellen.
- MDD + DSLs
- Metainformationen?: hier noch nichts gefunden, was auf Modellebene "modellnah" beschreibt was für Oberfläche wichtig. Grundlagen (z.B. Typisierung und deren WIchtigkeit. . . .")

6 Anforderungen an eine multikanalfähige Modellbeschreibung in *mimesis.model*

Das im Projekt verfolgte Ziel, weitgehend automatisiert kanalspezifische Varianten einer Anwendung aus einem Basismodell herleiten zu können, motiviert für den Bereich der Dialogschnittstellen die Forderung, Varianten der Dialoge basierend auf Modellinformationen herzuleiten.

Im Themenfeld *mimesis.ui* [Hit13a] wird untersucht, wie eine technologieneutrale Schnittstellenbeschreibung aussehen kann, aus der konkrete Benutzerschnittstellen für unterschiedliche *technische Kanäle* hergeleitet werden. Diese geht davon aus, dass die Dialogbeschreibungen bereits aus *fachliche Kanäle* und deren Belange zugeschnitten sind. Dier hierzu benötigten Informationen müssen in einem Ausgangsmodell enthalten sein.

Um *fachlichen Eigenschaften* in kanalspezifische Varianten von Dialogbeschreibungen münden zu lassen, bedarf es im Ausgangsmodell weiterer Informationen. Diese müssen beschreiben können, welche Teile des Modells für den betrachten Kanal relevant sind.

Da das Ausgangsmodell nicht nur für die Herleitung einer Benutzerschnittstelle relevant ist, sondern auch eine Referenz für die Abläufe der Anwendung darstellt, ist es freizuhalten von Eigenschaften, die explizit dialogspezifisch sind. Es muss eine *datenzentrierte* Beschreibung gefunden werden.

Die grundsätzlichen Anforderungen an das Datenmodell, welches im Rahmen des Ansatzes gelten sollen, sind im Folgenden zusammengefasst:

- Es muss möglich sein, automatisiert Dialogschnittstellen-Beschreibungen aus dem Ausgangsmodell herzuleiten.
- Die Bildung von Varianten für Kanäle / Kanalkombinationen muss beschrieben werden können. Die Beschreibung der für einen Kanal relevanten Untermengen des Datenmodells soll an einer Stelle definierbar sein, sodass einfach neue Varianten hinzugefügt werden können.
- Die Formulierung des Ausgangsmodells soll frei vom Kontext sein, in dem die Modellbeschreibung später eingesetzt werden soll. Dies bedeutet, dass die Metainformationen datenbezogen definiert und von Spezifika einer Dialogschnittstelle frei sein müssen. Dennoch müssen alle Informationen für die automatisierte Herleitung einer Dialogschnittstelle ableitbar sein.

- Es soll eine möglichst geringe Zahl konsistent zu haltender Artefakte verwendet werden.

Im Folgenden werden diese und sich daraus ergebende Anforderungen näher beleuchtet.

6.1 Automatische Herleitung von Dialogschnittstellen aus einem Datenmodell

Das Modell muss alle Informationen enthalten, die zur Erzeugung einer Dialogschnittstelle[2] benötigt werden. Da die Art der Dialogschnittstellen in einem Multikanalsystem sehr unterschiedlich sein kann, muss die Modellbeschreibung abstrakt genug sein.

In *mimesis.ui* [Hit13a] wurden bereits Anforderungen formuliert, die eine Dialogschnittstelle erfüllen muss und daraus Beschreibungsmöglichkeiten abgeleitet. Die hierzu benötigten Informationen müssen bei der automatischen Herleitung der Beschreibungen verfügbar sein und sind somit im Ausgangsmodell zu berücksichtigen. Dadurch kann die Analyse in *mimesis.ui* auf die Grundlage der Betrachtungen für das Datenmodell sein. Die dort betrachteten Anforderungen waren:

- Es muss eine Aufteilung der zu erfassenden Daten in in Dialogeinheiten erfolgen und eine Navigation zwischen den Einheiten hergeleitet werden können.
- Passende **Ein-/Ausgabekomponenten** für Daten bestimmten Typs müssen angeboten werden.
- Eine **Validierung** der eingegebenen Daten muss möglich sein.
- Eine **Sichtbarkeitssteuerung** von relevanten / irrelvanten Bereichen muss umgesetzt werden können.
- **Dynamische Inhalte und Initialwerte** für Felder müssen bestimmt werden können.
- **Bindung an das** Modell muss abgebildet werden können.
- **Benutzer-Aktionen** für Bereiche müssen angegeben werden können.

Die Beschränkung auf das Szenario der *datensammelnden Anwendungen* erlaubt die Annahme, dass für einen *sammelnden Schritt* alle Daten bekannt sind und das Datenmodell anwendungsbezogen definiert werden kann.

Zudem können Beziehungen und die Zusammengehörigkeit der Daten in diesem Kontext und abhängig vom Anwendungsfall angegeben werden[3]. Dies wiederum ermöglicht die Herleitung von Dialogschnittstellen für den Anwendungsfall.

Um eine Dialogschnittstelle aus einem Datenmodell herleiten zu können, muss bekannt sein, wie die Daten *semantisch zusammenhängen*. Semantisch zusammengehörende Daten werden in einem Erfassungsdialog üblicherweise gruppenweise erfragt. Die Herleitung einer Dialogschnittstelle umfasst die Aufgabe, solcher Sinn-Einheiten dem Nutzer präsentieren zu können (vgl. Betrachtungen in *mimesis.ui*).

[2] Die Verwendung des Wortes *Dialogschnittstelle* wurde gewählt, um zu verdeutlichen, dass es sich hierbei nicht unbedingt um eine grafische Benutzeroberfläche handeln muss. Hierunter könnte beispielsweise auch ein sprach-basiertes Dialogsteuerungssystem verstanden werden. Die Betrachtungen im Projekt beschränken sich jedoch auf grafische Benutzerschnittstellen.

[3] Eine grundsätzliche Eigenschaft von Modellen (nicht nur in der Informatik) ist, dass sie einen auf den Kontext des Anwendungsfalls (vereinfachten) Ausschnitt der Realität modellieren (vgl. [**?**]). Je weiter das Szenario gefasst wird, desto allgemeiner muss auch das Datenmodell formuliert werden. Die Einschränkung auf das angegebene Szenario gestattet eine Vereinfachung des Modells, das expliziter auf den Anwendungsfall zugeschnitten werden kann.

Diese Zusammenhänge zwischen den einzelnen Datenfeldern/ Gruppen müssen im Datenmodell abgebildet werden können, damit später eine Transformation der Daten in die geforderten Dialogeinheiten erfolgen kann.

Für die *typabhängige Darstellung* in einer Dialogschnittstelle wird zusätzlich der Typ jedes Datums benötigt. Dieser ist auch für die Umsetzung weiterer Anforderungen von Bedeutung (z.B. Validierung).

Das Ausgangsmodell ist nicht nur für die Herleitung einer Oberfläche gedacht, sondern findet auch an anderen Stellen in der Applikation Anwendung. Ein Beispiel hierfür ist die serverseitige Validierung vom Client übermittelter Daten (z.B. Konsistenz, Vollständigkeit). Um das Modell Anwendungsübergreifend nutzen zu können, muss jedes Datum im Modell eindeutig identifiziert werden können. Es muss die *Bindung* an das Datenmodell beschreiben werden können, damit ein Feld in der Dialogschnittstelle mit einem Modellelement assoziiert werden kann.

Auf (Teil-)Bereichen des Modells können Operationen ausgeführt werden. In einer grafischen Benutzerschnittstelle wären dies z.B. Buttons, denen eine Funktionalität hinterlegt ist, die auf dem (Teil-)Modell operieren. Um eine Beutzerschnittstelle dafür bereitzustellen, müssen die möglichen Operationen angegeben werden können.

Diese Anforderungen aus dem Dialogschnittstellen-Bereich können zur Herleitung einer Variante herangezogen werden. Jedoch ist dabei noch keine kanalspezifische Information enthalten.

6.2 Varianten für einzelne Kanäle

Zur Herleitung kanalspezifischer Varianten müssen die Spezifika beschrieben werden können. Diese liegen grundsätzlich in der Beschreibung der Teilmenge der Daten, die für die Nutzergruppe relevant ist und weiteren Constraints, die in den Auswahlprozess mit einfließen (z.B. den Detaillierungsgrad der darzustellenden Informationen, die ein technischer Kanal ”mobile” mit sich bringt).

Um schnell und kostengünstig neue Kanäle durch die Anwendung behandeln zu können, bedarf es einer Möglichkeit, die für den Kanal relevante Untermenge des Modells mit einfachen Mitteln zu bestimmen.

Hierbei sind Lösungen dafür zu finden, wie diese Untermengen abhängig vom gewählten technischen oder fachlichen Kanal effizient definiert werden können (explizit durch Angabe der Elemente, implizit durch Regeln wie z.B. der Wichtigkeit für ein Datum). Hierbei können ganze Gruppen von Informationen betroffen sein oder einzelne Elemente des Datenmodells.

Hierzu werden Konzepte im Modell / Metamodell beschrieben, die eine solche Angabe ermöglichen.

6.3 Zu verwaltende Artefakte und Konsistenz

Um den Einsatz des Konzeptes zu vereinfachen, soll die Anzahl der Stellen im Modell minimiert werden, an denen es konsistent gehalten werden muss. Dies lässt sich erreichen, wenn das Ausgangsmodell kompakt gehalten und nicht über viele Dateien verteilt wird.

??? Wo kommt das her ??? Trotzdem soll ein Modell nicht Aspekte vermischen. So ist es nicht angebracht, beispielsweise in einem Datenmodell konkrete Visualisierungsinformationen zu verankern (z.B. Pixelangaben). Dennoch können Informationen qualitativ und auf die Daten bezogen angegeben werden, aus denen solche spezifischen Informationen in einem späteren Schritt abgeleitet werden können.

Es kann beispielsweise einer Postleitzahl der Typ *postleitzahl* gegeben werden, woraus später bei der

Transformation in eine grafische Repräsentation in einer Abbildungsvorschrift hinterlegt werden kann, dass ein Feld diesen Typs 20 Pixel breit sein soll. Auf diese Weise sind diese Informationen implizit enthalten, das Modell ist aber dennoch abstrakt gehalten.

Im Rahmen des Projektes soll die Zahl der Artefakte so gering wie möglich gehalten werden. Es ist abzuwägen, wie weitgehend dies erreicht werden kann.

6.4 Kontext und dessen Spezifika

Anmerkung: Dieser Abschnitt ist temporär hier angesiedelt. Ggf. muss eine Verschiebung in *mimesis.ui* erfolgen, wenn sich ein stärkerer Bezug zur Oberfläche herausstellen sollte.

Die Dialogschnittstellen, die aus dem Modell hergeleitet werden sollen, unterscheiden sich in ihren Anforderungen hinsichtlich des Aussehens und des Verhaltens. Zur Umsetzung dieser Aspekte in einer Dialogbeschreibung sind Informationen notwendig, die teilweise abhängig vom Kontext der Dialogschnittstelle oder vom Einsatz der Anwendung abhängig sind.

Hier können sollen exmplarisch zwei Arten von Kontextabhängigkeiten unterschieden werden[4]:

- **Dialogschnittstellen-Kontext:** variable Metainformationen, die für bestimmte Optionen bei Dialogschnittstellen relevant sind.
- **Einsatz-Kontext des Modells:** variable Metainformationen, die vom Einsatz-Umfeld des Modells wie z.B. die Lokation des Nutzers abhängen.

Im letzten Abschnitt wurden die Anforderungen aus der Herleitung einer Dialogschnittstelle betrachtet. Diese sind aus der Analyse eines konkreten Anwendungsfalls entstanden, nämlich der Herleitung von grafischen Benutzeroberflächen. Auch wenn diese mittels der in *mimesis.ui* entwickelten Dialogschnittstellen-Beschreibung technologieneutral formuliert wurden, hängen sie doch vom Kontext *grafische Benutzeroberflächen* ab.

Ein Beispiel hierfür ist der Themenbereich *syntaktische Validierung einer Eingabe*. Dieser benötigt Regeln nach denen eine Eingabe geprüft wird. In der in *mimesis.ui* beschriebenen Umsetzung des Konzeptes wurde hierzu ein regulärer Ausdruck verwendet, anhand dessen die Validierung erfolgen kann.

Die Angabe eines regulären Ausdrucks ist jedoch in einem anderen Kontext nicht sinnvoll. Eine Beispiel hierfür wäre die Eingabe eines Wertes per Sprachsteuerung. Die Prüfung auf einen regulären Ausdruck ist hier obsolet oder sind ggf. andere Mechanismen notwendig, die alternativ zu einer Validierung durch reguläre Ausdrücke einzusetzen sind.

Ein Beispiel für einen Einsatz-Kontext wäre die Nutzung einer Anwendung in Deutschland versus dem Einsatz in den Vereinigten Staaten. In einem solchen Fall würden beispielsweise bei der Darstellung und Prüfung der Postleitzahl andere Regeln gelten. Das Modell müsste dies bei der Herleitung der Dialogschnittstelle berücksichtigen und ggf. ein anderes ”Set” an Informationen für die Herleitung der Dialogschnittstellen verwenden.

Es muss möglich sein, die Informationen für die spezifischen Aspekte und variable Aspekte hinzuzumischen. Hierfür ist ein Konzept vorzusehen. Das Modell an sich ist von solchen Spezifika freizuhalten.

[4] Die Liste der Kontextarten kann sich zuküftig noch erweitern. Die gefundenen Arten sind exemplarisch zu sehen und entsprechen dem „aktuellen Stand“.

7 Lösungsansatz in *mimesis.model*

7.1 Überblick und Einordnung des Konzeptes in *mimesis.application*

Der Ansatz, der im Rahmen der Arbeiten verfolgt wird, basiert darauf, das Datenmodell der Anwendung um Metainformationen zu erweitern, die es in einem weiteren Schritt gestatten, eine technologieneutrale aber kanalspezifische Seitenbeschreibung herzuleiten. Diese Seitenbeschreibung wird in einem letzten Schritt in eine Maske transformiert, welche für ein bestimmtest Gerät darstellbar und sinnvoll ist. In Abbildung 1 ist dies dargestellt.

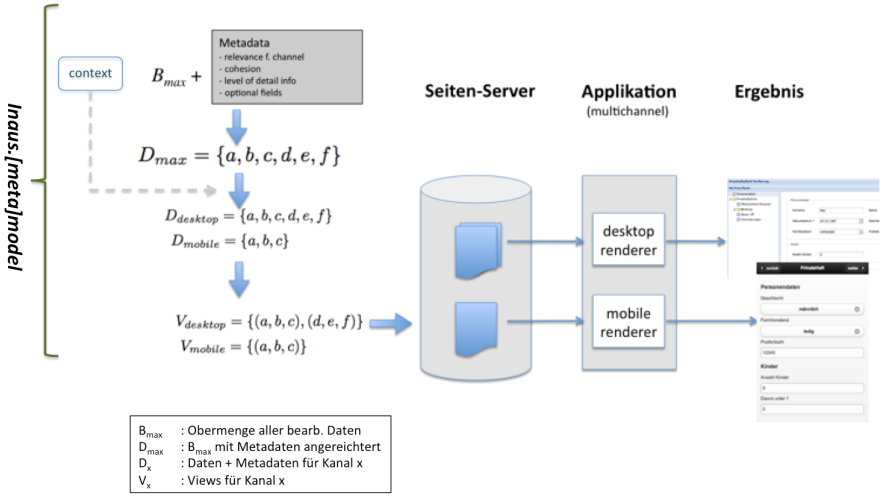

Abbildung 1: Lösungsansatz im Projekt mimesis

Der Bereich, der im *mimesis.model* behandelt wird, ist in der Abbildung markiert. Es handelt sich hierbei um die Definition des Datenmodells mit den benötigten Metainformationen, die Ableitung von kanalspezifischen Teilmengen und deren Transformation in eine Dialogschnittstellen-Beschreibung.

7.2 Lösungsansatz zur Definition einer Modellbeschreibung in *mimesis.model*

Das Ziel des Konzeptes in *mimesis.application* als übergeordnetes Projekt ist es, für einen Anwendungsfall eine einzige Applikation zu erstellen, die für alle die Anwendung nutzenden Kanäle verwendet werden kann und deren Varianten in Umfang der gesammelten Daten, den Abläufen und den möglichen Benutzerschnittstellen abdeckt.

Zur Annäherung an die Problemstellung, ein einheitliches Modell für alle Varianten in einer multikanalfähigen Anwendung zu finden, kann von den Abläufen ausgegangen werden. Im Rahmen der hier betrachteten Anwendungen wird davon ausgegangen, dass es Basisabläufe gibt, die für den jeweils unterstützen Kanal leicht variiert werden (siehe hierzu die Betrachtungen in *mimesis.flow*). Jeder Schritt in einem solchen Ablauf arbeitet auf einer Menge von Daten. In den Ablauf-Varianten werden Schritte hinzukommen oder entfernt. Zudem können die verarbeiteten Daten eines jeden Schritts je nach nutzendem Kanal im Umfang variieren. So lässt sich für jede Ablaufvariante die Menge an verarbeiteten

Daten als Vereinigung der Daten beschreiben, die jeder Schritt verarbeitet.

Um ein Datenmodell für eine multikanalfähige Anwendung zu erhalten, müssen die Daten bestimmt werden, die über alle Varianten der Anwendung hinweg verarbeitet werden. Die Ausgangsbasis für das Datenmodell bildet also in der *Vereinigungsmenge* aller Daten, die in den einzelnen kanalspezifischen Anwendungsvarianten verarbeitet werden sollen. In Abbildung 1 ist diese Basismenge mit B_max bezeichnet.

Die in B_max definierten Elemente müssen nun mit Metainformationen versehen werden, die eine detailliertere Beschreibung der Elemente vornehmen. Diese Metainformationen müssen dazu geeignet sein, einerseits die Zuordnung der für einen bestimmten Kanal relevanten Daten vornehmen zu können, andererseits die Erzeugung einer Dialogschnittstellenbeschreibung zu ermöglichen.

Die Erweiterung von B_max um die Metainformationen ist in Abbildung 1 als Menge D_max bezeichnet. Die in dieser Menge enthaltenen Elemente sind Tupel aus dem Datenfeld aus B_max und den für dieses Element definierten Metainformationen.

D_max stellt nun die Grundlage für die weiteren Transformationsschritte dar, die zu einer Dialogbeschreibung führen. Der in Abbildung 1 folgende Schritt dient der Bestimmung der Untermengen für konkrete Kanäle. Hierzu werden die Metainformationen herangezogen, die eine Auswahl der Daten ermöglichen (z.B. Relevanz für einen fachlichen Kanal, Detaillierungsgrad (level of detail) auf dem der Kanal arbeitet und damit Bestimmung von Feldern, die optional sind und bei Bedarf weggelassen werden können).

In Abbildung 1 wurden als Kanäle exemplarisch die technischen Kanäle *desktop* und *mobile* verwendet. Die aus dem Auswahlschritt resultierenden Teilmengen sind entsprechend mit $D\_desktop$ und D_mobile bezeichnet und enthalten alle Felder aus D_max mit ihren Metainformationen, die für den entsprechenden Kanal aufgrund ihrer Metainformationen als relevant betrachtet wurden.

Der nächste Schritt besteht nun in einer Transformation der Teilmennge in eine Dialogschnittstellen-Beschreibung, welche die Felder *sinnvoll* gruppiert und so für den späteren Anwender der Applikation aufbereitet. Hierzu muss das Modell Metainformationen bereitstellen, die eine solche Gruppierung möglich machen. Die hier relevanten Informationen sind insbesondere diejenigen, die die Beziehungen der einzelnen Datenelemente zueinander beschreiben und so die Bindung der Elemente aneinander ausdrücken. Diese Information kann zu einer Aufteilung der Daten auf Dialogeinheiten / Seiten verwendet werden.

Die hieraus resultierenden Mengen sind für die exemplarisch betrachteten Kanäle in Abbildung 1 mit $V\_desktop$ und V_mobile bezeichnet. Diese Mengen bestehen aus Teilmengen der Ausgangsmengen $D\_desktop$ und D_mobile, welche die enthaltenen Elemente sinnvoll gruppieren.

Der letzte Schritt besteht nun in der Transformation dieser Mengen V_x in die Dialogschnittstellenbeschreibungen in Form der *mimesis*-Dateien für den jeweiligen Kanal. Für die Transformation der einzelnen Elemente werden nun weitere Informationen aus dem Metamodell verwendet, welche die Ausgabe gemäß den Anforderungen der *mimesis.ui*-Beschreibung erfüllen.

Die weiteren Schritte sind in *mimesis.ui* angesiedelt und nicht mehr Teil des hier betrachteten Themenbereichs.

7.3 Umsetzungskonzept für die Transformation von *mimesis.model* in *mimesis.ui*

In Abbildung 2 ist das Vorgehen dargestellt, welches aus einer Modellbeschreibung die kanalspezifischen Dialogschnittstellenbescheibungen erzeugt.

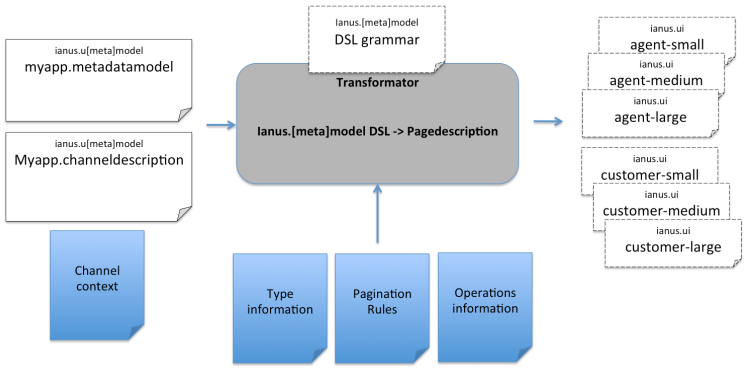

Abbildung 2: Lösungskonzept für die Umsetzung von mimesis.mode

Die Grundlage für die Transformation bildet eine Modellbeschreibung mit den enthaltenen Metainformationen. Im letzten Abschnitt in Abbildung 1 wurde die der Beschreibung zugrunde liegende Menge mit D_max bezeichnet. Sie enthält die Datenfelder und deren Metainformationen und wird in einer eigenen *domain specific language* (DSL) analog der Vorgehensweise im Themenbereich *mimesis.ui* beschrieben. Diese DSL wird in den folgenden Abschnitten hergeleitet.

In Abbildung 2 ist als Input neben der Modellbeschreibung (*myapp.metadatamodel*) getrennt die Beschreibung der Kanäle (*myapp.channeldescription*) als Eingabe aufgeführt. Es ist an dieser Stelle sinnvoll, die Metainformationen für die Kanalbeschreibung von der des Datenmodells zu trennen[5], da sich das Datenmodell selbst weniger häufig ändern wird als die Kanaldefinitionen. Eine gesonderte Definition hält die Modellbeschreibung der Daten frei von Kanalkonzepten.

Der Transformator für die *mimesis.model*-Beschreibung führt nun die im letzten Abschnitt beschriebenen Schritte (Auswahl der kanalrelevanten Daten, Verteilung der Daten auf Dialogeinheiten und Ausgabe einer Dialogschnitstellenbeschreibung) durch. Dabei werden kontextspezifische Informationen z.B. hinsichtlich des Typs der Daten, weitere kanalspezifische Einstellungen (Z.B. Regeln zur Verteilung der Daten auf Seiten) oder die Informationen zu Operationen *zugemischt*.

Die Ausgabe des Transformators besteht nun in der technologieneutralen Dialogschnittstellenbeschreibung für die definierten Kanäle, die von mimesis.ui weiterverarbeitet werden kann.

8 Multikanalfähige Modellbeschreibung in mimesis.model

Ziel dieses Abschnitts ist die Herleitung und Beschreibung der in mimesis.modell verwendeten Modellbeschreibung. Es werden die verwendeten Konzepte und Umsetzung entlang der Anforderungen beschrieben, die zu Beginn formuliert wurden oder sich als Folge ergeben haben.

5

8.1 Grundlegende Entscheidungen zum Aufbau der Modellbeschreibung

Im Rahmen des Projektes wurde zur Beschreibung des Modells eine eigene Syntax verwendet, die sich nicht an bestehenden Konzepten oder Sprachen orientiert. Der Grund dafür liegt darin, dass die Sprache dadurch auf das wesentliche reduziert werden kann und nicht von den Rahmenbedingungen universeller, ggf. mächtigerer aber auch komplexerer Sprachen abhängt.

Ein Beispiel hierfür wäre die Verwendung von XML oder JSON, welche zur Befriedigung der Syntax und Einhaltung der Struktur bereits viel redundante Zeichen benötigen und die eine Struktur vorgeben, die im gegebenen Kontext nicht notwendig ist. Diese Sprachen sind sehr flexibel und lassen sich sehr gut in bestimmten Situationen einsetzen (z.B. JSON im client-bereich zur einfachen Erzeugung von JavaScript-Objekten oder XML zur Validierung von Wertebereichen). Diese Vorteile können aber in unserem Kontext an dieser Stelle nicht genutzt werden.

Es ist eine Grundmotivation bei der Verwendung von *domian specific languages*, eine eigene, auf das wesentliche reduzierte Sprache für einen Problembereich anzubieten [Fow10]. Hierdurch kann die Sprache auf das wesentliche reduziert und im vorliegenden Fall eine einfach zu wartende Lösung gefunden werden.

Sollte sich im Verlauf des Projektes ergeben, dass bereits geeignete Sprachen in einer einsetzbaren Form existieren, so kann darauf ausgewichen werden. Die Eignung kann durch die Abbildbarkeit der verwendeten Konzepte bestimmt werden.

8.2 Aufbau einer Modellbeschreibung zur automatischen Herleitung von Dialogschnittstellen

Im Anforderungskapitel wurden bereits Problemstellungen identifiziert, die für eine automatische Herleitung gelöst werden müssen und aus den Anforderungen abgeleitet sind, die durch die Dialogschnittstellen-Generierung herrühren. Die Dialogschnittstellenbeschreibung benötigt insbesondere die Informationen für folgende Aufgabenstellungen:

- Aufteilung in Dialogeinheiten + Navigation
- Ein-/Ausgabekomponenten für Daten bestimmten Typs
- Validierung
- Sichtbarkeitssteuerung
- Dynamische Inhalte
- Datenhaltung, Bindung, Initialisierung
- Aktionen

In den folgenden Abschnitten werden die Konzepte zur Umsetzung dieser Aufgaben betrachtet.

8.2.1 Konzepte zur Modellierung semantischer Zusammenhänge zwischen Daten

Zusammenhänge zwischen Daten Zur Herleitung einer Dialogbeschreibung - insbesondere zur Bestimmung der Reihenfolge, Gruppierung und der Verteilung der abzufragenden Daten müssen aus der Modellbeschreibung Beziehungsinformationen abgeleitet werden können.

In einem Modell, das in vielen Anwendungsszenarien verwendet werden soll, ist es nur schwer möglich, anhand des Modells eine Reihenfolge herzuleiten, in welcher Daten erfragt werden sollen.

Die Fokussierung auf den Anwendungsfall der datensammelnden Anwendungen jedoch kann das Modell so beschrieben werden, dass auf eine explizite Modellierung der Visualisierung verzichtet werden kann. Hierzu können Eigenschaften verwendet werden, welche die Zusammenhänge zwischen den Daten qualitativ beschreiben und eine entsprechende Modellierung erfolgen.

Die grundsätzlich sich stellende Frage lautet: *Gibt es bereits inhärente semantische Zusammenhänge zwischen Datenfeldern, die für die Strukturierung von Daten verwendet werden können?*

Es gibt semantische Zusammenhänge, welche bei der Modellierung verwendet werden können:

- **Objekteigenschaft**: Daten von Objekten unserer Umwelt bilden per se eine Gruppierung von Daten
- **Kohäsion der Daten**: Es kann für Daten angegeben werden, wie stark der Zusammenhalt zu anderen Daten ist.
- **starke Bildung von Daten**: In manchen Fällen bilden Daten eine starke Einheit und sind als solche zu behandeln.
- **logische Reihenfolge von Daten**: Daten einer Sinneinheit besitzen oft eine semantisch logische Reihenfolge.

Betrachtet man den Beriech der objektorientierten Modellierung, finden sich hier bereits die ersten offensichtlichen Ansatzpunkte, die eine **Gruppierung von Daten** betreffen. Objekte werden hier bereits als Gruppen von Daten modelliert und in Beziehung zueinander gesetzt.

So kann beispielsweise eine *Person* aus *Name, Vorname, Geburtsdatum* und eine *Adresse* aus *Straße, Hausnummer, Postleitzahl* und *Ort* bestehend modelliert werden. Zudem kann eine Beziehung zwischen Person und Adresse angegeben werden (z.B. Person hat eine *private* und eine *geschäftliche Adresse*). In Abbildung 3 ist ein solcher Zusammenhang dargestellt.

Abbildung 3: Gruppierung anhand einer Objekteigenschaft

Unter der **Kohäsion** (Zusammenhalt) kann verstanden werden, wie eng Daten in Objekten oder Objekte miteinander in Beziehung stehen. Objekte, die enger miteinander verbunden sind werden meist auch in einer Dialogschnittstelle räumlich näher abgefragt.

Ein Beispiel ist in Abbildung 4 dargestellt. So besitzen die Felder *Straße / Nr.* einen stärkeren Zusammenhalt mit *PLZ/Ort* als mit *Land*. Zudem besitzen alle im Adressbereich befindlichen Felder eine stärkere Kohäsion zueinander als beispielsweise zum *Vornamen* der Adresse.

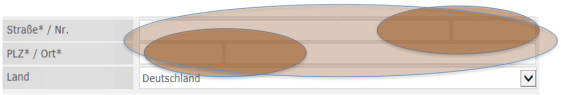

Abbildung 4: Anordnung anhand der Kohäsion

Eine noch stärkere Version der Kohäsion ist die hier mit **enger Bindung** bezeichnete Zusammenhalt. Sie ist hier gesondert aufgeführt, da sie auch an der Oberfläche eine gesonderet Repräsentation erfordert. Ein Beispiel hierfür ist ebenfalls in Abbildung 4 dargestellt. Zwischen *PLZ* und *Ort* besteht eine solche enge Bindung. Die Felder hängen so stark zusammen, dass sie in Oberflächen üblicherweise nebeneinander dargestellt werden. Analog verhält es sich mit *Straße* und *Hausnummer*. Die enge Bindung beschreibt Einheiten, die grundsätzlich nicht getrennt werden sollten.

Innerhalb von Sinngruppen lässt sich häufig eine **logische Reihenfolge** der Daten angeben, die aus Erfahrungswerten oder Konventionen stammt. So wird bei der Angabe einer Adresse üblicherweise die *Straße* und *Hausnummer* vor der Angabe von *Postleitzahl* und *Ort* erfolgen.

Person und *Adresse* sind im objektorientierten Sinne Klassen; in der prozeduralen Programmierung könnte man hieraus (unter Vernachlässigung der Methoden) auch abstrakte *Datentypen* daraus erstellen. In einem Anwendungkontext werden hiervon Ausprägungen verwendet (Instanzen), die im Kontext der Anwendung eine **Rolle** übernehmen. So wäre eine Rolle vom *Typ* Person die *versicherte Person* oder der *Versicherungsnehmer*. Dieser hätte ggf. zwei Adressen: eine *Privatanschrift* und eine *Geschäftsanschrift*.

Diese Zusammenhänge kann man im Rahmen einer Anwendung, die diese Daten erfassen soll, auch hierarchisch darstellen[6]. In Abbildung 5 ist dies dargestellt. Hier sind die Anschriften Teil der versicherten Person.

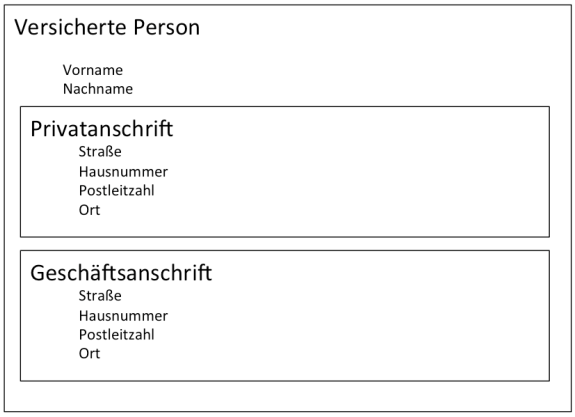

Abbildung 5: Hierarchische Repräsentation

[6]Dies ist dann sinnvoll, wenn die Daten nicht in einem größeren Kontext eingesetzt und referenziert werden sollen - wie dies in einem generell einsetzbaren Modell üblicherweise der Fall ist. Im betrachteten Szenario der datensammelnden Anwendung werden die Daten jedoch lediglich zur Erfassung dargestellt und können damit ”in Sito” an der Stelle erfasst werden, an der sie sinnvollerweise in der Hierarchie stehen.

Modellierung der Zusammenhänge Die Modellierung dieser Zusammenhänge könnte nun erfolgen, indem man die Beziehungen eines Datums zu anderen Daten explizit angibt. Hierbei müsste die Kohäsion für jedes Feld definiert und verwaltet werden.

Da im vorliegenden Anwendungsfall die Daten hierarchisch dargestellt werden können, kann ein einfacherer Weg gewählt und die Gruppierung und der Zusammenhalt hierarchisch repräsentiert werden.

Die **Gruppierung** der Informationen in der *mimesis.model*-Beschreibung erfolgt durch die Bildung von Blöcken. Soll innerhalb eines Blocks zwischen Elementen eine stärkere Kohäsion dargestellt werden, so werden diese Elemente wieder in einem Block zusammengefasst (vgl. (1) in Abbildung 6). Eine enge Bindung, wie sie beispielsweise zwischen *Straße* und *Hausnummer* besteht, wird durch eine besondere Auszeichnung hervorgehoben, sodass erkannt werden kann, dass diese Felder eine Einheit bilden.

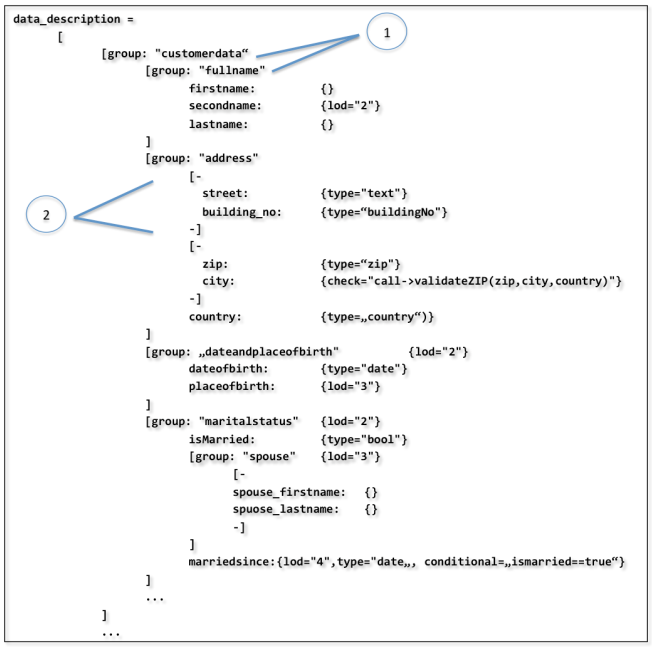

Abbildung 6: Hierarchische Struktur zur Gruppierung in mimesis.model

Die Reihenfolge der Elemente ist durch die Reihenfolge in dieser Beschreibung festgelegt. Da die Elemente innerhalb eines Blocks eine stärkere Bindung zueinander als zu außenliegenden Elementen besitzen, kann diese Reihenfolge auch nicht gestört werden und es entstehen sinnvolle Einheiten. Innerhalb eines Blocks haben alle Elemente eine Kohäsion gleicher Stärke. Die Reihenfolge der Elemente

Nutzen für die Herleitung der Dialogbeschreibung Diese Strukturierung des Modells ist Teil der Metainformationen, welche die Modellbeschreibung mit sich führt. Der Informationsgehalt liegt in der Strukturierung der Daten. Diese Informationen könnten zwar auch als Attribute an jedem einzelnen Datenelement als Metainformationen angebracht werden, dies würde allerdings sehr unübersichtich werden und die Wartbarkeit drastisch einschränken. Die Gruppierung hat hier deutliche Vorteile (wird ggf. in zukünftigen Veröffentlichungen weiter ausgeführt - der Nachweis wird hier als offensichtlich angenommen :-)).

Bei der Transformation ist diese Aufbereitung insbesondere für die Verteilung der Informationen auf Dialogeinheiten von großem Nutzen. Die Gruppierung der Informationen und die damit verbundene Information, wie eng die Daten miteinander verbunden sind, kann in Kombination mit Regeln für die Aufteilung der Informationsblöcke auf Dialogeinheiten direkt genutzt werden.

So wird beispielsweise eine Regel des Transformators für Geräte mit kleinen Darstellungsoberflächen (z.B. mobile Endgeräte) besagen, dass pro Seite max. 15(+/- 2) Eingabefelder sinnvoll sind. Der Transformator kann nun bei der Verteilung auf die Seiten entscheiden, welche Blöcke voneinander getrennt werden können ohne deren inneren Zusammenhänge zu zerstören.

8.2.2 Identifikation von Elementen im Modell

Die Dialogschnittstellen arbeiten auf einem lokalen Modell, in welchem die eingegebenen Daten gehalten werden müssen. Dort gibt einen eindeutigen Bezug von einem dort lokalen Modellelement zu einem Eingabefeld (im Falle einer grafischen Benutzeroberfläche).

Das hier zu definierende Modell wird jedoch nicht nur zur Herleitung einer Dialogschnittstellenbeschreibung verwendet, sondern beschreibt das Datenmodell der ganzen Anwendung und wird auch auf Prozess/Workflowebene eingesetzt werden - beispielsweise zur Identifikatin der Daten die vom Client zum Server übertragen werden.

Daraus erwächst die Anforderung, dass jedes Element im Modell eindeutig identifizierbar sein muss um eine über alle aus dem Modell generierten Artefakte einheitliche Referenzierung ermöglichen zu können.

Die im letzten Abschnitt eingeführte hierarchische Definition des Modells gestattet eine einfache Lösung für diese Anforderung, die sich an bereits verbreiteten und erprobten Mechanismen bedient. So wird bereits in JEE-Technologien (Java Enterprise Edition) zur Kennzeichnung der Bindung von Oberflächenelementen an das Datenmodell eine Notation verwendet, die hier ebenfalls nutzbringend eingesetzt werden kann. In Java Server Faces (JSF) wird die Unified Expression Language (UEL) verwendet, welche die Elemente des Objektmodells einer Anwendung addressierbar macht.

Um ein Modellelement in der Modellbeschreibung zu adressieren, wird im Rahmen des Projektes eine Punkt-Notation verwendet analog der in der UEL definierten verwendet.

Hierzu müssen allen Gruppen und Elementen eindeutige Namen vergeben werden. Jedes Feld hat ohnehin einen Bezeichner - und die definierten Gruppen, die sinnvolle Einheiten bilden, besitzen ebenfalls einen eindeutigen Namen (vgl. Abbildung 6), die sie aus der Rolle beziehen, die der Block im Rahmen des Modells spielt.

Diese Namen werden zur Adressierung eines Datums durch einen Punkt getrennt aneinandergehängt.

Anhand des Modell-Ausschnittes aus Abbildung 6 wäre die Adressierung des Vor- und Nachnamens des Ehegattens (spouse_firstname, spouse_lastname) damit

```
customerdata.maritalstatus.spouse.spouse\_firstname
bzw.
customerdata.maritalstatus.spouse.spouse\_lastname
```

Würde die Adresse als Block refernziert werden, so würde die Angabe

```
customerdata.address
```

als eindeutiger Schlüssel zur Adressierung verwendet. Diese Schlüssel können durchgängig verwendet werden.

8.2.3 Elementinformationen

Die Metainformationen, die für ein einzelnes Element benötigt werden, können direkt bei der Definition des Elements angegeben werden. Diese Informationen beschreiben das Element näher und geben dem Transformator hinweise, wie das Element zu behandeln ist.

In den vorangegangenen Betrachtungen wurde Bedarf an folgenden Informationen festgehalten, die sich aus den Anforderungen der Herleitung der Dialogschnittstellenbeschreibung ergaben:

- Typ des Datums
- Validierungsregeln
- Datenbeschaffung und Initialisierung

Der Typ des Datums ist dabei eine zentrale Angabe, aus der eine Reihe von weiteren Informationen für den Transformator herleitbar sind. So kann der Transformator schon aufgrund der Typinformationen syntaktische Validierungsregeln festlegen, die für alle Elemente diesen Typs gelten (z.B. E-Mail-Syntaxvalidierung) oder die Anzahl möglicher Zeichen bestimmen (Beispiel: Potleitzahl besteht aus 5 Zeichen). All diese Informationen können bei einer Transformation hinzugemischt werden.

Zusätzliche Validierungen oder semantische Validierungen, für die die Ausführung von Programmcode nötig wird, müssen jedoch weiterhin am Element selbst ausgegeben werden.

An dieser Stelle kann keine erschöpfende Liste möglicher Werte angegeben werden. Lediglich die verwendete Notation für diese Werte.

Zur Definition des Typs und weiterer Metainformationen befindet sich hinter der Bezeichnung des Datenfeldes ein durch geschweifte Klammern angegebener Bereich, in welchem die Metainformationen in Form von Name/Wert-Paaren definiert werden können.

In Abbildung 7 ist dieser Bereich am Beispiel für Typ-Infomrationen und Validierungen markiert (siehe (1), (2)). Auch Initialwerte und mögliche Auswahloptionen können aus diese Weise festgelegt werden.

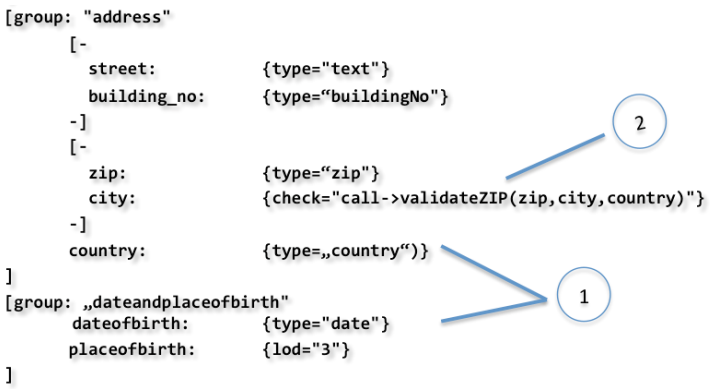

Abbildung 7: Metainformationen als Elementinformation

Die Datenbeschaffung für ein Feld, sofern Programmcode dafür nötig wird, erfolgt analog dem Aufruf von Validierungscode. Hierzu wird eine Operation/Aktion angegeben, deren ergebnis mögliche Daten beschafft oder herleitet.

8.2.4 Existenzabhängigkeiten zwischen Daten

Ein hier noch nicht näher beleuchteter Punkt ist die Abfrage von Daten abhängig von der Existenz anderer Felder oder Feldinhalte. Ein Beispiel hierfür ist die Angabe eines Vor- und Nachnamens für einen Ehepartner. Diese Abfrage ist nur sinnvoll, wenn der Befragte auch verheiratet ist.

Es existieren Bereiche im Datenmodell, die nur relevant sind, wenn bestimmte Bedingungen erfüllt sich, die auf anderen Daten des Modells beruhen.

An einer grafischen Benutzerschnittstelle wird diese Information benötigt, um Felder anzuzeigen oder ggf. zu aktivieren (sichtbarkeitssteuerung). Der Hintergrund ist aber modellgetrieben und damit bereits im Modell zu beschreiben.

Der Weg, der im Rahmen des Projektes zur Beschreibung dieser Zusammenhänge verwendet wird, ist ebenfalls über die Angabe von Informationen in den Metainformationen eines betroffenen Feldes (bzw. einer Gruppe). Hier kann durch Angabe von Bedingungen, die auf Feldern des Datenmodells definiert sind, bestimmt werden, ob ein Datum oder eine Gruppe zum aktuellen Zeitpunkt relevant ist.

In Abbildung 8 ist dies am Beispiel des Familienstandes dargestellt. Hierbei werden Bedinungen für eine Feld und eine Gruppe definiert (*conditional*), die auf Angaben im Modell beruhen. So wird das Feld *marriedsince* nur relevant, wenn *isMarried* den Wert *true* besitzt und die Gruppe *spouse* erst relevant, wenn zu dieser Bedingung auch noch das Jahr der Heirat nach 2004 lag.

```
data_description =
      [
            [group: "customerdata“
                  ...
                  [group: "maritalstatus"
                        isMarried:          {type="bool"}
                        marriedsince:       {type="date“, conditional =„customerdata.maritalstatus.ismarried==true"}
                        [group: "spouse"    {conditional =„customerdata.maritalstatus.ismarried==true
                                                      AND customerdata.maritalstatus.marriedsince.Year>“2004“}
                              spouse_firstname:   {}
                              spuose_lastname:    {}
                        ]
                  ]
                  ...
            ]
            ...
```

Abbildung 8: Existenzabhängigkeit über Bedingungen

8.2.5 Aktionen

Dieses Thema ist aktuell noch nicht ausreichend bearbeitet. Eine Bearbeitung findet im Rahmen aktueller Arbeiten statt.

Der verfolgte Ansatz besteht darin, dass Aktionen für Gruppen oder Einzelelemente definiert werden können. Die Definition bei der Gruppe gestattet es bei der Generierung einer Dialogschnittstellenbeschreibung die Aktionsauslösenden Elemente (in einer grafischen Benutzeroberfläche z.B. ein Button) ebenfalls bei den Gruppen anbeiten zu können, die dann ggf. auf unterschiedliche Seiten verteilt sind. Die Angabe bei einem Element kann dazu genutzt werden, weiterführende Aktivitäten für die Eingabe des Elements zu steuern - in einer graphischen Benutzeroberfläche wäre dies beispielsweise die Anzeige eines modalen Dialogs, der zu einer Aktivitäten durchführt, die die Eingabe erleichtern (Beispiel: komplexe Auswahl von Broduktbausteinen, die in einem Produktbausteincode münden).

8.3 Kontext und dessen Spezifika

Wie in den Anforderungen als Anmerkung formuliert, besteht hier noch die Frage, ob diese Thematik nicht eher bei mimesis.ui anzusiedeln ist. Aus diesem Grund wird an dieser Stelle nicht näher darauf eingegangen. Ein Lösungsansatz könnte sein, Kontextspezifika bei den hinzuzumischenden Typprelevanten Eigenschaften (in Abbildung 2 als ”Typeinformation” bezeichnet) zu plazieren. Der Kontext könnte durch den Austausch der Typeinformation-Beschreibung erfolgen.

8.4 Definition von kanalspezifischen Varianten

Die zentrale Aufgabe zur Herleitung kanalspezifischer Varianten von Dialogbeschreibungen besteht in der Auswahl der für den Kanal relevanten Daten. Ziel ist es im ersten Schritt der beschriebenen Transformation die Teilmenge der in der Modellbeschreibung enthaltenen Daten zu bestimmen, die für einen Kanal relevant sind.

Eine Aufgabenstellung besteht darin, die *Granularität* der angezeigten Daten zu steuern. Die ist insbesondere für die Erzeugung von Dialogbeschreibungen für die technischen Kanäle, also für Ausgabegeräte mit Darstellungsflächen unterschiedlicher Größe, relevant. Dies kann zwar auch zur Bestimmung der Relevanz für fachliche Kanäle genutzt werden, genügt hier jedoch nicht, da in diesen Fällen ggf. ganze Teilbereiche des Modells irrelevant werden. Bei einer Kombination fachlicher mit technischen Kanäle (z.B. eine desktop/mobile Variante je für einen Vertreter und einen Endkunden) entstünden Vrainaten, die mit der einfachen Granularitäts-Bestimmung nicht mehr im Modell beschrieben werden können. Es ist damit notwendig, auch eine Bestimmung der Relevanz für ganze Bereiche des Modells für einen Kanal vornehmen zu können.

Zur Lösung der Aufgabenstellungen können mehrere Ansätze angewandt werden, die eine Auswahl gestatten und die im Folgenden kurz erläutert werden.

8.4.1 Level-of-detail-Konzept (LOD) für optionale Felder

Insbesondere bei technischen Kanälen, die eine beschränkte Anzeigefläche besitzen, besteht die Aufgabe darin, aus der Menge der Gesamtdaten eine reduzierte Teilmenge zu extrahieren, die trotzdem für den anschließenden Verarbeitungsschritt verarbeitbar ist.

Ein erster Ansatz für die Transformation wäre es, Daten zu ignorieren, die im Modell als *optional* gekennzeichnet sind. Problematisch ist hierbei, dass bei diesem Vorgehen entweder alle optionalen Felder als relevant oder irrelevant eingestuft werden müssen, da die Transformation keine Informationen über die *Wertigkeit* des Datums besitzt (Grad der Wichtigkeit).

Eine Lösung dieses Problems besteht in der Einführung eines *level-of-detail*-Konzeptes für Gruppen und Einzeldaten. Hierbei wird bei einem Element in seinen Metainformationen ein *level-of-detail*-Wert zugewiesen, der besagt, ab welchem Detaillierungsgrad das Feld berücksichtigt werden soll.

Die Transformation kann festlegen, welcher Detaillierungsgrad für den bearbeiteten Kanal relevant ist und anhand der Metainfomrationen für ein Feld entscheiden, ob es berücksichtigt werden soll.

Mittels dieses Ansatzes können nun Anwendungsvarianten erzeugt werden, die mehr oder minder detailliert sind und damit die Granularität gesteuert werden. Bei der Transformation kann ein konkreter Kanal einer Detaillierungsstufe zugeordnet und die passende Beschreibung erzeugt werden.

Dieser Ansatz löst allerdings lediglich die einleitend erwähnte Aufgabenstellung der Granularitätsteuerung.

Eine wahlfreie Auswahl von Modellbereichen für Kanäle ist hiermit nicht möglich.

8.4.2 Definition von Aspekten/Features im Modell und Zuordnung zu einem Kanal

Um nun Bereiche des Modells einzelnen Kanälen zuordnen zu können, können Bereiche des Modells als Aspekte bzw. Features einer Anwendung aufgefasst und modelliert werden.

Ein (bereits an anderer Stelle schon angeführtes) Beispiel aus dem Bereich Versicherung könnte sein, dass in einer Tarifierungsanwendung für eine Lebensversicherung in der Variante für einen Vertreter ein Fragenblock zu Gesundheitsfragen gestellt wird, der in der Variante für einen Kunden nicht vorhanden sein soll.

Der Bereich / die Gruppe des Modells, der die Eingaben zu den Gesundheitsfragen enthält, könnte als *Gesundheitsfragen-Feature* ausgezeichnet werden. Im Rahmen einer Kanal-Definition könnte nun für den Kanal *Vertreter* angegeben werden, dass dieses Feature hier anwendbar sein soll. Im Falle des *Kunden*-Kanals würde das Feature nicht zugeordnet.

Dieser Ansatz würde es nun gestatten, Modellbereiche explizit und nach fachlichen Kriterien zuzuordnen. Allerdings betrifft dies lediglich geschlossene Bereiche und alle darin enthaltenen Gruppen von Daten. Ein Ausschluss von im Feature enthaltenen Bereichen wäre so nicht möglich - fachlich aber durchaus denkbar.

8.4.3 Explizite Festlegung der Relevanz von Modellbereichen für Kanäle

Ein Ansatz, diesem Problem zu begegnen, besteht in der expliziten Definition der Kanalzugehörigkeit von Bereichen des Modells zu einem Kanal. Hierbei wird festgelegt, welcher Modellbereich für welchen Kanal relevant ist.

Der naive Ansatz könnte so aussehen, dass für jeden Bereich in den Metainformationen einer Gruppe oder eines Feldes vermerkt ist, für welchen Kanal er relevant ist. Dies würde dazu führen, dass für jeden neu hinzukommenden Kanal an jedem Modellelement vermerkt werden müsste, ob es für diesen Kanal relevant ist. Jedes Modellelement muss geprüft und eine Zuordung vorgenommen werden - dies ist sehr aufwending und Fehleranfällig, da die Kanalinformationen über das gesamte Modell verstreut sind. Das Modell ist zudem mit kanalspezifischen Informationen durchsetzt.

Ein weitaus geschickterer Ansatz ist es, neben dem Modell selbst eine gesonderte Beschreibung für die Kanäle zu erstellen. Diese Beschreibung enthält für jeden Kanal eine Definition, die bestimmt, welche Gruppen/Elemente für einen Kanal relevant sind und welche Gruppen / Elemente ggf. ausgeschlossen werden sollen. Hiermit ist eine wahlfreie Definition der Relevanz von Modellelementen zu einem Kanal möglich.

Ein Vorteil, den man zudem aus der in *mimesis.model* erfolgten hierarchischen Beschreibung des Modells ziehen kann, liegt in einer vereinfachten Beschreibung der Kanäle durch Annahmen.

So kann prinzipiell angenommen werden, dass für einen Kanal alle Daten relevant sind. Über eine Exklusions-Beschreibung können Bereiche des Modells ausgeschlossen werden (zur Identifikation der Bereiche steht der in einem vergangenen Abschnitt beschriebene Vorgehen über eine Punkt-Notation zur Verfügung). Sollten Teilbereiche eines ausgeschlossenen Bereichs wieder hinzugenommen werden, kann dies über eine Inklusions-Beschreibung erfolgen.

Über diesen Lösungsansatz sind alle Kombinationsmöglichkeiten abbildbar und mit geringem Aufwand zu definieren. Eine neue Kanalbeschreibung kann ohne großen Aufwand hinzugefügt werden. Die Definition erfolgt fokussiert auf einen Kanal und erfolgt zentral an einer Stelle der Modellbeschreibung.

8.4.4 Umsetzung der Varaintenbeschreibung in *mimesis.model*

Die Variantenbeschreibung in mimesis soll die Auswahl von Teilmodellen gestatten. Hierbei soll von vom Gesamtmodell ausgegangen werden und dabei Bereiche ausgeschlossen werden, die für die aktuell zu definierende Variante nicht relevant sind. Dieser Bereich ist noch nicht vollständig erarbeitet und wird in zukünftigen Berichten verfeinert.

Ein Beispiel, wie dieses Konzept zur Formulierung einer Kanalbeschreibung umgesetzt werden könnte, ist in Abbildung 9 dargestellt (mit Änderungen entnommen aus einer aktuell laufenden Studienarbeit, Hr. Ebsen) Hier werden drei Kanäle definiert: *customers*, *employees*, und *engineers*. Die Definition für den Kanal *customers* ist eine von anderen unabhängige Definition. Hier werden im ersten Schritt alle Bereiche des Modells ausgeschlossen. Danach folgt die Hinzunahme der Modellbereiche *customerData* und *orderData*. Die Definition des Kanals *employees* basiert auf der Definition von *customers* (extends). Hier wird der Modellbereich *paymentData* hinzugenommen, jedoch die Adresse im Bereich der Kundendaten als nicht relevant markiert (exculde *customerData.address*). Abschließend erfolgt die Definition für den Kanal *engineers*, die auf *employees* basiert aber keine Kundendaten benötigt.

```
channel_description : {

     channel : "customers" {
          exclude : “*“
          include : "customerData“,
          include : "orderData“
     },

     channel : "employees" {
          extends : "customers“,
          include : "paymentData“,
          exclude : "customerData.address"

     },

     channel : "engineers" {
          extends : "employees“,
          exclude : "customerData“
     }
}
```

Abbildung 9: Kanaldefintionen in mimesis.model

9 Ergebnisse und Stand der Arbeiten

Die Stand der Konzeption zur datenzentrierten Beschreibung von Anwendungsvarianten und der relevanten Modellelemente wurde in den vergangenen Abschnitten dargestellt. Im Zuge dieser ersten Schritte wurde hierbei die dort angegebene *provisorische* Syntax für die Modellbeschreibung verwendet - um eine Beschreibungsbasis für die Umsetzung der Ansätze aufzubauen. In dieser Form wird sie auch in ersten Veröffentlichungen verwendet. Die Beschreibung ist jedoch noch nicht als final zu sehen, da erst noch Erfahrungen mit der Umsetzung gesammelt werden müssen.

Aktuell existiert auf Basis dieser Beschreibung/Syntax eine Implementierung für die Transformation des Beschreibungsmodells in die Dialogschnittstellenbeschreibung, an welcher grundlegende technische Aspekte der Umsetzung evaluiert wurden (Formulierung einer Grammatik für die Sprache, Erzeugung eines Compilers mit ANTLR4, Verwendung von Templates zur Generierung der Dialogschnittstellenbeschreibung). Diese wird in den weiteren Schritten an neue Erkenntnisse angepasst.

10 Studentische Arbeiten zu mimesis

Aktuell (vorauss. Ende: Mai 2014) läuft eine Studienarbeit im Informatik-Masterstudiengang der DHBW-Stuttgart an (Studierender: Michael Ebsen, DEKRA), welche eine erste Umsetzung/Implementierung der Konzepte zum Ziel hat. Im Zuge der Arbeit wird in Zusammenarbeit mit dem Studierenden auch die konkrete Syntax für die Modell- und Variantenbeschreibung basierend auf den in diesem Papier dargestellten Konzepten festgelegt und prototypisch umgesetzt.

Eine Bachelorarbeit im Bereich *mimesis.application* soll zudem klären, ob eine generische, von einer Workflowengine getriebene Anwendung gebaut werden kann, welche die Koordination der Dialogschnittstelle mit dem serverseitigen Anwendungsteil untersuchen soll. (vorauss. Ende: Mai 2014, Studierender Wirtschaftsinformatik: Dennis Büssing, Allianz Deutschland AG). Diese Arbeit soll klären, ob der in mimesis verfolgte Ansatz auf workflowgetriebene Anwendungen erweitert werden soll.

11 Abgrenzung / Ausblicke / Offene Punkte

Die gewählte Modellbeschreibung wurde mit Blick auf die vollständig automatisierte Herleitung von Dialogschnittstellen entwickelt. Der gewählte Ansatz gestattet die Herleitung von Dialogschnittstellen zur Erfassung von Daten aus dem Modell für unterschiedliche Kanäle, wie sie insbesondere für die Kategorie der datensammelnden Anwendungen benötigt werden. Für diese Anwendungen kann angenommen werden, dass die zu erfassenden Daten semantisch zusammenhängend in Gruppen erfasst werden und dass eine verstreute bzw. beliebige Verteilung dieser Daten nicht erforderlich ist.

Hierin liegt auch eine grundsätzliche Einschränkung der automatischen Generierung der Dialogschnittstellen. Es ist mit diesem Modell ohne weitergehende Informationen nicht möglich, Eine Dialogschnittstelle herzuleiten, in welcher Datengruppen an beliebigen Stellen in der Dialogfolge erfragt werden können. Wird beispielsweise in der Dialogfolge für den Kanal *Endkunde* die Adresse der versicherten Person erst am Ende der Dialogfolge abgefragt, für einen Kanal *Vertreter* jedoch bereits zu Beginn, so kann dies nicht automatisch ohne weitere Angaben generiert werden.

Das hier dargestellte Modellbeschreibung ist jedoch prinzipiell unabhängig vom Dialogfluss verwendbar - aber für den ”Standardfall” generisch verwendbar. Es ist auch einsetzbar für die Erzeugung anderer als der automatisch herleitbaren Dialogflüsse einsetzbar, da hier lediglich Zusammenhänge beschrieben werden, die den Daten innewohnen und die zur automatischen Herleitung ausgenutzt werden. Unter Hinzunahme einer zusätzlichen Dialogfolge-Beschreibung könnte dasselbe Modell auch für andere, nicht dem Standardfall entsprechende Dialogfüsse verwendet werden.

Eine Dialogfolge-Beschreibung könnte basierend auf den Gruppen-/Elementnamen des Modells eine anderer Reihenfolge festlegen und unter Einführung eines In-/Exklusionskonzeptes (wie bei der Definition der Kanäle) eine freie Folge der erfragten Daten definieren.

Eine spezielle Dialogfolge-Beschreibung könnte als Ergänzung zum Standardverfahren definiert wer-

den. Soll eine solche alternative Dialogfolge für einen Kanal angewendet werden, kann dies bei der Definition des Kanals angegeben werden. Der Transformator könnte dann vom Standardverfahren abweichen und eine Dialogbeschreibung auf dem selben Modell aber basierend auf den expliziten Angaben in der Dialogfolgebeschreibung zurückgreifen, statt die impliziten Informationen der Modellbeschreibung zu verwenden.

Vorrangiges Ziel der Untersuchungen ist es nicht, eine allgemeine Lösung für alle erdenklichen Fälle zu erstellen, sondern den Fokus auf die häufigsten Fälle zu setzen. Aus diesem Grund wird dieser Ansatz aktuell nicht weiterverfolgt und als Ausblick behandelt.

Literatur

[CCT+02] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon und Jean Vanderdonckt. The CAMELEON Reference Framework. 2002.

[CCT+03] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon und Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308, 2003.

[Fow10] Martin Fowler. *Domain-specific languages*. Addison-Wesley Professional, 2010.

[Hit13a] Michael Hitz. Interner Projektbericht mimesis.ui. Bericht, DHBW-Stuttgart, Stuttgart, 2013.

[Hit13b] Michael Hitz. Konzept zur Variantenbildung von Oberflächen in einer Multikanal-Umgebung. In *Lecture Notes in Informatics INFORMATIK 2013 - GI-Jahrestagung 2013*, Seiten 2664–2678. GI, Köllen Druck+Verlag GmbH, Bonn., 2013.