Vom fachlichen Modell zum ausführbaren Workflow

Am Beispiel von ARIS und Intalio|BPMS

Prof. Dr. Thomas Allweyer

Fachhochschule Kaiserslautern

Februar 2008

Kontakt:

Prof. Dr. Thomas Allweyer
Fachhochschule Kaiserslautern
Fachbereich Informatik und Mikrosystemtechnik
Standort Zweibrücken
Amerikastr. 1
66482 Zweibrücken
Tel.: 06332/914-324
E-Mail: thomas.allweyer@fh-kl.de

Inhalt

1 Einleitung

Moderne *Workflow*- oder *Business Process Management*-Systeme (BPMS) ermöglichen die automatisierte Ausführung von Abläufen auf Grundlage von Modellen. Grafische Modelle erleichtern das Verständnis der implementierten Abläufe auch für Personen, die selbst nicht programmieren können. Die grafische Darstellung eines solchen ausführbaren *Workflow*-Modells sieht auf den ersten Blick sehr ähnlich aus wie das fachliche Modell eines zu unterstützenden Geschäftsprozesses. Teilweise schlagen die BPMS-Hersteller gar vor, dass die ausführbaren Prozessmodelle direkt von Fachanwendern erstellt werden sollen. Zumindest aber sollte dies die korrekte Umsetzung der fachlichen Anforderungen in die informationstechnische Lösung erleichtern.

Im vorliegenden Arbeitspapier wird die Umsetzung eines einfachen fachlichen Prozessmodells auf Basis von ereignisgesteuerten Prozessketten (EPK) in ein ausführbares Modell für das System Intalio|BPMS gezeigt. Hierbei werden nicht nur die Modellierung und Umsetzung der reinen Kontrollflusslogik beschrieben, sondern die komplette Definition aller für die spätere Ausführung des Prozesses erforderlichen Aspekte, wie z. B. die Rollenstruktur, Dialoge, Datenstrukturen und Datentransformationen.

Da es für eine derartige Umsetzung von fachlichen in ausführbare Modelle keine einheitliche Vorgehensweise gibt und diese zudem von den verwendeten Notationen und Systemen abhängt, handelt es sich bei dem dargestellten Vorgehensweise um einen anhand des Beispiels entwickelten ersten Vorschlag, keineswegs um eine fertig entwickelte Methodik.

Das verwendete Intalio|BPMS basiert zu einem großen Teil auf *Open Source*-Software-Komponenten, auch wenn es selbst nicht *Open Source* ist. Verwendet wurde die *Community Edition*, die lizenzkostenfrei erhältlich ist. Interessierte Leser können das vorgestellte Beispiel somit komplett nachvollziehen. Die fachlichen Modelle wurden z. T. mit ARIS *Business Designer* erstellt. Diese dienen jedoch lediglich als Ausgangspunkt für die weitere Entwicklung, so dass außer der frei erhältlichen Software von Intalio keine weitere spezielle Software erforderlich ist.

Die Erarbeitung des vorliegenden Beispiels diente zugleich dazu, das System Intalio|BPMS näher kennenzulernen. Es ist daher nicht auszuschließen, dass sich manche Sachverhalte mit diesem System auf andere Weise eleganter lösen lassen würden.

Um die Nachvollziehbarkeit mit Hilfe des Intalio-Systems zu erleichtern werden Tipps und Hinweise, die für die Arbeit mit Intalio|BPMS nützlich sind, in grauen Kästen dargestellt. Diese sind für das Verständnis der Vorgehensweise nicht erforderlich. Leser, die das Beispiel nicht selbst am System nachvollziehen wollen, können die grau hinterlegten Abschnitte daher überspringen.

Das komplette Beispiel kann im Übrigen auch als fertig ausgearbeitetes Projekt von der Webseite http://kurze-prozesse.de/?page_id=41 heruntergeladen und in den Intalio|BPMS *Designer* importiert werden. Hierzu muss die zip-Datei heruntergeladen und gespeichert werden. Im BPMS *Designer* ist unter „*File*" der Punkt „*Import*" auszuwählen. Anschließend muss „*Existing Projects into Workspace*" selektiert werden. Im folgenden Dialog klickt man „*Select archive file*" an und wählt die heruntergeladene zip-Datei aus. Schließlich wird das einzige angezeigte Projekt „Kreditbearbeitung" angeklickt und der Import mit „*Finish*" beenden.

Im vorliegenden Arbeitspapier werden die Begriffe „*Workflow Management-System*“ (WFMS) und „*Business Process Management-System*“ (BPMS) synonym gebraucht. Wird von einem fachlichen Prozessmodell oder einem Geschäftsprozessmodell gesprochen, so ist von einem Modell die Rede, das einen Ablauf aus Sicht der jeweiligen Fachabteilung darstellt und unabhängig von einer konkreten Implementierung ist. Ein *Workflow*-Modell oder ausführbares (Prozess-)Modell bezeichnet hingegen ein Modell, das direkt von einem konkreten BPMS ausgeführt werden kann. Wird in Kurzform nur von einem „Prozess“ gesprochen, ergibt sich aus dem jeweiligen Kontext, ob es sich um einen kompletten Geschäftsprozess oder einen ausführbaren *Workflow* handelt.

Die Arbeit ist folgendermaßen aufgebaut:

Kapitel 2 stellt die fachliche Modellierung des zu unterstützenden Prozesses in Form einer ereignisgesteuerten Prozesskette (EPK) sowie die verwendeten Datenstrukturen und das Rollenmodell vor.

Auf Grundlage dieses fachlichen Modells wird in Kapitel 3 zunächst das Rollenmodell implementiert. Anschließend wird das ausführbare *Workflow*-Modell inklusive der benötigten Datenstrukturen, der Benutzungsoberfläche und der erforderlichen Datentransformationen entwickelt und getestet.

Gegenstand von Kapitel 4 ist die ausführliche Darstellung eines kompletten Durchlaufs des entwickelten *Workflows*.

Kapitel 5 fasst das beschriebene Vorgehen zusammen und diskutiert die wesentlichen Erfahrungen bei der Entwicklung und Umsetzung dieses Vorgehens.

Ich danke Herrn Reiner Röhm von der Firma Ancud IT für seine Unterstützung und nützliche Hinweise zu diesem Paper.

2 Fachliche Modellierung

2.1 Geschäftsprozess Kreditvergabe

Es soll der in Abbildung 1 als ereignisgesteuerte Prozesskette (EPK) abgebildete Kreditvergabeprozess unterstützt werden.

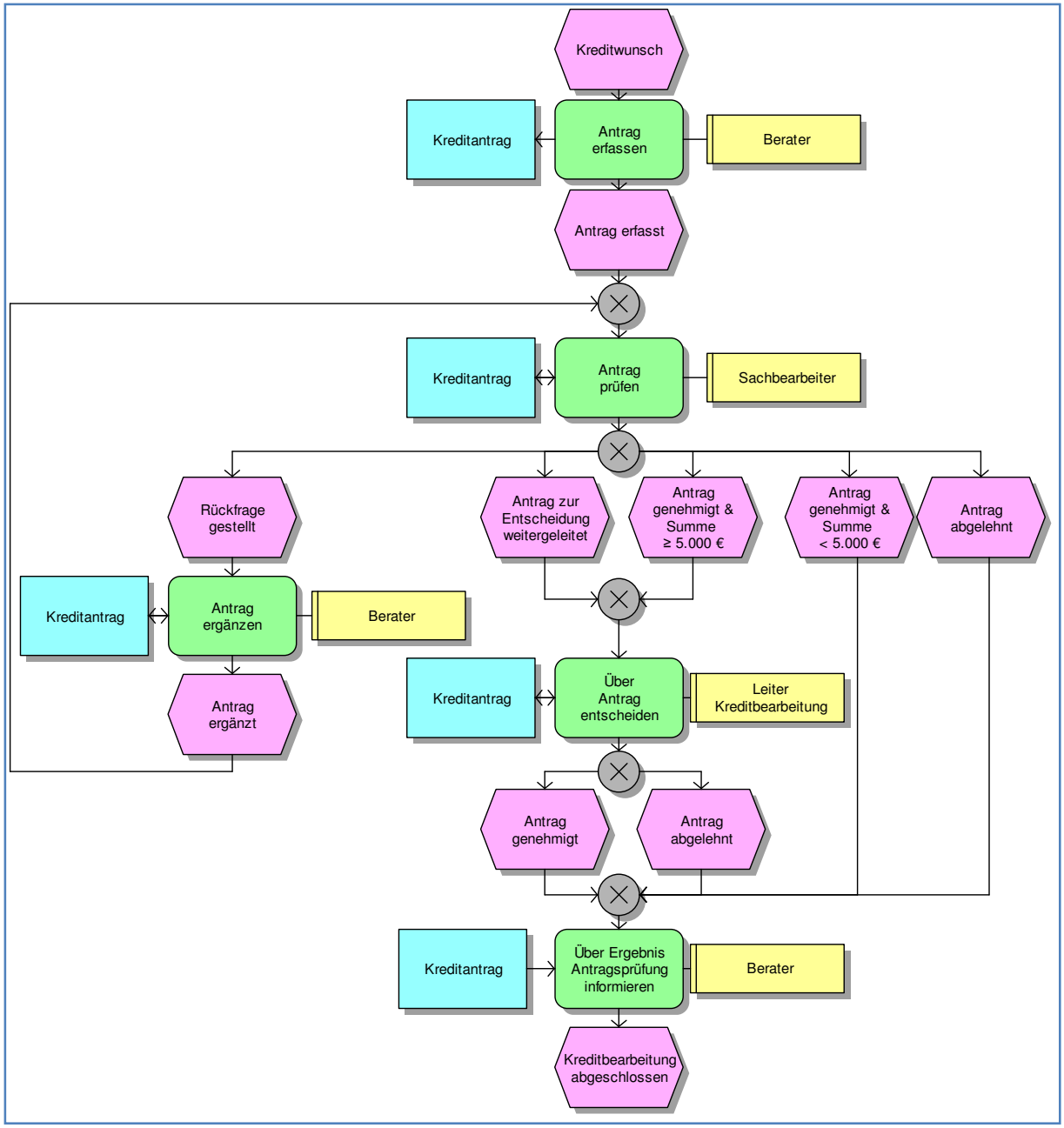

Abbildung 1: Geschäftsprozess Kreditvergabe

Der Ablauf beginnt mit dem als magentafarbenes Sechseck dargestellten Ereignis „Kreditwunsch“. Die Rolle „Berater“ (gelber Kasten) führt daraufhin die Funktion „Antrag erfassen“ (grünes, abgerundetes Rechteck) durch. Hierbei wird ein Informationsobjekt „Kreditantrag“ erzeugt. Ist die Funktion abgeschlossen, so tritt das Ereignis „Antrag erfasst“ ein. Hierauf prüft ein Sachbearbeiter den Antrag.

Die Antragsprüfung hat mehrere mögliche Endereignisse: Lehnt der Sachbearbeiter den Antrag ab oder genehmigt er ihn bei einer Antragssumme von unter 5.000 €, so wird als nächstes der Berater über das Ergebnis der Antragsprüfung informiert, und der Prozess ist zu Ende. Genehmigt der Sachbearbeiter den Antrag bei einer Antragssumme von 5.000 € oder höher oder leitet er ihn zur Entscheidung weiter, so muss als nächstes der Leiter der Kreditbearbeitung über den Antrag entscheiden. Weiterhin ist es möglich, dass der Sachbearbeiter noch Rückfragen hat. In diesem Fall muss der Berater den Antrag ergänzen, woraufhin dieser wiederum zur Prüfung durch den Sachbearbeiter geleitet wird.

Die fünf Endereignisse der Funktion „Antrag prüfen“ sind alternativ zueinander, d. h. es tritt immer genau eines dieser Ereignisse ein, nicht jedoch zwei oder mehr gleichzeitig. Dies wird durch das „X“ in dem kreisförmigen Konnektor ausgedrückt, das für „XOR“ (exklusives Oder) steht. Beim Zusammenführen alternativer Pfade wird ebenso ein XOR-Konnektor verwendet. Als Ergebnis der Funktion „Über Antrag entscheiden“ ist der Antrag entweder genehmigt oder abgelehnt. In beiden Fällen wird der Berater über das Ergebnis der Antragsprüfung informiert, womit die Kreditbearbeitung abgeschlossen ist. Die Ergebnisse der jeweiligen Funktionen, z. B. die getroffene Entscheidung, Ergänzungen durch den Berater etc. gehen jeweils in das Informationsobjekt „Kreditantrag“ ein.

Das Modell ist unabhängig von einer konkreten Technologie. So ist beispielsweise nicht festgelegt, ob es sich bei dem Informationsobjekt „Kreditantrag“ um ein Papierdokument oder um einen Datensatz in einer Datenbank handelt. Ebenso wenig ist dargestellt, ob die Weiterleitung des Antrags manuell in Papierform, per E-Mail oder durch ein BPMS erfolgt.

Tabelle 1 enthält die detaillierten Beschreibungen der einzelnen Funktionen. Die Rollen und Informationsobjekte werden in den folgenden Abschnitten genauer modelliert und näher erläutert.

Funktion	Beschreibung
Antrag erfassen	Es werden die erforderlichen Angaben für einen Kreditantrag in ein Formular eingetragen. Zusätzlich ist es möglich, eine Datei hochzuladen, die z. B. eine Kopie der Verdienstbescheinigung, Informationen zur Beurteilung von Immobilien o.ä. enthält.
Antrag prüfen	Der Antrag wird auf Vollständigkeit und inhaltlich geprüft. Sind noch Fragen offen, so werden diese eingetragen, und der Antrag wird zur Ergänzung zurück an den Berater geleitet. Bestehen keine Fragen mehr, so trifft der Prüfende eine Entscheidung über die Annahme oder Ablehnung des Antrags. Ist er sich über die Entscheidung unsicher, so kann er den Antrag zur Entscheidung an den Leiter der Kreditabteilung weiterleiten.
Antrag ergänzen	Es wird eine Antwort auf die Rückfrage(n) zu dem Antrag eingetragen. Außerdem ist es möglich, eine neue Datei mit Verdienstbescheinigungen, Sicherheiten etc. hochzuladen.
Über Antrag entscheiden	Die Antragsdaten werden angesehen, und der Antrag kann abgelehnt oder angenommen werden.
Über Ergebnis Antragsprüfung informieren	Der Berater bekommt die Entscheidung über den Antrag mitgeteilt.

Tabelle 1: Beschreibung der Funktionen

2.2 Informationsobjekt Kreditantrag

Struktur und Inhalt des Informationsobjektes „Kreditantrag“ sind in Abbildung 2 in Form eines UML Klassendiagramms dargestellt. Der Kreditantrag enthält eine Reihe von Attributen wie Antragsdatum, Antragssumme usw. Außerdem enthält ein Kreditantrag genau ein Informationsobjekt „Kunde“. Dieses ist hier über eine Komposition als Bestandteil des Kreditantrages dargestellt.

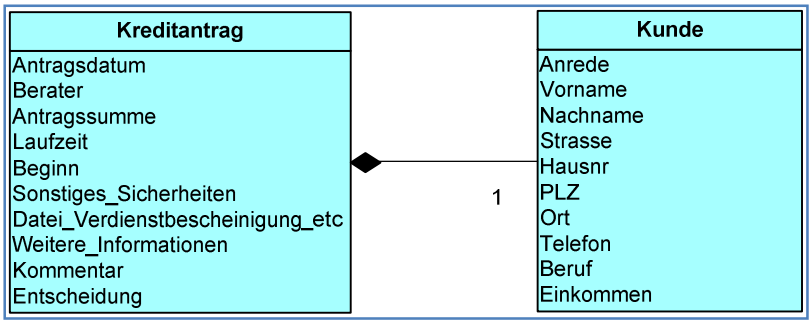

Abbildung 2: Informationsobjekte Kreditantrag und Kunde

Prinzipiell wäre es sinnvoller, Kunden-Objekte separat zu verwalten, um diese nicht jedesmal neu erfassen zu müssen und z. B. leicht herausfinden zu können, welche Kreditanträge ein bestimmter Kunde bereits gestellt hat. In diesem Fall müsste im oben vorgestellten Prozess zusätzlich dargestellt werden, dass zunächst das passende Kunden-Objekt ausgewählt und ggf. geändert wird (z. B. wenn die Adresse nicht mehr stimmt). Falls noch kein Objekt für den betreffenden Kunden besteht, müsste es neu angelegt werden. Um das Beispiel zu vereinfachen und weil später bei der Umsetzung in das Workflow System keine separate Datenhaltung implementiert werden soll, wurde hier die vorgestellte Modellierung gewählt. Da das Informationsobjekt „Kunde“ Teil des Informationsobjektes „Kreditantrag“ ist, wurde dies in der EPK nicht separat aufgeführt. In der Praxis würde man sicherlich eine separate Verwaltung der Kunden-Objekte realisieren.

Tabelle 2 beschreibt die Informationsobjekte näher. Hierbei handelt es sich ebenfalls um fachlich relevante Festlegungen, die noch unabhängig von der konkreten Implementierung sind. Später werden diese benötigt, um z. B. konkret zu verwendende Datentypen und Eingabeüberprüfungen zu ermöglichen.

Auch bei den Informationsobjekten wurde wiederum eine Reihe von Vereinfachungen zugrunde gelegt. So können beispielsweise nur Kunden mit Wohnsitz im Inland und Einkommen in Euro berücksichtigt werden. Auch wurde im vorliegenden Fall darauf verzichtet, gesondert Nummern für Kreditanträge und Kunden zu vergeben. Die Verwaltung der zu den ausführbaren Prozessen gehörenden Daten soll später komplett vom BPMS übernommen werden. Zur Identifikation eines Kreditantrages kann dann die vom BPMS vergebene Nummer der Prozessinstanz verwendet werden. Kundennummern entfallen aufgrund des oben erwähnten Verzichts auf eine eigenständige Kundendatenverwaltung.

Attribut	Datentyp	Mussfeld	Restriktionen	Beschreibung
Informationsobjekt: Kreditantrag				
Antragsdatum	Datum	ja		Das Datum, an dem der Antrag angelegt wurde.
Berater	Text, 30 Zeichen	Ja		Name des Beraters, der den Antrag erfasst hat
Antragssumme	Dezimalzahl, zwei Nachkommastellen	ja	>0,00; ≤1.000.000,00	Die beantragte Auszahlungssumme in Euro
Laufzeit	Positive, ganze Zahl	ja	≤30	Die Laufzeit des Kredites in Jahren
Beginn	Datum	ja		Datum, zu dem der Kreditbetrag dem Kreditnehmer zur Verfügung stehen soll.
Sonstiges_Sicher-heiten	Text, 2000 Zeichen	ja		Text mit sonstigen Informationen zu dem Antrag, insbesondere über vorliegende Sicherheiten
Datei_Verdienst-bescheinigung_etc	Datei	nein		Datei mit eingescannten Dokumenten, z. B. eine Verdienstbescheinigung, Besitzurkunden, Fotos und Pläne von Immobilien etc.
Weitere_Informa-tionen	Text, 2000 Zeichen	nein		Feld für Rückfragen des Sachbearbeiters und Antworten des Beraters
Kommentar	Text, 2000 Zeichen	nein		Kommentar des Antragsprüfers oder Entscheiders.
Entscheidung	Aufzählung	nein	„genehmigt“ oder „abgelehnt“	Gibt am Ende der Antragsbearbeitung an, ob der Antrag genehmigt oder abgelehnt worden ist.
Informationsobjekt: Kunde				
Anrede	Aufzählung	ja	„Herr“ oder „Frau“	Geschlecht des Kunden laut Identitätsnachweis
Vorname	Text, 30 Zeichen	ja		Vorname des Kunden laut Identitätsnachweis
Nachname	Text, 30 Zeichen	ja		Nachname des Kunden laut Identitätsnachweis
Straße	Text, 30 Zeichen	ja		Straße des Hauptwohnsitzes des Kunden
Hausnr	Text, 30 Zeichen	ja		Hausnummer des Hauptwohnsitzes des Kunden. Typischerweise eine Zahl, evtl. aber auch ein Buchstabe oder eine Zahl-Buchstabenkombination
PLZ	Text, 5 Zeichen	ja	Nur Ziffern erlaubt	Postleitzahl des Hauptwohnsitzes des Kunden
Ort	Text, 30 Zeichen	ja		Wohnort des Hauptwohnsitzes des Kunden
Telefon	Text, 30 Zeichen	nein	Nur Ziffern sowie -, /, (und) erlaubt	Telefonnummer des Kunden, unter der er tagsüber erreichbar ist.
Beruf	Text, 30 Zeichen	ja		Derzeit ausgeübter Beruf.
Einkommen	Dezimalzahl, zwei Nachkommastellen	ja	>0,00; ≤100.000,00	Monatliches Einkommen in Euro laut Einkommensnachweis

Tabelle 2: Beschreibung der Informationsobjekte

2.3 Rollen im Kreditvergabeprozess

Das in Abbildung 3 dargestellte Rollenmodell ist ziemlich einfach. Es besteht aus den drei bereits in der EPK verwendeten Rollen. Sie sind in Tabelle 3 näher beschrieben. Die einzig modellierte Beziehung drückt aus, dass die Rolle „Sachbearbeiter“ eine Verallgemeinerung der Rolle „Leiter Kreditbearbeitung“ darstellt. Dies bedeutet, dass ein Leiter der Kreditbearbeitung auch alle Eigenschaften eines Sachbearbeiters besitzt, insbesondere kann er auch die der Rolle Sachbearbeiter zugeordneten Funktionen ausführen.

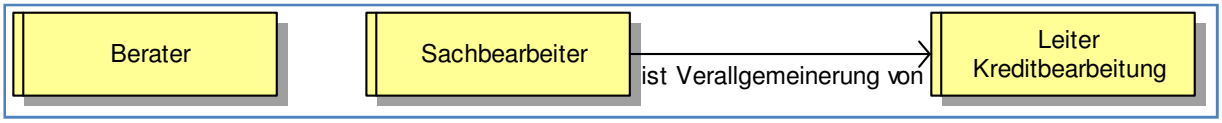

Abbildung 3: Rollenmodell

Bei den Rollen handelt es sich nicht notwendigerweise um Stellen in einem Unternehmen, sondern lediglich um die Rollen, die die betreffenden Personen aus Sicht des durchgeführten Prozesses einnehmen.

Rolle	Beschreibung
Berater	Ein Mitarbeiter einer Filiale oder ein freier Finanzberater, der Kunden bzgl. der Kreditaufnahme berät und den direkten Ansprechpartner des Kunden darstellt.
Sachbearbeiter	Ein Mitarbeiter einer Kreditabteilung, der Kreditanträge prüft, bearbeitet und verwaltet.
Leiter Kreditbearbeitung	Ein Leiter einer Kreditabteilung ist verantwortlich für das gesamte Kreditgeschäft, insbesondere für die Entscheidung über größere Kreditsummen und Nicht-Standard-Fälle.

Tabelle 3: Beschreibung der Rollen

In einem BPMS erfolgt die Rollenvergabe durch Anlegen eines Benutzers in dem System und seiner Zuordnung zu einer oder mehrerer in dem System angelegten Rollen. Die Besetzung der verschiedenen Rollen kann sich auch öfter ändern. Im vorliegenden Fall sind die Rollen des Prozesses jedoch weitgehend aus den Stellenbeschreibungen des Kreditinstituts abgeleitet. Die in Abbildung 4 gezeigte Rollenbesetzung ändert durch einzelne Mitarbeiter ändert sich eher selten. Sie lässt sich aus dem Organigramm des Unternehmens ableiten.

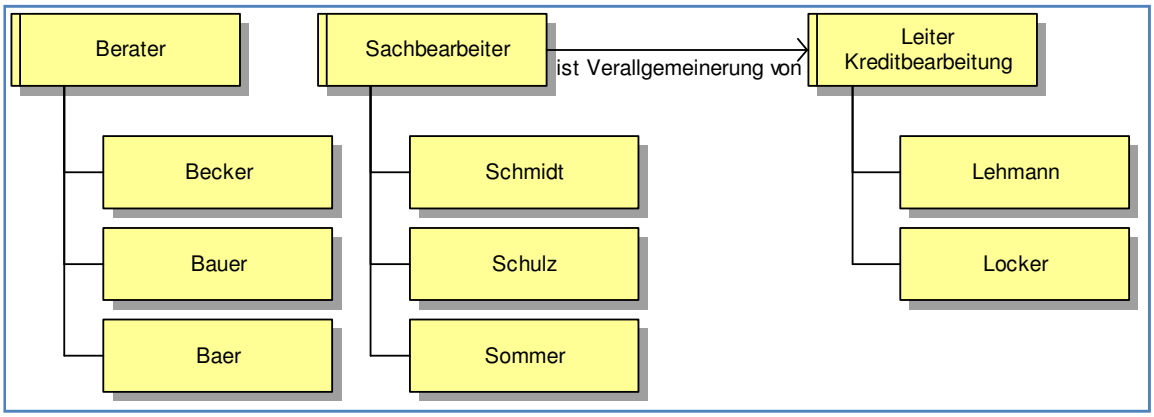

Abbildung 4: Rollenbesetzungen

3 Entwicklung des ausführbaren Workflow-Modells

3.1 Voraussetzungen

Für die Entwicklung des ausführbaren Modells wird die Modellierungskomponente des BPMS verwendet. Für die Ausführung des entwickelten Modells wird anschließend das Herzstück des BPMS, die *Process Engine* benötigt, auf die über eine ebenfalls vom BPMS bereitgestellte Benutzungsoberfläche zugegriffen wird.

Verwendet wurden:

- Intalio|BPMS *Designer* 5.1 *for Windows* als Modellierungskomponente
- Intalio|BPMS *Server* 5.1 als *Process Engine*

Die beiden genannten Systeme können nach Registrierung von der Webseite http://bpms.intalio.com heruntergeladen und gemäß der ebenfalls auf der genannten Seite veröffentlichten Installationsanleitung aufgesetzt werden. Voraussetzung ist ein installiertes Java 5 oder Java 6 *Run Time Environment* (JRE), das gegebenenfalls von http://java.sun.com heruntergeladen und installiert werden kann.

Die Installation des BPMS *Designers* wird durch Doppelklick auf das Installationsprogramm gestartet (hier verwendet: „designer.installer-5.1.0.009.jar“). Für den *Server* muss die heruntergeladene zip-Datei (hier verwendet: „intalio-bpms-server-5.1.0.013.zip“) in das gewünschte Installationsverzeichnis entpackt werden.

Falls es wegen zu langer Pfadnamen Probleme beim Entpacken der Zip-Datei gibt, sollte ggf. ein anderes Programm zum Entpacken verwendet werden (z. B. Winrar).

Der Pfadname des Installationsverzeichnisses darf keine Leerzeichen enthalten. Unter *Windows Vista* kann daher z. B. der Ordner „C:\Programme“ nicht verwendet werden, da dieser intern eigentlich „C:*Program Files*“ heißt.

Zum Starten des *Designers* muss lediglich die Datei „designer.exe“ ausgeführt werden.

Der Server wird über die Eingabeaufforderung gestartet. Hierin wird zunächst im Installationsverzeichnis in das „bin“-Verzeichnis gewechselt und anschließend „startup.bat“ aufgerufen. Möchte man den Server per Mausklick starten können, so kann man sich im bin-Verzeichnis eine Datei „startupintalio.bat“ anlegen und mit einem einfachen Texteditor folgende zwei Zeilen eintragen:

```
cd [Installationsverzeichnis]\bin
startup.bat
```

Das Starten des Servers dauert eine Zeitlang. Es wurde erfolgreich gestartet, wenn in der Eingabeaufforderung „*Geronimo Application Server started*“ ausgegeben wird. Das Geronimo-Fenster kann minimiert werden. Schließen des Fensters oder Eingabe von „Strg-C“ stoppt den Server wieder.

3.2 Anlegen von Rollen und Benutzern

Das Anlegen von Rollen und Benutzern ist in der frei erhältlichen *Community Edition* von Intalio|BPMS etwas spartanisch realisiert. Sämtliche Benutzer werden samt Passwörtern und Rollenzuordnungen einfach in eine XML-Datei auf dem *Server* eingetragen. Für einen produktiven Einsatz wäre dies wenig geeignet. Im laufenden Betrieb ist es erforderlich, Benutzer über eine entsprechende Oberfläche verwalten zu können. Ebenso sollten Benutzer ihrer eigenen Daten und Passwörter pflegen können. Das Ablegen unverschlüsselter Passwörter in einer zentralen Datei ist zudem unter Sicherheitsaspekten abzulehnen.

Für den produktiven Einsatz bietet Intalio u. a. einen kostenpflichtigen LDAP-*Connector* an, über den das BPMS mittels LDAP (*Lightweight Directory Access Protocol*) an eine zentral genutzte Benutzerverwaltung angeschlossen werden kann. Für Test- und Entwicklungszwecke wie in dem vorliegenden Beispiel kann die Datei-basierte Benutzerverwaltung verwendet werden.

Im Installationsverzeichnis des Servers finden sich unter „*[Installationsverzeichnis Server]\var\config*“ die Dateien „*security.xml*“ und „*securityConfig.xml*“, die angepasst werden müssen. Sicherheitshalber sollten Änderungen an diesen Dateien durchgeführt werden, wenn der Server nicht läuft. Außerdem empfiehlt es sich, Sicherheitskopien der Originaldateien anzulegen, um die geänderten Dateien ggf. wieder durch die auf jeden Fall funktionierenden ursprünglichen Versionen zu ersetzen.

Die Rollen- und Mitarbeiter-Informationen aus dem Organigramm werden in die Datei „security.xml“ eingetragen. Hierzu wird zunächst ein neuer Bereich (engl. *realm*) „Kreditbank“ eingetragen:

```
<realm identifier="kreditbank">
    …
</realm>
```

Listing 1: Eintrag eines neuen Bereichs

Innerhalb dieses Bereichs (d. h. zwischen dem öffnenden <realm ..> und dem schließenden </realm>) werden nun die Rollen gemäß Abbildung 3 eingetragen. Jede Rolle erhält einen *identifier* als eindeutige Bezeichnung und eine Beschreibung (*description*). Da die Beschreibung im BPMS nicht weiter verwendet wird, wird sie hier sehr knapp gehalten. Da ein Sachbearbeiter eine Verallgemeinerung eines Leiters der Kreditbearbeitung ist, wird bei der Rolle „leiter-kredit“ mittels „descendantRole“ eingetragen, dass diese ein Abkömmling (*descendant*) der Rolle „sachbearbeiter“ ist. Man erhält folgende XML-Darstellung der Rollen-Definitionen:

```
<role identifier="leiter-kredit">
    <description>Leiter Kreditbearbeitung</description>
    <descendantRole>sachbearbeiter</descendantRole>
</role>

<role identifier="berater">
    <description>Berater</description>
</role>

<role identifier="sachbearbeiter">
    <description>Sachbearbeiter</description>
</role>
```

Listing 2: Eintrag der Rollen

Schließlich sind noch die Benutzer mit ihren Rollenzuordnungen gemäß Abbildung 4 einzutragen. Dies geschieht ebenfalls innerhalb des Bereichs „kreditbank“.

Beispielhaft wird hier nur der Eintrag für einen der Benutzer dargestellt:

```
<user identifier="lehmann">
    <name>Lorenz Lehmann</name>
    <email>lorenz.lehmann@examples.com</email>
    <password>password</password>
    <assignRole>leiter-kredit</assignRole>
</user>
```

Listing 3: Eintrag eines Benutzers

Für jeden Benutzer (*user*) werden außer seinem Benutzernamen (*identifier*) noch der volle Name, die E-Mail-Adresse und das Passwort eingetragen. Zum Anmelden werden später Benutzername und Passwort benötigt.

Mittels „*assignRole*“ wird dem Benutzer seine Rolle zugeordnet. Für einen Benutzer kann es auch mehrere „*assignRole*“-Einträge geben. Da die Rolle „leiter-kredit“ bereits als „*descendantRole*“ der Rolle „sachbearbeiter“ eingetragen ist, erhält der Benutzer in dem obigen Beispiel bereits automatisch alle Eigenschaften und Rechte der Rolle „sachbearbeiter“, d. h. die Rolle „sachbearbeiter“ muss ihm nicht mehr separat zugewiesen werden. Die Einträge für die anderen Benutzer erfolgen analog.

Es empfiehlt sich, den Bereich „kreditbank“ zur voreingestellten „*defaultRealm*“ zu machen. Hierdurch können sich die oben angelegten Benutzer direkt mit Ihren Benutzernamen ohne die ansonsten erforderliche Angabe des Bereichs anmelden. Hierzu wird in der Datei „*security.xml*“ die „*defaultRealm*“ von „intalio“ zu „kreditbank“ geändert:

```
<defaultRealm>kreditbank</defaultRealm>
```

Weiterhin muss in der Datei „*securityConfig.xml*“ der erste Eintrag unter „*<beans>*“ geändert werden, so dass er folgendermaßen aussieht:

```
<bean id="realms" class="org.intalio.tempo.security.impl.Realms">
      <property name="defaultRealm" value="kreditbank"/>
      <property name="securityProviders">
            <list>
                  <ref bean="securityProvider"/>
            </list>
      </property>
</bean>
```

Die Änderung der „*defaultRealm*“ hat zur Folge, dass sich der voreingestellte Administrations-Benutzer „admin“ von nun an mit Angabe des Bereichs als „intalio/admin“ anmelden muss. Um andere Beispiele von bpms.intalio.org gemäß den jeweiligen Anleitungen ausführen zu können, müssen die Dateien „*security.xml*“ und „*securityConfig.xml*“ wieder gegen die Originaldateien ausgetauscht werden.

Für Testzwecke ist es gelegentlich nützlich, einen Benutzer zu haben, der alle definierten Rollen in einer Person vereinigt. Dann muss beim Testen eines Workflows über mehrere Rollen hinweg nicht

ständig der Benutzer gewechselt werden. Hierzu können beispielsweise dem Administrator noch zusätzlich die neu definierten Rollen zugeordnet werden:

```
<assignRole>kreditbank\leiter-kredit</assignRole>
<assignRole>kreditbank\berater</assignRole>
```

Die Sachbearbeiter-Rolle muss wiederum nicht gesondert zugeordnet werden, da sie wie oben dargestellt in der Rolle „leiter-kredit“ mit enthalten ist.

3.3 Anlegen des Projektes

Im Designer wird eine neues „Intalio|*BPMS Business Process Project*“ namens „Kreditbearbeitung“ angelegt.

Der Intalio|*BPMS Designer* basiert auf der Entwicklungsumgebung „*Eclipse*“. Die Bedienung von *Eclipse* wird hier nicht im Einzelnen erläutert. Erläuterungen hierzu finden sich in der Hilfe und auf der *Eclipse*-Homepage www.eclipse.org. Die Hilfe im Intalio|*BPMS Designer* enthält andererseits über *Eclipse*-Grundlagen hinaus keine *BPMS Designer*-spezifische Hilfe. Hierzu muss auf die Informationen auf bpms.intalio.com zurückgegriffen werden.

Beim Anlegen eines Projektes wird ein Dialog mit einer Reihe von Einstellungen angezeigt, die alle unverändert übernommen werden können.

Zur Organisation der Dateien wird empfohlen, in dem Projekt die folgenden Verzeichnisse anzulegen:

- *data* Ablage von Datenstrukturen
- *forms* Ablage von Formularen der Benutzeroberfläche
- *processes* Ablage von Prozessmodellen

Um alle Projektinformationen zusammenzuhalten, können Kopien der Dateien „*security.xml*“ und „*securityConfig.xml*“ ebenfalls im Projekt in einem security-Verzeichnis abgelegt werden.

Vom System wird ein Verzeichnis „*build*“ angelegt. Hier werden vom Designer automatisch generierte Dateien gespeichert. Diese sollten keinesfalls direkt bearbeitet werden, da man das Projekt hierdurch komplett in einen inkonsistenten, nicht mehr zu verwendenden Zustand versetzen kann.

Da unter widrigen Umständen solche inkonsistenten Zustände herbeigeführt werden können, empfiehlt es sich, bei Erreichen eines funktionierenden Projektstandes die kompletten Inhalte des Projektverzeichnisses aus dem *Workspace*-Verzeichnis zu kopieren und unter einer neuen Versionsbezeichnung zu speichern. So kann bei Bedarf immer auf einen funktionierenden Projektstand zurück gewechselt werden.

3.4 Anlegen des Prozessmodells

Es wird zunächst ein neues Prozessmodell namens „Kreditvergabe“ angelegt. Das Prozessmodell wird in BPMN (*Business Process Modeling Notation*) erstellt.

In dem Modell werden insgesamt vier „*Pools*“ angelegt: Drei für die verschiedenen am Prozess beteiligten Rollen, sowie eine für das System selbst. In Abbildung 5 wurden die *Pools* der Rollen verschieden eingefärbt: Die externe Berater-Rolle gelb, die internen Rollen orange und der System-*Pool* blau. Dies ist allerdings für die Funktion des Systems unerheblich.

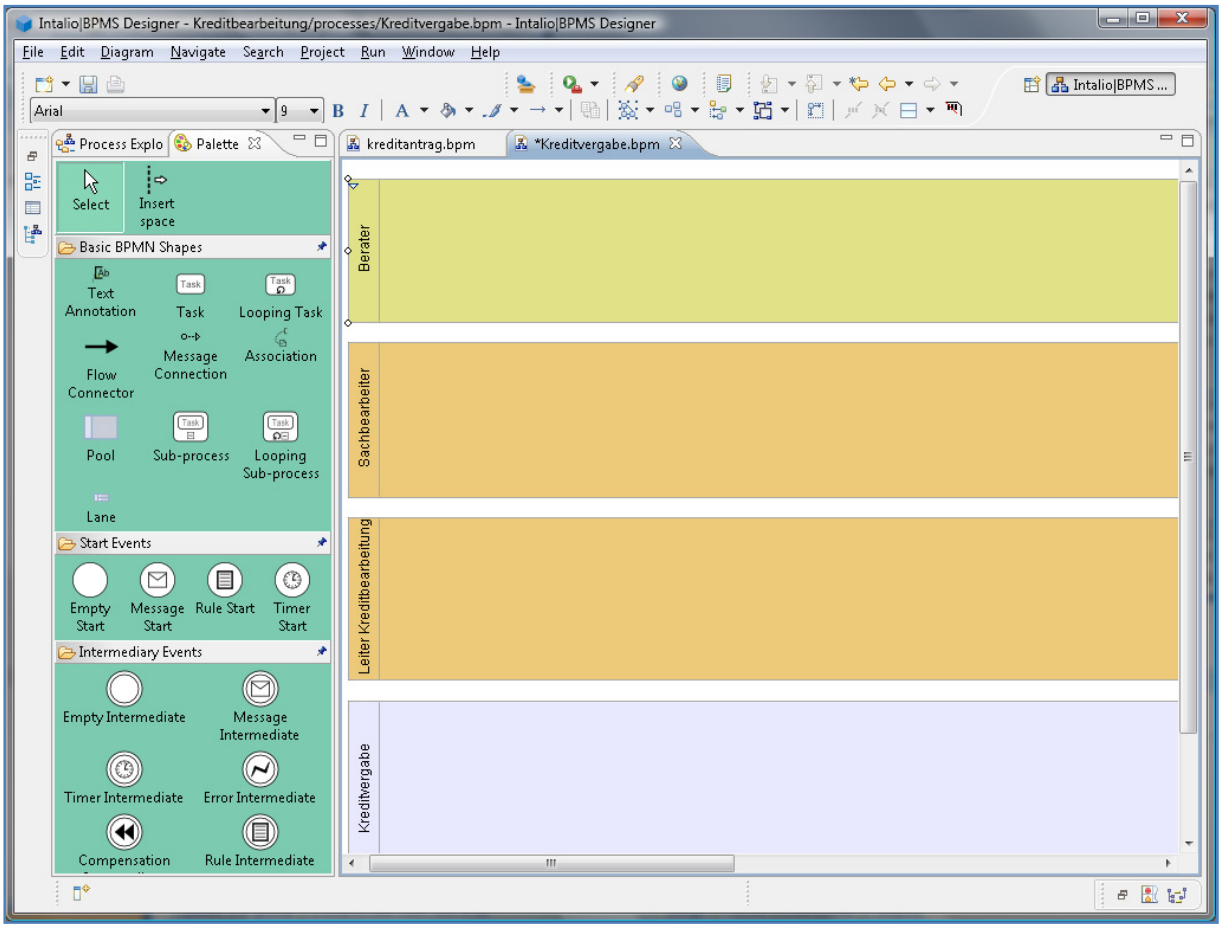

Abbildung 5: Pools im BPMS Designer

Im zweiten Schritt sollen den oberen drei *Pools* nun die Rollen zugeordnet werden. Zunächst muss für diese *Pools* noch definiert werden, dass diese nicht ausführbar sind (*non executable*), d. h. sie werden nicht von der *Process Engine* abgearbeitet, sondern von den Benutzern. Nur der vierte *Pool* soll ausführbar sein. Anschließend werden für jeden der drei *Pools* die oben definierten Rollen eingetragen (Abbildung 6). Die Zuordnung wird grafisch durch ein kleines Personen-Symbol dargestellt.

Das Anlegen des Prozessmodells erfolgt im Ordner „*processes*“ mittels „*New*“ und Auswahl von „*Business Process Diagram*“.

Der System-*Pool* sollte den Namen des Prozesses erhalten (und nicht etwa „System“ oder ähnliches), da dieser später in der Administrationsoberfläche in der Liste der Prozesse angezeigt wird.

Pools können über das Kontextmenü der rechten Maustaste „*executable*“ oder „*non executable*“ gesetzt werden. Bei nicht ausführbaren *Pools* wird das Titelfeld dunkler dargestellt.

Zur Zuordnung einer Rolle wird der *Pool* selektiert. Anschließend wird die Rolle im Fenster „*Properties*“ unter „*Workflow*“ in das Feld „*Role(s)*“ eingetragen. Hierbei ist die „*realm*“ voranzusetzen und durch einen Schrägstrich vom Rollennamen zu trennen. „*Realms*“ und Rollen müssen den Einträgen in der Datei „*security.xml*“ entsprechen, da der *Server* sonst später keine Zuordnung zu Benutzern vornehmen kann.

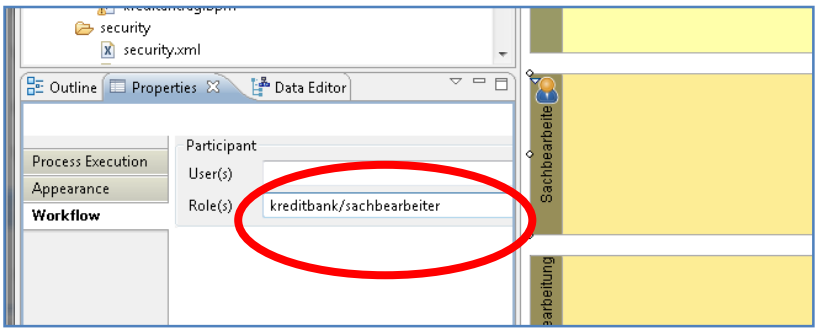

Abbildung 6: Zuordnung der Rollen

3.5 Festlegen der Datenstrukturen

Zur Umsetzung der Datenstrukturen sind geeignete Datentypen für den Kreditantrag und den Kunden gemäß Abbildung 2 zu spezifizieren. Anschließend wird eine Prozessvariable vom Typ „Kreditantrag“ angelegt, die während des Prozesses die Daten des Antrags aufnimmt.

In einfachen Fällen kann man auch auf gesonderte Prozessvariable verzichten, indem man die *Output*-Daten einer Benutzeraktivität direkt in die folgende Benutzeraktivität eingehen lässt. Bei komplexeren Abläufen wird dies jedoch schnell unübersichtlich, und insbesondere bei mehrfach durchlaufenen Schleifen lässt sich dann nicht mehr eindeutig definieren, welche Daten von welchem Durchlauf verwendet werden sollen.

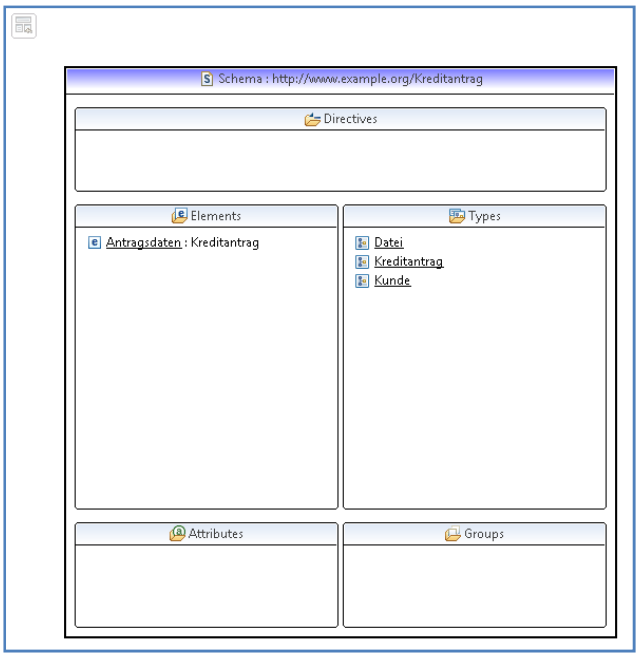

Abbildung 7:Darstellung der XML Schema-Datei

Zur Definition von Datenstrukturen wird eine XML Schema-Datei angelegt. Die Inhalte werden mit Hilfe des im Designer enthaltenen XML *Schema Editors* gepflegt. Abbildung 7 zeigt auf der rechten Seite die angelegten Datentypen. Neben dem Kreditantrag wurde für den Kunden ein eigener Datentyp angelegt, der die verschiedenen Kunden-Attribute festlegt. Der dritte Datentyp, Datei, enthält Strukturen für die Informationen, die der Intalio *Server* benötigt, um Dateien hochzuladen und auf hochgeladene Dateien wieder zuzugreifen und sie zu öffnen.

Auf der linken Seite ist das konkrete Datenelement „Antragsdaten“ definiert, das später als Prozessvariable dienen soll. Es ist vom Typ „Kreditantrag“ und enthält somit alle Attribute, die für den Kreditantrag definiert wurden.

Die Darstellung der Struktur und Zusammenhänge der Datentypen im XML *Schema Editor* wird in Abbildung 8 gezeigt. Hier finden sich die Attribute aus Abbildung 2 wieder. Die Beziehung zwischen Kreditantrag und Kunde wird realisiert, indem im Kreditantrag ein Feld „Kunde“ eingetragen wird, das als Datentyp „Kunde“ zugeordnet bekommt.

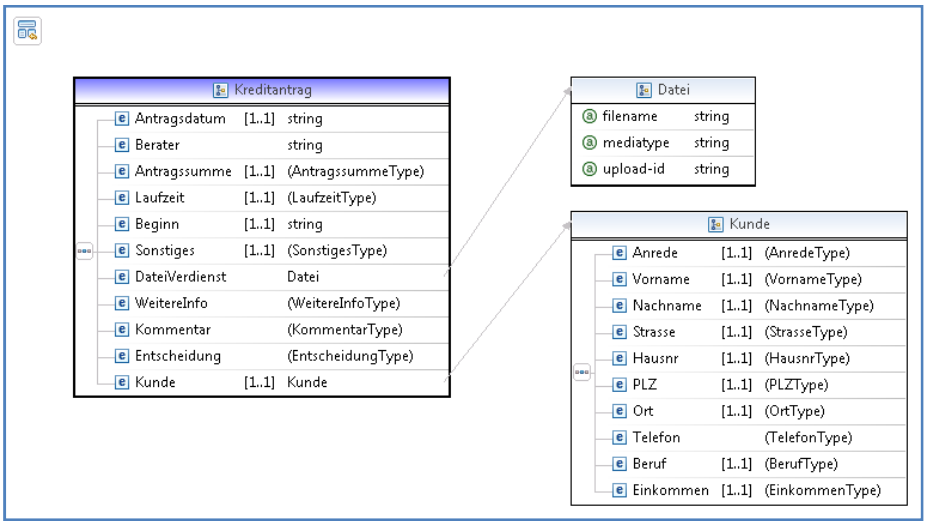

Abbildung 8: Struktur und Zusammenhang der Datentypen

Die neue Schema-Datei wird im Verzeichnis „*data*“ über „*New*“ > „*Other*“ angelegt, wobei im Auswahlmenü unter „XML“ der Eintrag „XML Schema“ ausgewählt werden muss. Im Beispiel wurde die Datei „Kreditantrag.xsd“ genannt.

Neue Elemente werden im jeweiligen Bereich des *Schema Editors* über das mit der rechten Maustaste erreichbare Kontext-Menü angelegt. Zunächst wird in der rechten Hälfte unter „*Types*“ mit „*Add Complex Type*“ ein zusammengesetzter Datentyp angelegt und mit „Datei“ bezeichnet. Durch Doppelklick auf diesen neuen Datentyp „Datei“ erhält man eine Detaildarstellung. Dort kann man mit „*Add Attribute*“ ein neues Feld hinzufügen. Über das Symbol links oben gelangt man zurück zur Hauptansicht. Bei den Feldern der Datentypen Kunde und Kreditantrag handelt es sich nicht um Attribute, die selbst nur einen einfachen Datentyp haben können (z. B. *string*), sondern um Elemente, die selbst wieder einen komplexen Datentyp haben können. Die Elemente werden mit „*Add Element*“ erstellt.

Standardmäßig ist für jedes Feld der Datentyp *string* eingestellt. Um einen anderen Datentypen zu erhalten muss man das Feld selektieren und auf „*string*“ klicken. Dann erhält man eine Auswahlliste

von Standard-Datentypen. Mittels „*Browse*“ lassen sich weitere, selbst definierte Datentypen auswählen. Damit diese in der Liste vorhanden sind, ist die Schema-Datei vorher zu speichern. Im vorliegenden Fall ist für das Feld „DateiVerdienst“ der gerade definierte Datentyp „Datei“ auszuwählen, für das Feld „Kunde“ der Datentyp „Kunde“.

Als Detaildarstellung des Datentyps „Kreditantrag“ ergibt sich das in Abbildung 8 gezeigte Bild. Um die im Folgenden beschriebenen weiteren Festlegungen für die Datentypen der einzelnen Felder zu treffen, ist das jeweilige Feld zu selektieren. Dann werden die Details im Fenster „*Properties*“ angezeigt, wo sie auch verändert werden können.

Das Datenelement „Antrag“ wird in der linken Hälfte des *Schema Editors* unter „*Elements*“ mittels „*Add Element*“ angelegt. Es erhält den gerade angelegten Datentyp „Kreditantrag“.

Die anderen Attribute erhalten einfache Datentypen, in der Regel „*string*“ (Zeichenkette). Für die meisten Attribute sind gemäß Tabelle 2 eine Reihe von Restriktionen einzuhalten. Diese werden über eigene Dialoge eingegeben. In der Übersichtsdarstellung lässt sich beispielsweise der Angabe „(Laufzeit*Type*)“ entnehmen, dass es hierfür noch genauere Festlegungen gibt, dass aber kein eigener Datentyp dafür definiert wurde. Ein separat definierter Datentyp wird ohne Klammer dargestellt.

„Antragsdatum“ und „Beginn“ sind hier vom Typ „*string*“, obwohl es einen eigenen Datentyp „*date*“ (Datum) gibt. Die Darstellung des Datentyps „*date*“ erfolgt bei der Ausgabe jedoch standardmäßig im amerikanischen Format. Zudem werden manuell eingegebene und automatisch ermittelte Datumsangaben verschieden ausgegeben. Daher sollen im Sinne einer einheitlichen Ausgabe sämtliche Datumsangaben in *Strings* nach deutschem Datumsformat umgewandelt werden.

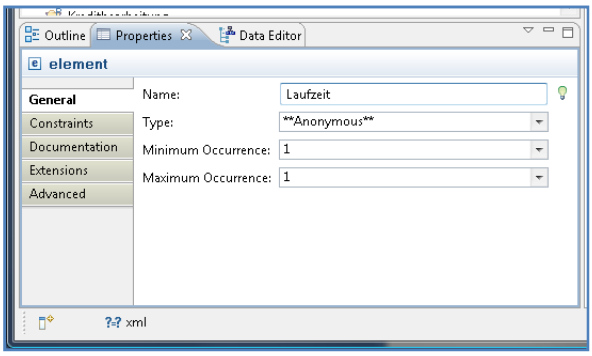

Abbildung 9: Eigenschaften des Attributs „Laufzeit“: Minimales und maximales Auftreten des Attributs

Die genauen Angaben zum Attribut „Laufzeit“ finden sich im *Properties* (Eigenschaften)-Fenster dieses Atttributs (Abbildung 9). Hier ist zunächst in den Feldern „*Minimum Occurrence*“ und „*Maximum Occurrence*“ eingetragen, dass die Laufzeit sowohl mindestens einmal als auch maximal einmal auftreten muss – also genau einmal, da es sich ja laut Tabelle 2 um ein Muss-Attribut handelt. In Abbildung 8 ist dies für die verschiedenen Attribute in Form von „[1..1]“ angegeben. Für Kann-Attribute wurde diese Festlegung nicht getroffen. Hier hätte man zwar [0..1] eintragen können, doch kann einem gewöhnlichen Attribut sowieso nur einen Wert zugeordnet werden, wenn kein spezieller Datentyp für eine Liste o. ä. verwendet wurde.

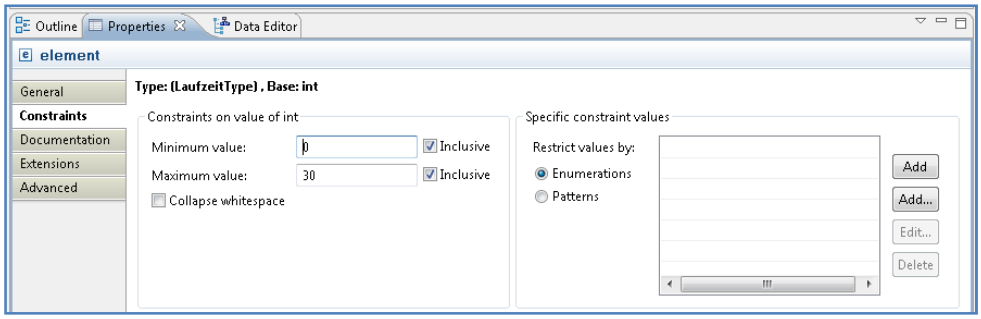

Abbildung 10: Eigenschaften des Attributs „Laufzeit“: Restriktionen

Abbildung 10 zeigt den Basisdatentyp „*int*“ (*Integer*), also eine ganze Zahl, sowie den kleinsten und größten erlaubten Wert. Auch die Definition von Aufzählungstypen ist möglich. So zeigt Abbildung 11, dass für das Attribut „Entscheidung“ nur die Werte „genehmigt“ und „abgelehnt“ möglich sind.

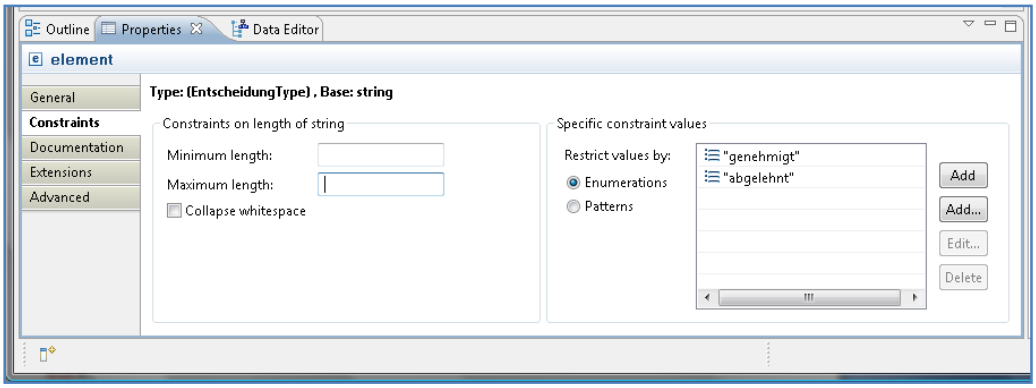

Abbildung 11: Eigenschaften des Attributs „Entscheidung“

Insgesamt lassen sich die verschiedenen Einschränkungen sehr genau spezifizieren. Hierzu sind z. T. etwas genauere Kenntnisse über XML Schema sowie die darin verwendeten Datentypen und regulären Ausdrücke erforderlich. Diese lassen sich teilweise über den in Abbildung 11 gezeigten „*Properties*“-Editor eingeben, z. T. ist gar eine direkte Eingabe im XML-Quellcode erforderlich, der im Register „*Source*“ angezeigt werden kann (Abbildung 12). So kann z. B. die Einschränkung, dass Geldsummen nur über zwei Nachkommastellen verfügen, nur direkt über den XML-*Code* eingegeben werden.

Um ein ablauffähiges Modell zu erhalten reicht es aber auch aus, lediglich die Basisdatentypen wie „*string*“ (Text), „*int*“ (ganze Zahl), „*decimal*“ (Dezimalzahl) und „*date*“ (Datum) festzulegen. Zwar sind dann prinzipiell auch recht sinnlose Eingaben möglich, doch dienen die meisten Attribute nur dazu, in späteren Arbeitsschritten angezeigt zu werden, so dass der prinzipielle Ablauf nicht eingeschränkt wird. Die Basisdatentypen lassen sich aus den Angaben in Tabelle 2 ermitteln.

Wichtig ist insbesondere, dass das Attribut „Antragssumme“ vom Typ „*decimal*“ ist, da die Antragssumme später zur Entscheidung über den weiteren Prozessverlauf mit einem Grenzwert verglichen wird. In Abbildung 8 ist als Datentyp „(Antragssumme*Type*)“ angegeben, da der Basis-datentyp „*decimal*“ im *Properties*-Fenster den „*Minimum Value*“ 0 erhalten hat, damit nur positive Werte möglich sind.

Die Überprüfung der erlaubten Werte darf jedoch nicht erst erfolgen, wenn die eingegebenen Daten der Prozessvariablen zugeordnet werden. Um sicherzustellen, dass nur erlaubte Werte eingegeben werden, sind diese bereits bei der Eingabe zu überprüfen. Dies wird weiter unten beim Anlegen des Eingabe-Dialogs erläutert.

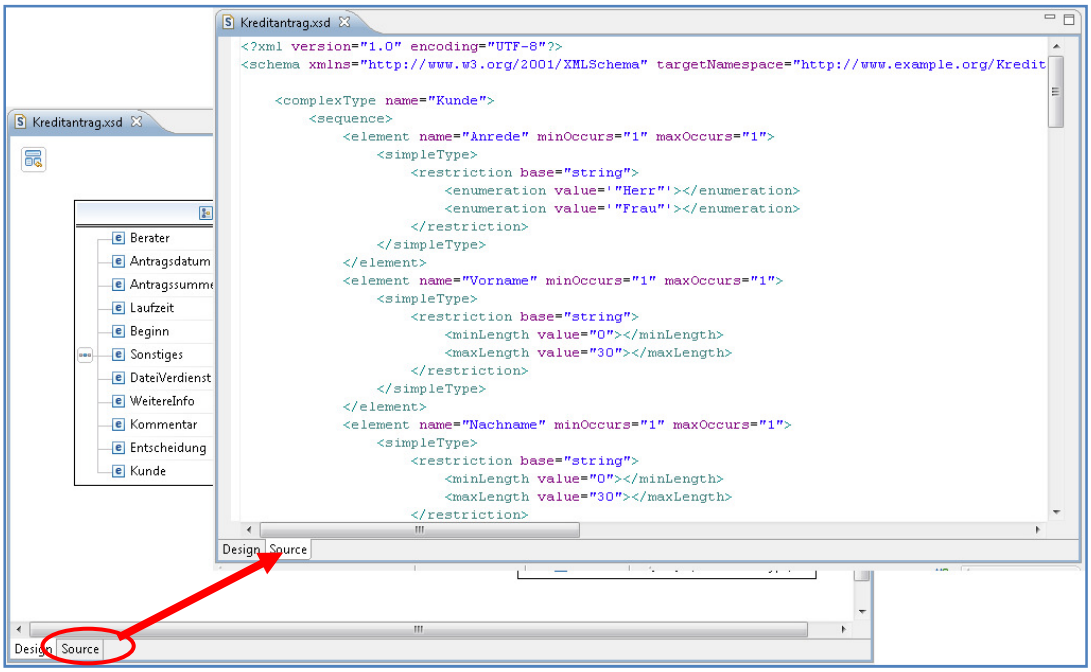

Abbildung 12: XML-Darstellung des Schemas

3.6 Zuordnen der Prozessvariablen

Nachdem diese Definitionen alle durchgeführt wurden, kann nun das Datenelement „Antragsdaten“ als Prozessvariable verwendet werden. Hierzu wird das Schema-Element „Antragsdaten“ aus dem *Process Explorer* in den System-Pool „Kreditvergabe“ gezogen (Abbildung 13).

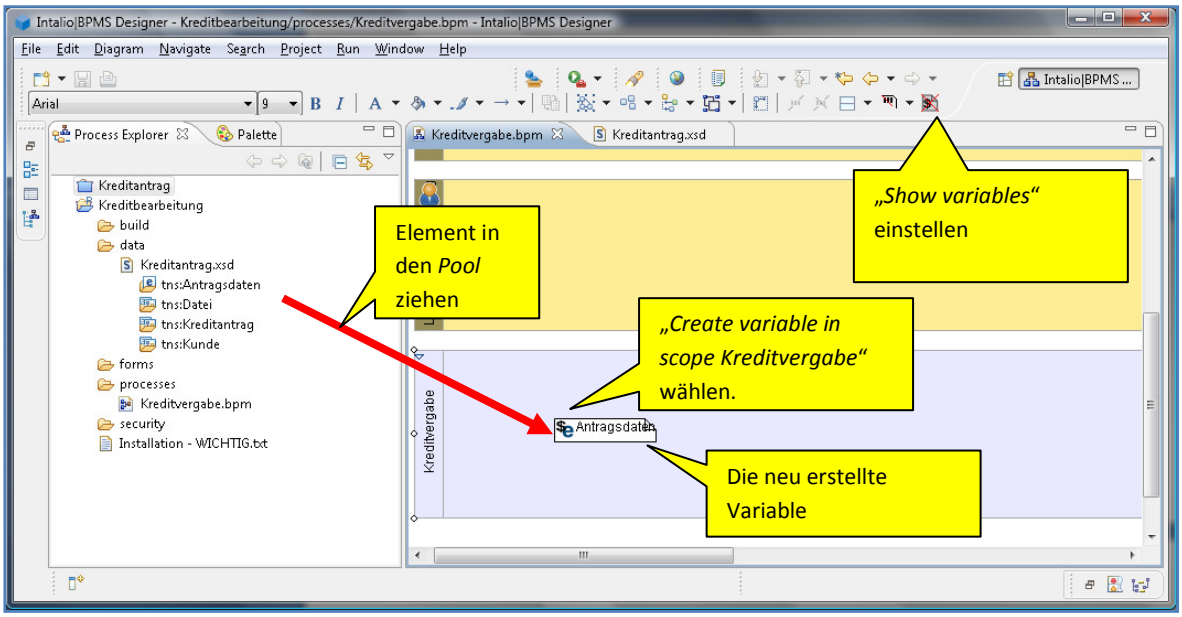

Abbildung 13: Schema-Element als Variable definieren

Um Variablen im Prozess nutzen zu können, muss zunächst das Variablenmanagement in den Diagrammen eingeschaltet werden. Hierzu ist unter „*Window* > *Preferences*“ bei „Intalio|BPMS“ im Bereich „*Advanced*“ vor „*Enable variable management in the BPMN-Diagram*“ ein Häkchen zu setzen. Über die Schaltfläche  lässt sich die Anzeige von Variablen im Prozessmodell ein- und ausschalten.

Die Datei „Kreditantrag.xsd“ lässt sich im „*Process Explorer*“ ebenso aufklappen wie ein Verzeichnis. Von hier aus wird das Element „Antragsdaten“ in das Modell in den ausführbaren *Pool* „Kreditvergabe“ gezogen. Beim Ablegen der Variable ist die Option „*Create variable in scope Kreditvergabe*“ auszuwählen.

Wichtig (da gerne übersehen)**:** Bevor die Variable verwendet werden kann, muss sie initialisiert werden. Hierzu muss das Prozessmodell im Vordergrund sein. Es wird das Fenster „*Data Editor*“ ausgewählt. Im *Data Editor* wird das Symbol für den *Pool* „Kreditvergabe“ und darunter der Eintrag „*declarations*“ aufgeklappt. Darunter findet sich ein Eintrag für die Prozessvariable „Antragsdaten“ (siehe Abbildung 14). Wenn dieser selektiert ist, muss über das Kontextmenü der rechten Maustaste „*Initialize variable content with default value*“ ausgewählt werden. Anschließend ändert sich der Eintrag zu „Antragsdaten *initialized*“.

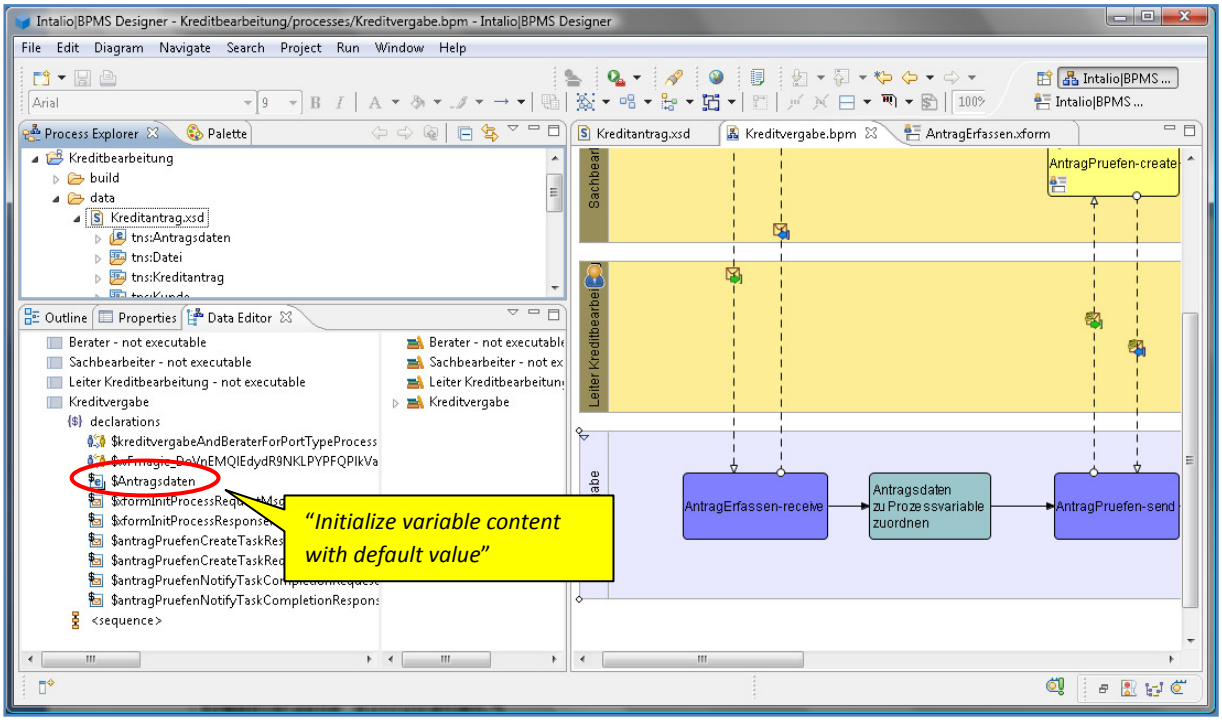

Abbildung 14: Variable initialisieren

3.7 Kontrollfluss und Benutzerdialoge

Der eigentliche Kontrollfluss wird nun im Prozessmodell grafisch modelliert. In den Kontrollfluss ist eine Reihe von Benutzer-Aktionen eingebunden. Für diese müssen jeweils Benutzerdialoge bereitgestellt werden. Im Folgenden wird der Kontrollfluss schrittweise entwickelt, wobei jeweils die benötigten Benutzerdialoge entworfen und eingebunden werden. Grundlage für die Entwicklung des Kontrollflusses ist die EPK aus Abbildung 1.

3.7.1 Dialog Antrag erfassen

Liegt ein Kreditwunsch vor, beginnt der Ablauf mit dem Erfassen des Antrags. Hierzu wird zunächst der Benutzerdialog entwickelt. Dies erfolgt mit Hilfe des im BPM *Designer* enthaltenen *Form Editors* (Abbildung 15). Hier lassen sich die gewünschten Dialog-Elemente mit der Maus in den Dialog-Entwurf ziehen und mit weiteren Angaben versehen.

Im Verzeichnis „*forms*“ wird über das Kontextmenü mittels „*New*“ > „*Workflow Form*“ ein neuer Dialog „AntragErfassen.xform“ erstellt. Zur Bearbeitung muss über „*Window*“ > „*Open Perspective*“ oder über das *Perspective*-Symbol oben rechts in die Perspektive „Intalio|BPMS *Form Editor*“ gewechselt werden. Es steht dann auf der linken Seite die „*Workflow Form Editor Palette*“ mit einer Auswahl von Dialog-Elementen zur Verfügung.

Bei der Vorstellung eines kompletten Durchlaufs des entwickelten Workflows in Kapitel 4 werden die entwickelten Dialoge im Einsatz gezeigt.

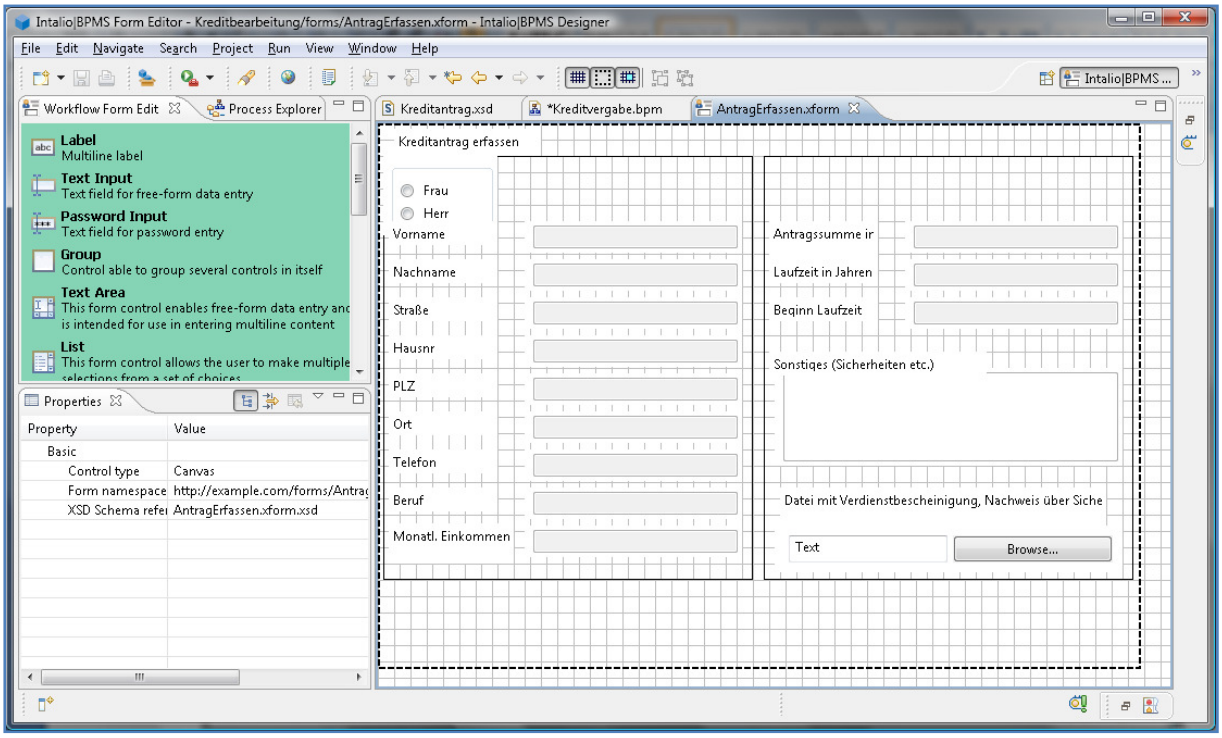

Abbildung 15: Der *Form-Editor*

Für den Dialog „Kreditantrag erfassen“ werden die nachfolgend beschriebenen Dialog-Elemente benötigt. Die Position sowie Höhe und Breite lassen sich mit der Maus anpassen. Um eine genaue Positionierung zu erreichen, ist es aber meist hilfreich, diese Werte im *Properties*-Fenster des jeweiligen Dialog-Elements einzugeben.

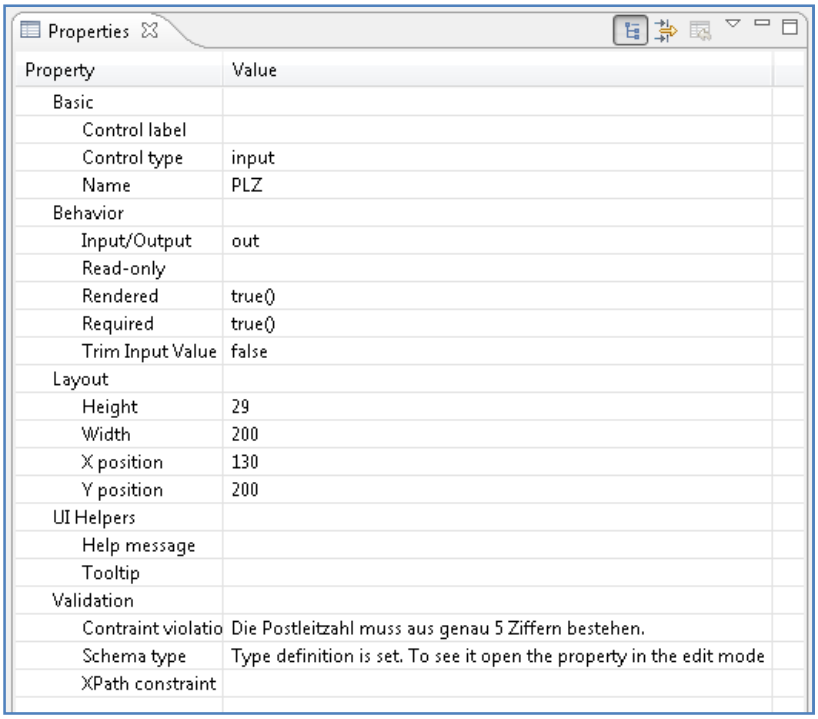

Abbildung 16: *Property*-Fenster des Eingabefeldes PLZ

Zur Gruppierung der Dialogelemente in zwei Spalten wurden zwei „*Groups*“ verwendet. Diese werden später nicht angezeigt, doch kann man die anderen Elemente darauf positionieren und dann die Spalte komplett mit allen Elementen verschieben. Es wurden eine Höhe (*height*) von 390 und eine Breite (*width*) von 340 gewählt.

Für die Überschrift sowie die Beschriftungen der Eingabefelder wurden „*Label*“ eingesetzt. Zwar kann man auch direkt bei den Eingabefeldern eine Beschriftung eingeben, doch lässt sich damit die Breite von Beschriftungen und Eingabefeldern nicht getrennt kontrollieren, so dass keine gleichmäßige Ausrichtung möglich ist. Der Text kann direkt in die *Labels* eingegeben werden. Es empfiehlt sich, den Namen immer mit „*Label_*“ beginnen zu lassen, also z. B. „*Label_Vorname*“. Wenn später auf die Inhalte des Dialogs zugegriffen werden soll, lassen sich die Eingabefelder mit den eigentlichen Inhalten dann besser von den Beschriftungs-Feldern unterscheiden.

Im Beispiel hat das *Label* für „Vorname“ in der *Group* die Position (x=5, y=60). Die y-Werte der folgenden *Labels* erhöhen sich jeweils um 35. Von einem kleineren vertikalen Abstand ist abzuraten, da später Hinweise auf falsche Eingaben ggf. unter den Eingabefeldern eingeblendet werden.

Die einzeiligen Eingabefelder sind „*Text Inputs*“. Sie erhalten als Namen sinnvollerweise die Bezeichnung ihres einzugebenden Inhaltes, also z. B. „Vorname“. Im *Properties*-Fenster sollte unter „*Input/Output*“ der Wert „*out*“ gewählt werden. Hierdurch wird angegeben, dass die Inhalte dieses Feldes nur aus dem Dialog heraus gehen (zur *Process Engine*), aber vorher von ihr keine Werte hinein gegeben werden. Weitere Möglichkeiten sind „*in*“, d. h. es gehen nur Werte in den Dialog hinein, aber keine heraus, und „*in-out*“, d. h. es gehen sowohl Werte hinein als auch wieder heraus (wenn z. B. schon Werte vorgegeben werden, die vom Benutzer des Dialogs aber wieder geändert werden können. Soll eine Eingabe verpflichtend sein, so ist unter „*Required*“ der Wert „*true()*“ einzutragen (mit den beiden Klammern). Im Beispiel wurde für alle Eingabe-Elemente mit Ausnahme der Telefonnummer und der Datei mit Verdienstbescheinigung u. ä. *true()* gewählt.

Abbildung 16 zeigt beispielhaft die Inhalte des *Property*-Fensters für das Eingabefeld der Postleitzahl. „*Constraint Violation*“ und „*Schema Type*“ werden später noch erläutert. Dort sind auch die Positions-Koordinaten angegeben, wobei sich die y-Werte für die darunter angeordneten Elemente wiederum jeweils um 35 erhöhen.

Für die Eingabe von „Sonstiges (Sicherheiten etc.)“ wird ein „*Text Area*“ verwendet, das mehrzeilige Eingaben erlaubt. Ansonsten gilt das oben für *Inputs* Gesagte.

Für die Auswahl der Anrede wird ein Dialog-Element „*Radio Button*“ benutzt. Selektiert man das Element und wählt man mit der rechten Maustaste „*Edit Items*“, so kann man die verschiedenen Auswahlmöglichkeiten definieren. Dies ist in Abbildung 17 dargestellt. Mit „*New*“ wird eine neue Auswahlmöglichkeit hinzugefügt, für die man dann in der Tabelle oben sowohl das im Dialog angezeigte *Label* als auch den bei Auswahl des betreffenden Punktes übergebene Wert eingeben kann. Im Beispiel sind die beiden Einträge jeweils gleich, d. h. wenn „Frau“ ausgewählt wird, soll auch der *String* „Frau“ an die *Process Engine* übergeben werden.

Das Dialog-Element zum Hochladen einer Datei ist ein *Upload*-Element. Auch für dieses sowie für die *Radio Buttons* ist der Wert für *Input/Output* „*out*“.

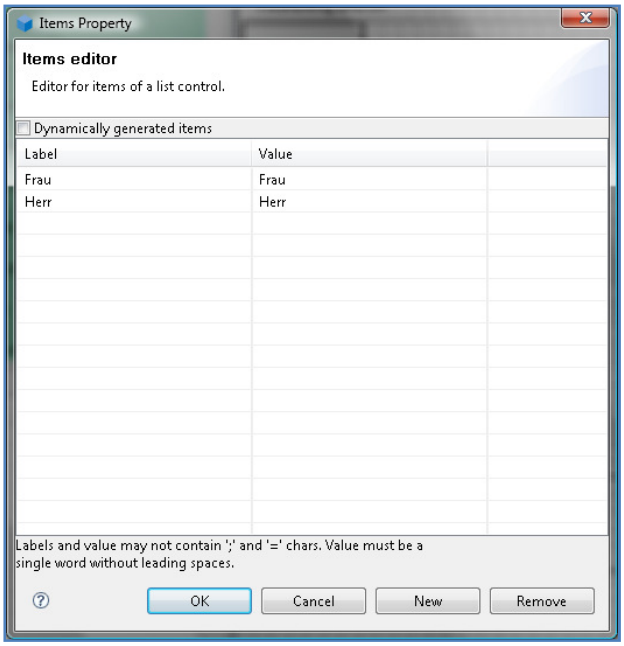

Abbildung 17: Auswahlmöglichkeiten für Radio Buttons festlegen

Wie bereits erwähnt ist es sehr empfehlenswert, die Eingabewerte auf ihre Gültigkeit gemäß den Festlegungen in Tabelle 2 zu überprüfen. Mit Hilfe des *Properties*-Fensters lassen sich für jedes Eingabe-Element der Datentyp sowie evtl. erforderliche Einschränkungen der erlaubten Werte festlegen (Abbildung 18).

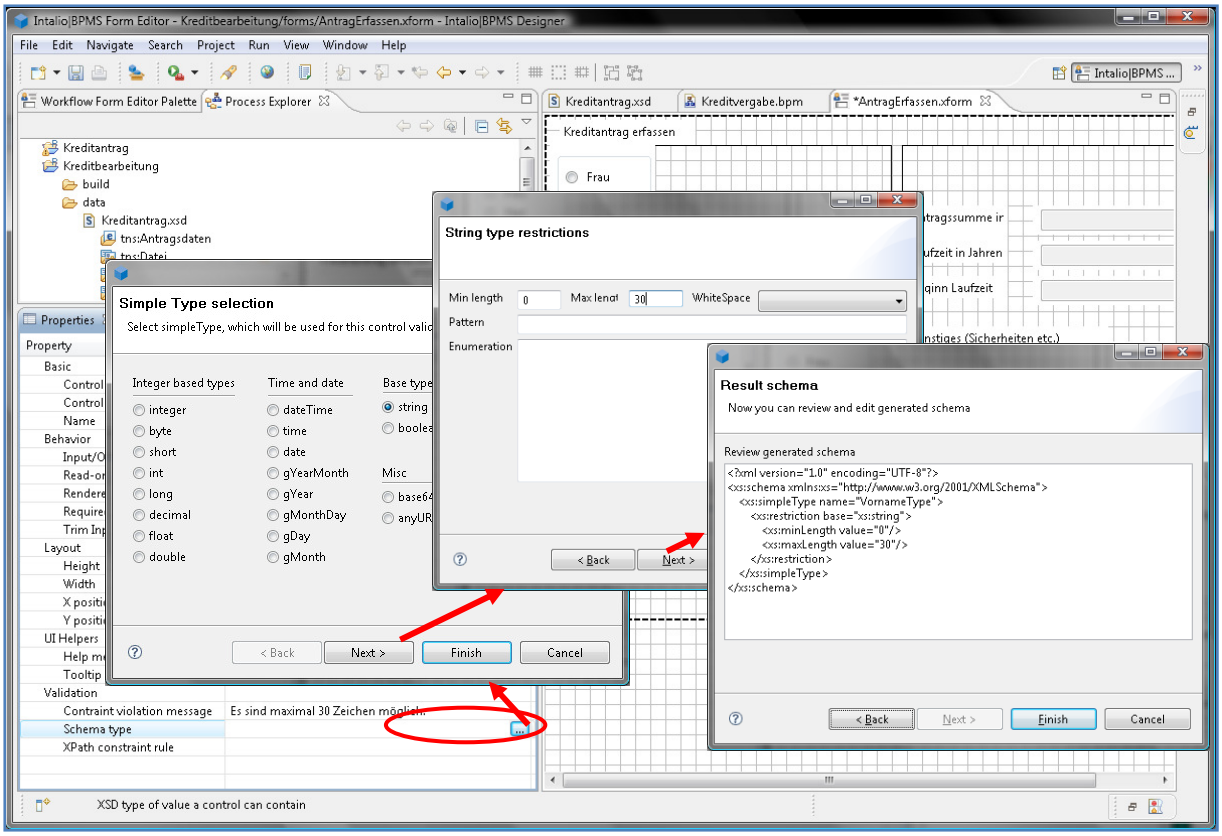

Abbildung 18: Angaben für die Eingabeprüfung erfassen

Im Feld „*Schema type*“ ist zunächst eingetragen „*Type definition is not set*“. Klickt man auf diesen Eintrag, wird ein Button mit der Aufschrift „…“ angezeigt, über den man zu den in Abbildung 18 gezeigten Dialogen gelangt. Zunächst kann man den Basisdatentyp auswählen. Für das Eingabefeld des Vornamens ist dies beispielsweise *string*. In dem über „*Next*“ folgenden Dialog kann man diesen weiter einschränken. Im angezeigten Beispiel wurde die Länge des einzugebenden Textes auf 0 bis 30 Zeichen festgelegt. Im Feld „*Pattern*“ können reguläre Ausdrücke eingegeben werden (siehe unten). Der dritte Dialog schließlich zeigt das Ergebnis in XML-Notation. XML-Experten können dies bei Bedarf noch direkt weiter bearbeiten.

Im Feld „*Constraint violation message*” (im *Property*-Fenster direkt über dem *Schema type*) sollte noch ein kurzer Text eingegeben werden, der später den Benutzer bei einer Fehleingabe über die zulässigen Eingabewerte informiert.

Für die meisten Eingabefelder des Beispiels wird der Datentyp *string* mit einer Länge von 0 bis 30 Zeichen verwendet. Der *String* bei „Sonstiges (Sicherheiten etc.)“ darf 0 bis 2000 Zeichen umfassen.

Die Geldbeträge bei dem monatlichen Einkommen sind vom Typ *decimal* mit einem Minimalwert von 0 und einem Maximalwert von 100 000 (Einkommen) bzw. 1 Mio. (Antragssumme). Außerdem wurde die Zahl der erlaubten Stellen hinter dem Komma (bzw. dem in Intalio nach der amerikanischen Schreibweise zu verwendenden Punkt) auf zwei beschränkt.

Die Einschränkung auf zwei Nachkommastellen musste direkt im XML-Code mittels des Auszeichners „*fractionDigits*“ vorgenommen werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
	<simpleType name="EinkommenType">
		<restriction base="decimal">
			<minInclusive value="0"/>
			<maxInclusive value="100000"/>
			<fractionDigits value="2"/>
		</restriction>
	</simpleType>
</schema>
```

Listing 4: Festlegung von Nachkommastellen

Die Laufzeit ist ein *integer.* Als Werte sind minimal 0 und maximal 30 erlaubt. „Beginn Laufzeit“ ist vom Typ *date*. Für den Datei-*Upload* sowie die *Radio Buttons* müssen keine Datentypen festgelegt werden.

Spezielle Restriktionen gelten noch für die erlaubten *strings* bei der Postleitzahl sowie der Telefonnummer. Deutsche Postleitzahlen bestehen aus genau fünf Ziffern. Da auch die 0 am Anfang erlaubt ist, kann hier nicht der Datentyp *integer* verwendet werden, denn bei der Speicherung und Anzeige einer Zahl werden führende Nullen weg gelassen. Es ist also ein *string* erforderlich.

Der reguläre Ausdruck „\d{5}“ legt fest, dass genau fünf Ziffern eingegeben werden können. „\d“ steht für eine Ziffer. Der Wert innerhalb der geschweiften Klammer gibt an, wie oft der vorangehende Ausdruck wiederholt wird.

Telefonnummern sind noch etwas komplizierter. Sie können beispielsweise nur aus Ziffern und Leerzeichen bestehen. Es ist aber auch möglich, dass Vorwahl und Nummer durch einen Schrägstrich getrennt werden, oder dass die Vorwahl eingeklammert wird. Will man alle diese Möglichkeiten zulassen, kann dies etwa durch den folgenden regulären Ausdruck erreicht werden.

```
((\d|\s)*) | (\((\d|\s)*\)(\d|\s)*)) | ((\d|\s)*\/(\d|\s)*)
```

„\d“ steht für eine Ziffer, „\s“ für ein Leerzeichen. „\/“ steht für den Schrägstrich, „\(“ und „\)“ für öffnende und schließende Klammern. „|“ trennt zwei Alternativen voneinander, und „*“ gibt an, dass der vorangehende Ausdruck beliebig oft wiederholt werden darf.

Auch für Telefonnummern sollte die Gesamtlänge z. B. auf 30 Zeichen beschränkt werden. Dies geschieht wie bei den anderen *strings* gemäß Abbildung 18.

Eine genaue Erläuterung der in XML Schema verwendbaren Datentypen sowie der regulären Ausdrücke findet sich in XML-Lehrbüchern.

Nach dem Speichern des Dialogs finden sich im *forms*-Verzeichnis zwei Dateien: Die gerade angelegte Datei mit der Endung „xform“ sowie eine zweite, vom System generierte Datei mit der Endung „xform.xsd“. Bei dieser Datei handelt es sich wieder um eine XML Schema-Datei, die die Datentypen und Felder des Dialogs enthält. Diese kann ebenso mit dem XML *Schema Editor* betrachtet werden, sollte aber keinesfalls direkt bearbeitet werden!

Im Test wurden für manche Dialog-Elemente nicht gleich alle *Properties*-Felder dargestellt (z. B. der *Schema Type* für *Uploads*). Dies konnte behoben werden, indem der Dialog geschlossen und noch

einmal geöffnet wurde. Um spätere Probleme zu vermeiden, wird empfohlen, die Dialoge vor dem *Deployen* einmal zu schließen und wieder zu öffnen.

3.7.2 Integration des Dialogs in den Workflow als auslösende Aktivität

Um den Dialog in den Workflow zu integrieren, zieht man die „xform"-Datei mit der Maus in das Prozessmodell auf den *Pool* der zugeordneten Rolle „Berater". Dabei muss angegeben werden, dass es sich um eine prozessauslösende Aktivität handelt. Der Editor legt nun automatisch eine Aktivität an, die mit dem gewählten Dialog verbunden ist (Abbildung 19).

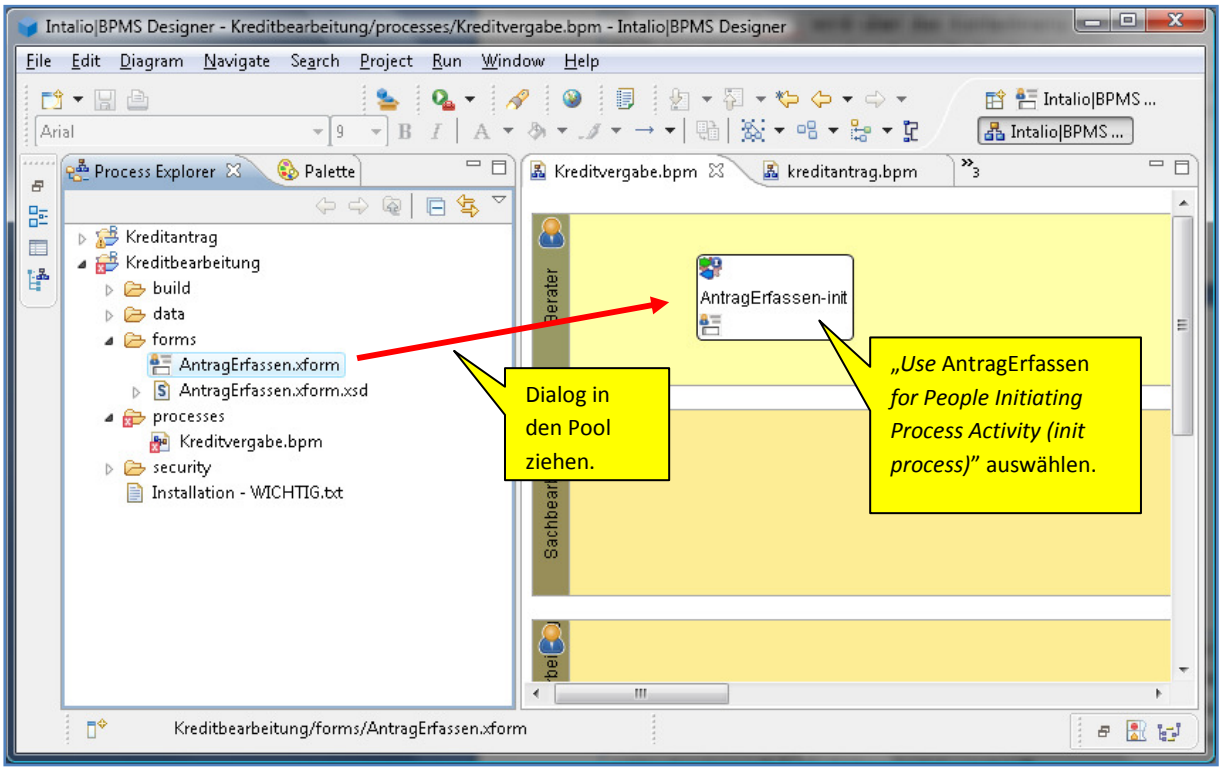

Abbildung 19: Zuordnen des Dialogs zum *Pool*

Beim Ablegen des Formulars auf dem Pool muss aus dem angezeigten Auswahlmenü der Punkt "*Use* AntragErfasssen *for People Initiating Process Activity (initProcess)*" gewählt werden. Der BPM *Designer* erzeugt dann eine Aktivität, mit deren Hilfe der spätere Benutzer den Kreditbearbeitungs-prozess starten kann. Diese Aktivität ist bereits mit dem erstellten Dialog verbunden.

Unter „*Properties*" der Aktivität „AntragErfassen-*init*" wird unter „*Workflow*" im Feld „*Description*" (Beschreibung) die Bezeichnung eingetragen, die später beim Benutzer in der Liste der von ihm durchführbaren Prozesse erscheint, beispielsweise „Kreditantrag erfassen".

Um den Dialog im Workflow verarbeiten zu können, muss das System das Ergebnis der vom Berater durchgeführten Aktivität entgegennehmen. Hierzu wird im System-*Pool* „Kreditverarbeitung" eine weitere Aktivität „AntragErfassen-*receive*" angelegt. Dieser wird nun mittels „*Message Connections*" (Nachrichtenverbindungen) mit „AntragErfassen-*init*" verbunden. Da die Aktivität vom Berater gestartet wird, fließt zunächst eine Nachricht von „AntragErfassen-*init*" zu „AntragErfassen-*receive*". Wurde das Ergebnis erfolgreich entgegengenommen, so fließt eine zweite Nachricht zurück, die nur

mitteilt, dass alles geklappt hat. Das Ergebnis ist in Abbildung 20 dargestellt. Die Benutzeraktivität ist hier gelb eingefärbt, die Systemaktivitäten zur Kommunikation mit einer Benutzeraktivität blau, was aber wiederum keine Auswirkung auf die Funktionalität hat.

Um diesen Nachrichtenfluss zu modellieren, muss *zuerst* eine „*Message Connection*“ von „Antrag-Erfassen-*init*“ zu „AntragErfassen-*receive*“ gezogen werden (im Bild von oben nach unten), *anschließend* eine zweite *Message Connection* in die Gegenrichtung von unten nach oben.

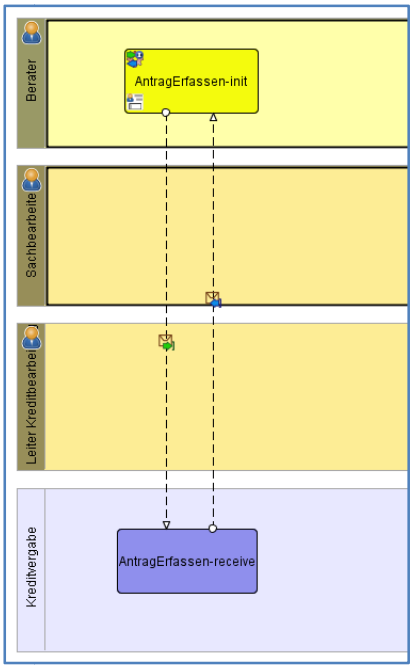

Abbildung 20: Nachrichtenfluss zwischen Dialog-Aktivität und System-Aktivität

3.7.3 Deployment der ersten Aktivität

Das erstellte Modell kann nun bereits auf den Server *deployed* und getestet werden. Unter *Deployment* (wörtlich: ausbringen, verteilen) versteht man, dass alle zu dem Workflow gehörenden Dateien automatisch an die richtigen Stellen auf dem Server kopiert und ggf. kompiliert werden und dass erforderliche Einstellungen getätigt werden, so dass der Workflow anschließend ausgeführt werden kann. Voraussetzung für das Deployment ist, dass der *Server* läuft.

Das *Deployment* erfolgt direkt aus dem BPM *Designer* heraus, entweder über „*Project*“ > „*Deploy Project*“, oder über das Symbol . Im folgenden Dialog (Abbildung 21) müssen alle benötigten Dialoge (im Moment „AntragErfassen.xform“ und „AntragErfassen.xform.xsd“) sowie *Workflow*-modelle (im Moment „Kreditvergabe.bpm“) ausgewählt werden. Die Schemadatei „Kreditantrag.xsd“ kann ebenfalls *deployed* werden, wird im Moment noch nicht benutzt. Die weiteren Einstellungen brauchen bei einer lokalen *Server*-Installation nicht verändert zu werden.

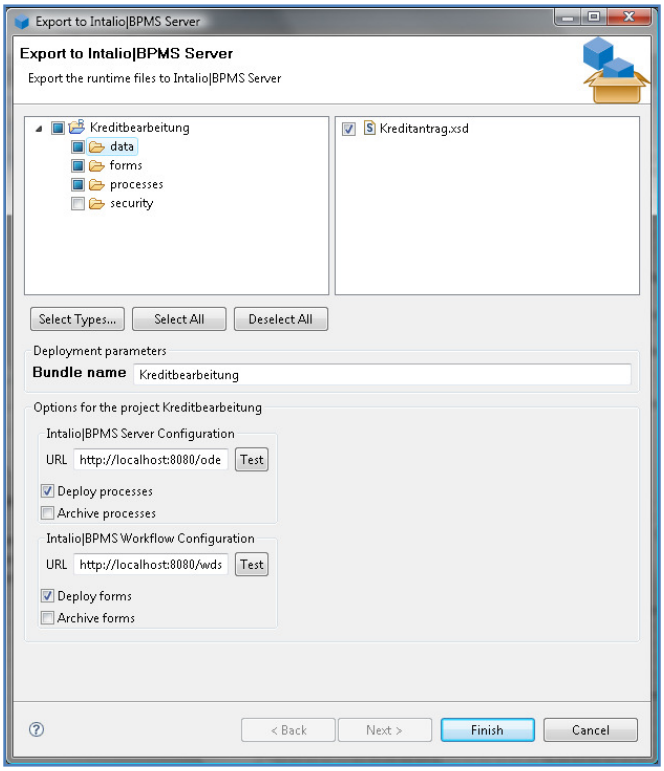

Abbildung 21: Einstellungen für das *Deployment*

Ist das *Deployment* erfolgreich abgeschlossen (Abbildung 22), so kann der *Workflow* auf dem Server getestet werden.

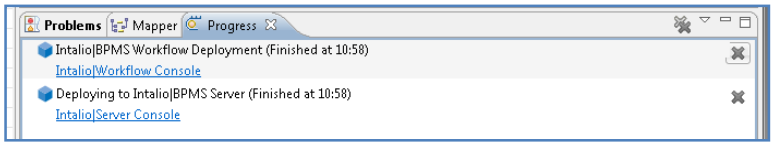

Abbildung 22: Das *Deployment* ist erfolgreich abgeschlossen

Zunächst soll der Prozess in der Administrationskonsole angesehen werden (Abbildung 23).

Hierzu wird einem *Web-Browser* die Adresse http://localhost:8080/bpms-console eingegeben. Hier sollte man sich als Administrator anmelden (Benutzer: „intalio/admin“, Passwort „changeit“). Bei den ersten Zugriffen auf den *Server* kann die Ladezeit recht lang sein, später verbessert sich dies.

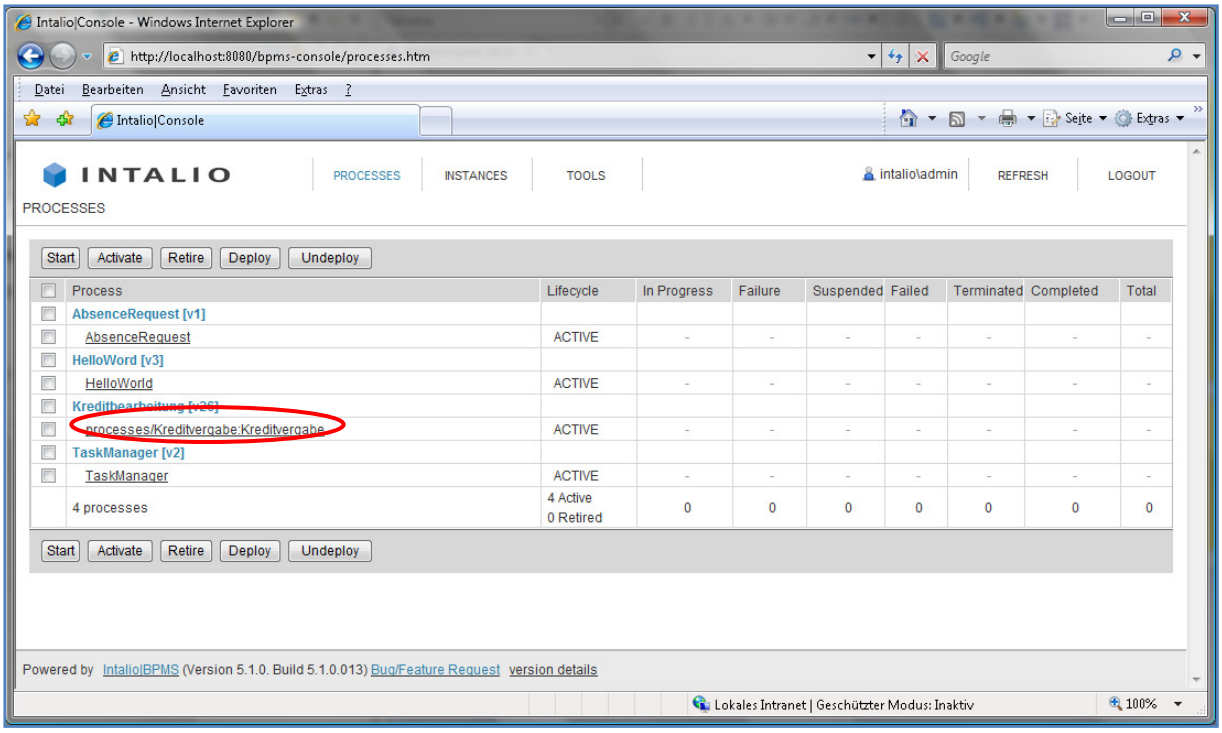

Abbildung 23: Der Prozess in der Administrations-Konsole

Der Prozess ist als *„Active“* gekennzeichnet, d. h. diese Version kann momentan ausgeführt werden. Wird der Prozess erneut *deployed*, so wird eine neue Version erstellt, und die alte Version erhält den Zustand *„retired“* (außer Dienst). Zu dem Prozess lassen die Details ansehen: Die Prozessgrafik und sämtliche Dateien, die zu dem *Workflow* gehören (Abbildung 24).

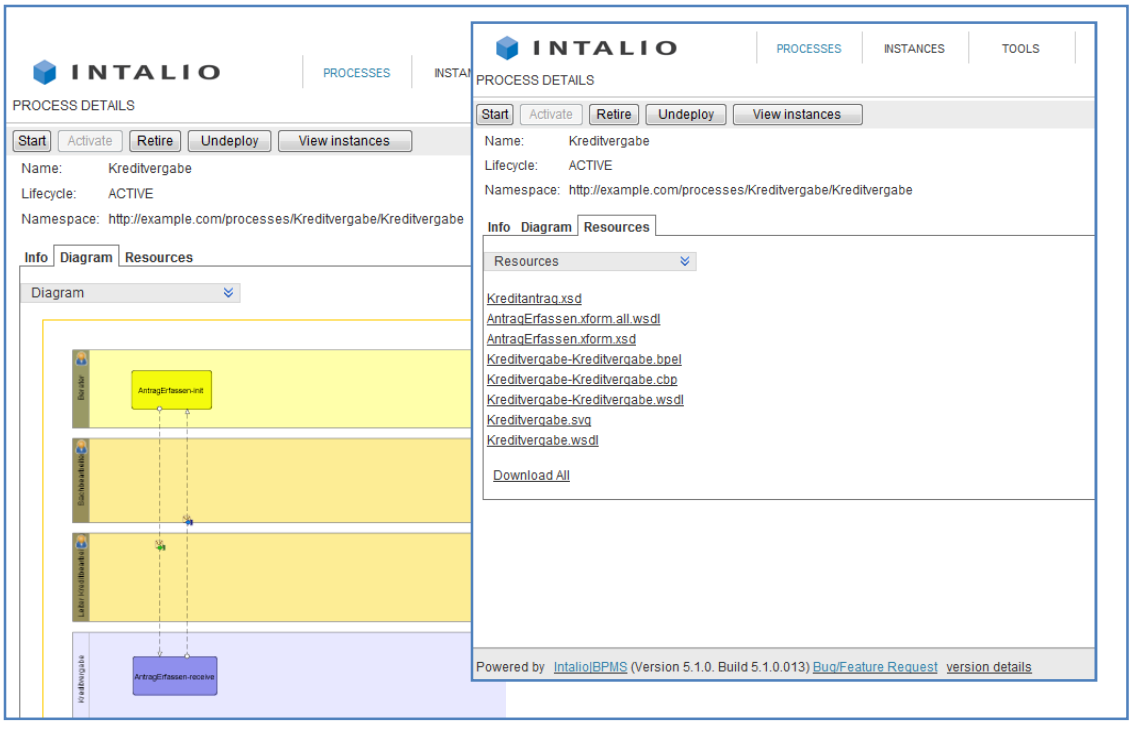

Abbildung 24: Details zum Prozess in der Administrations-Konsole

Bei den Dateien handelt es sich zunächst um die bereits bekannten Schema-Dateien zur Definition der Datenstrukturen sowie eine svg-Datei (*Scalable Vector Graphics*) mit der Prozessgrafik. Die wsdl-Dateien (*Web Service Definition Language*) beschreiben die *Web Service*-Schnittstellen, die zum Aufruf der Dialoge bzw. zum Aufruf des gesamten Prozesses erforderlich sind. Die bpel-Datei (*Business Process Execution Language*) beschreibt den modellierten Ablauf in einer für die *Process Engine* geeigneten Struktur. Diese Datei liegt außerdem noch in kompilierter Form als cpb-Datei vor (*Compiled BPEL Process*). Für eine nähere Beschäftigung mit diesen Dateiformaten wird auf Literatur zum Thema *Web Services* verwiesen.

3.7.4 Testen des ersten Aktivität

Zum Testen des Prozesses wird die Benutzungs-Oberfläche aufgerufen, wo eine Anmeldung mit einer Berater-Rolle erfolgt. Hier kann sich der Benutzer alle Prozesse anzeigen lassen, die von ihm gestartet werden können (Abbildung 25). Meldet man sich hingegen als ein Benutzer mit einer Sachbearbeiter-Rolle an, wird der Prozess nicht angezeigt, da der Prozess immer nur von einem Berater gestartet werden kann.

Hierzu wird im Web-Browser die Adresse http://localhost:8080/ui-fw eingegeben. Hat man sich als Administrator noch nicht ausgeloggt, ist dieser in der Oberfläche ebenfalls bereits eingeloggt. Um die verschiedenen Rollen zu testen, sollte man sich zunächst ausloggen und anschließend als Berater einloggen, z. B. als Benutzer „becker“ (Passwort: „password“). Zur Ansicht der Prozesse muss auf den Reiter „*Processes*“ geklickt werden.

Bei einem falschen Passwort oder Benutzernamen wird man übrigens nicht darauf hingewiesen, es wird lediglich der Anmeldedialog erneut angezeigt.

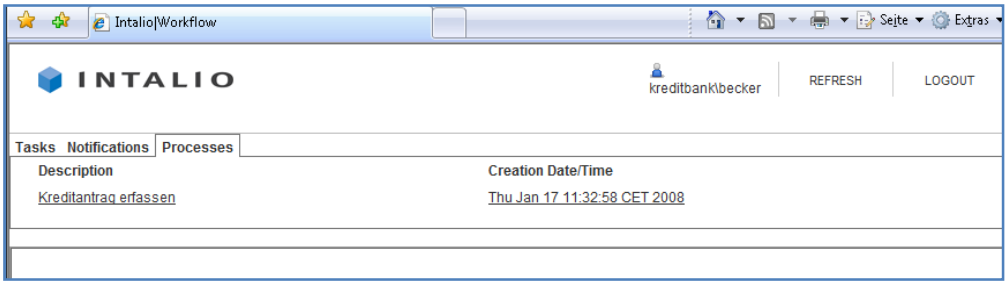

Abbildung 25: Der Prozess in der Benutzungs-Oberfläche

Durch Klicken auf den Prozess „Kreditantrag erfassen“ wird der soeben erstellte Dialog geöffnet (Abbildung 26). Dort sind die Mussfelder (bei denen die Eigenschaft *Required* den Wert *true()* hat), rot dargestellt um auf die Notwendigkeit einer Eingabe hinzuweisen. Sobald ein Wert eingegeben worden ist, verschwindet die rote Markierung.

Für den Beginn der Laufzeit wird aufgrund des eingestellten Datentyps *date* ein Kalendersymbol angezeigt, über das sich das Datum aus einem aufklappbaren Kalender auswählen lässt.

Unterhalb von Eingabeelementen, bei denen es etwas Spezielles zu beachten gibt, ist ein rotes Ausrufezeichen eingeblendet. Fährt man mit dem Mauszeiger auf ein Ausrufezeichen, so werden die eingegebenen Hinweistexte angezeigt.

Nach Eingabe eines gültigen Wertes verschwindet das betreffende Ausrufezeichen. Umgekehrt erscheinen Ausrufezeichen bei Eingabe ungültiger Werte, beispielsweise bei Eingabe eines Namens mit mehr als 30 Zeichen (Abbildung 27). Auch die Restriktionen bei Geldsummen, Telefonnumer und Postleitzahl werden überprüft.

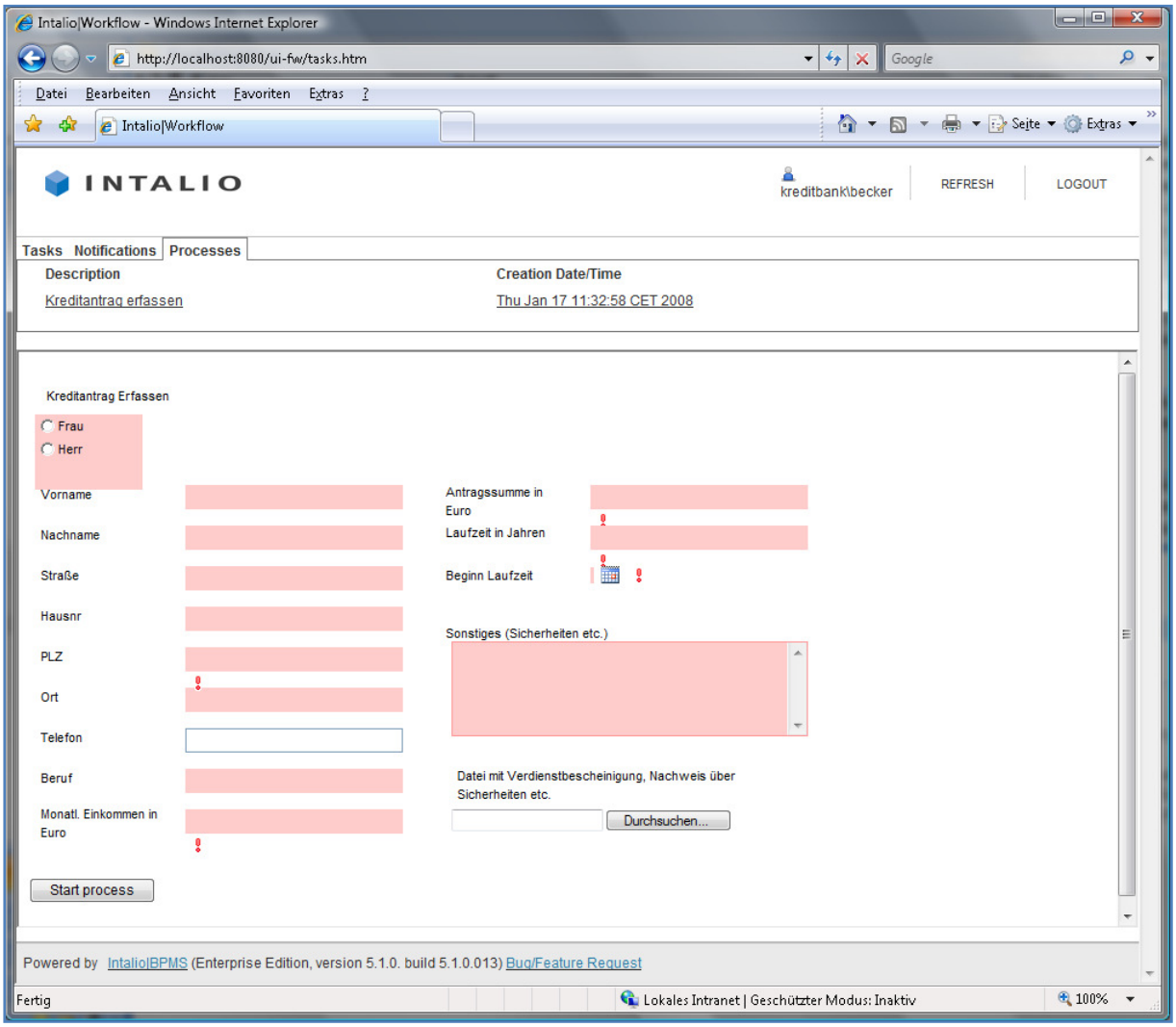

Abbildung 26: Der Dialog „Antrag erfassen“

Solange noch Eingaben fehlen oder ungültig sind, kann der Prozess nicht gestartet werden. Erst wenn alle Eingaben in Ordnung sind, lässt sich der Prozess über die Schaltfläche „*Start process*“ in Gang setzen.

Abbildung 27: Eingabeüberprüfung mit HInweis

Gelegentlich traten in umfangreicheren Tests mit mehrmaligem *Deployen* und zwischenzeitlich fehlerhaften Dialogen noch Ausrufezeichen auf, obwohl der Dialog korrekt und alle Eingaben in Ordnung waren. Oder aber es traten keine Ausrufezeichen auf, der Prozess ließ sich aber trotzdem nicht starten. Derartige Probleme ließen sich in der Regel durch einen Neustart des *Servers* beheben.

Nachdem der Prozess erfolgreich gestartet wurde, tut sich erwartungsgemäß nichts Weiteres, da bisher keine Folgeaktivitäten modelliert wurden.

In der Administrationsoberfläche lässt sich die durchgeführte Prozess-Instanz näher untersuchen. Ruft man dort nach Anmeldung als „intalio/admin“ wieder die Prozessdetails auf, erhält man über die Schaltfläche „*View instances*“ eine Liste aller Instanzen dieses Prozesses. Jedesmal, wenn der Prozess von einem der Berater gestartet wird, wird eine neue Instanz des Prozesses angelegt. Beim ersten Mal ist nur eine Instanz vorhanden, die gerade durchgeführte.

Klickt man auf diese Instanz, lassen sich sämtliche Einzelheiten ihrer Durchführung samt den übergebenen Daten untersuchen. So werden nicht nur der Zustand „*Completed*“ (abgeschlossen) sowie Beginn- und Endzeitpunkt angezeigt, sondern auch die zahlreichen bei der Abarbeitung aufgetretenen internen Ereignisse. Unter „*Data*“ lassen sich die in den Nachrichtenflüssen übergebenen Daten ansehen.

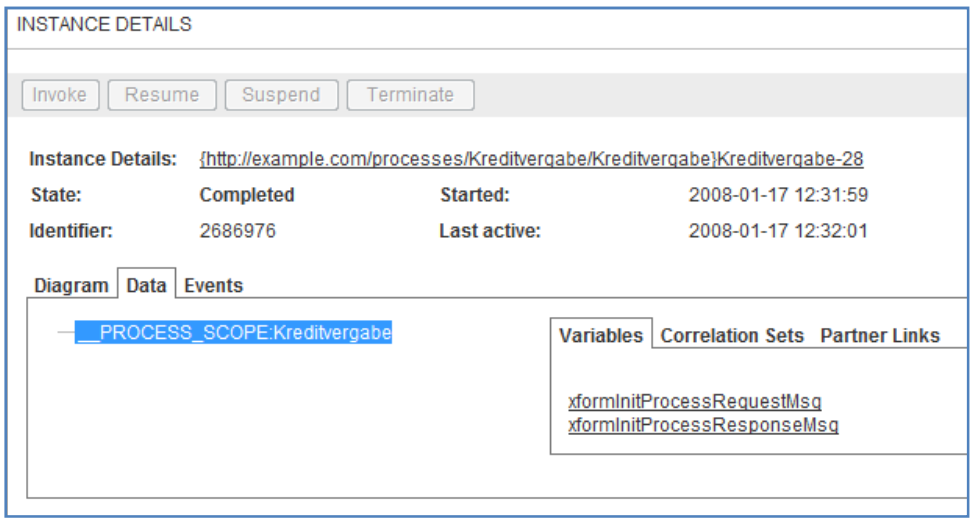

Abbildung 28: Details der Prozessinstanz: Daten

So sind in Abbildung 28 die Variablen „*xforminitProcessRequestMsg*“ und „*xforminitiProcessResponseMsg*“ zu sehen. Diese enthalten die Nachrichten, die zwischen dem Dialog „Antrag erfassen“ und der *Process Engine* ausgetauscht wurden.

Sieht man sich „*xforminitProcessRequestMsg*“ an, so findet man dort erwartungsgemäß die Daten, die der Berater gerade eingegeben hat (Abbildung 29). Diese Nachricht bewegt sich entlang des zuerst modellierten Nachrichtenflusses (in der Grafik nach unten). Die in Abbildung 30 gezeigte Antwort der Process Engine gemäß dem im Modell nach oben zeigenden Nachrichtenfluss ist in der „*xforminitiProcessResponseMsg*“ enthalten. Sie beinhaltet lediglich die Status-Information „OK“ zur Bestätigung, dass die Nachricht ordnungsgemäß empfangen und verarbeitet wurde.

```
<group_Kreditdetails>
  <upload34 filename="x.txt" mediatype="text/plain" upload-id="upload34">http://zwh242-ta4.ds.fh-
kl.de:8080/wds/attachments/57bf640d-52b1-4a5b-9642-deb5c5b0dbba/x.txt</upload34>
  <Antragssumme>5000.00</Antragssumme>
  <Laufzeit>5</Laufzeit>
  <Label_Laufzeit />
  <Label_Antragssumme />
  <Label_Datei />
  <Sonstiges>Frau Mustermann verfügt über eine Eigentumswohnung im Wert von 80.000 Euro.</Sonstiges>

  <Beginn>2008-03-01</Beginn>
  <Label_Beginn />
</group_Kreditdetails>
<group_Kundendetails>
  <Einkommen>2500</Einkommen>
  <Label_Einkommen />
  <Beruf>Workflow-Tester</Beruf>
  <Label_Beruf />
  <Telefon>(0999) 999 999</Telefon>
  <Label_Telefon />
  <Ort>Neustadt</Ort>
  <Label_Ort />
  <PLZ>99999</PLZ>
  <Label_PLZ />
  <Hausnummer>17</Hausnummer>
  <Label_Hausnr />
  <Strasse>Mozartstr.</Strasse>
  <Label_Strasse />
```

Abbildung 29: Die Nachricht mit den eingegebenen Daten (Ausschnitt)

Nachdem der *Workflow* mit der ersten Aktivität erfolgreich getestet wurde, kann nun mit der Modellierung des weiteren Ablaufs fortgefahren werden. Es wird empfohlen, nach jeder Erweiterung des Modells einen entsprechenden Test durchzuführen, da sich Fehler recht schwer lokalisieren lassen, wenn ein umfangreiches *Workflow*-Modell komplett erstellt wurde und dann nicht richtig ausgeführt wird.

Message: xformInitProcessResponseMsg

Part: root

```
<initProcessResponse xmlns="http://example.com/forms/AntragErfassen/xform"
xmlns:xform="http://example.com/forms/AntragErfassen/xform">
  <xform:status>OK</xform:status>
  <xform:errorCode>  </xform:errorCode>
  <xform:errorReason>  </xform:errorReason>
  <xform:taskMetaData>
    <xform:nextTaskId>  </xform:nextTaskId>
    <xform:nextTaskURL>  </xform:nextTaskURL>
  </xform:taskMetaData>
</initProcessResponse>
```

Abbildung 30: Die Bestätigung der *Process Engine*

3.7.5 Zuordnung der Daten zur Prozessvariablen

Die übergebenen Antragsdaten sollen nun in die in Abschnitt 3.5 erstellten Prozessvariablen geschrieben werden. Zwar ist es auch möglich, ohne Verwendung einer Prozessvariablen die *Output*-Daten eines Dialogs direkt als *Input*-Daten eines weiteren Dialogs zu verwenden, doch würde dies im vorliegenden Fall aufgrund der Schleife im Prozessablauf nicht genügen (siehe Abbildung 31). Die Aktivitäten „Antrag prüfen“ und „Antrag ergänzen“ können mehrfach durchlaufen werden. Dabei können manche Daten jeweils geändert bzw. ergänzt werden. Würde man in der Aktivität „Antrag prüfen“ die *Output*-Daten aus „Antrag erfassen“ als *Input*-Daten verwenden, so würde bei späteren Schleifendurchläufen die *Output*-Daten von „Antrag ergänzen“ nicht berücksichtigt. Daher wird die Prozessvariable „Antragsdaten“ verwendet. Jede Änderung der Daten wird in dieser Variablen gespeichert, und die Dialoge erhalten die aktuellen Daten der Variablen als *Input*-Daten zugeordnet.

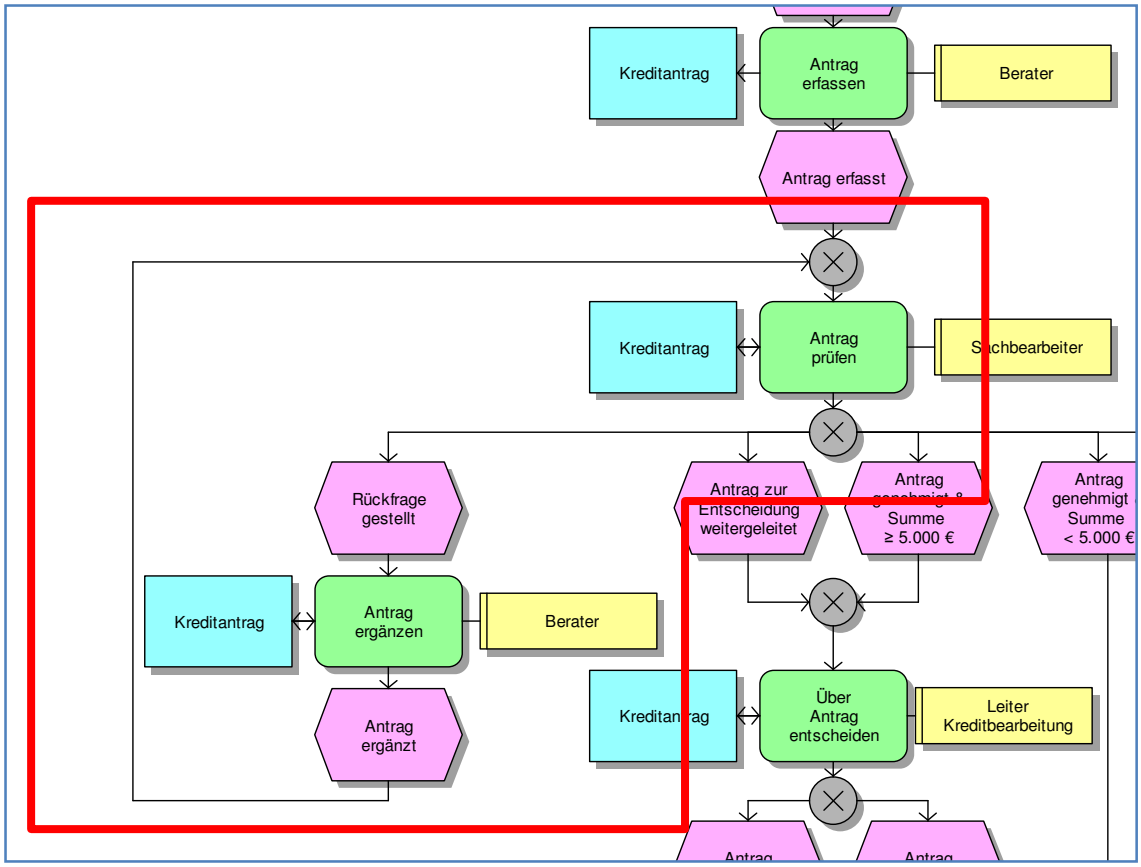

Abbildung 31: Die Schleife „Antrag prüfen“ und „Antrag ergänzen“ kann mehrfach durchlaufen werden (Ausschnitt aus Abbildung 1)

Um die Zuordnung der Daten zu der Variablen zu modellieren wird eine weitere Aktivität benötigt. Diese wird im System-*Pool* „Kreditvergabe“ über eine Kontrollflusskante mit der ersten Aktivität verbunden (Abbildung 32). Die Zuordnung und ggf. Transformation der Daten (das *Mapping*) wird im Fenster „*Mapper*“ festgelegt. Wenn die neu angelegte Aktivität selektiert ist, dann enthält dieses Fenster alle Daten, auf die in dieser Aktivität zugegriffen werden kann. Links sind mögliche *Input*-Daten dargestellt, rechts mögliche *Output*-Daten (untere Hälfte von Abbildung 32).

Nun müssen die gewünschten *Input*- und *Output*-Daten miteinander verbunden werden. In unserem Fall sollen Daten, die von dem Dialog „Antrag erfassen“ übermittelt wurden, der Variablen „Antragsdaten“ übergeben wurden. Es handelt sich um den Dialog mit dem der Prozess gestartet wird, und

die Daten werden (wie oben in der Administrationsoberfläche gesehen wurde) über die *Request*-Nachricht an das System übermittelt. Auf der linken Seite im Mapper findet sich die Datenstruktur „*$xformInitProcessRequestMsg.root*“, die somit als Quelle verwendet wird.

Klappt man diese Datenstruktur sowie „Antragsdaten“ auf, so findet man die Bezeichnungen aller Dialogelemente des Dialogs „Antrag erfassen“ wieder. Auf der rechten Seite wird die Variable „Antragsdaten“ aufgeklappt. Nun müssen die Datenfelder der linken Seite mit denen der rechten Seite verbunden werden. In Abbildung 33 ist dies bereits für „Einkommen“ und „Beruf“ durchgeführt worden. Auch die weiteren Felder müssen nun verbunden werden.

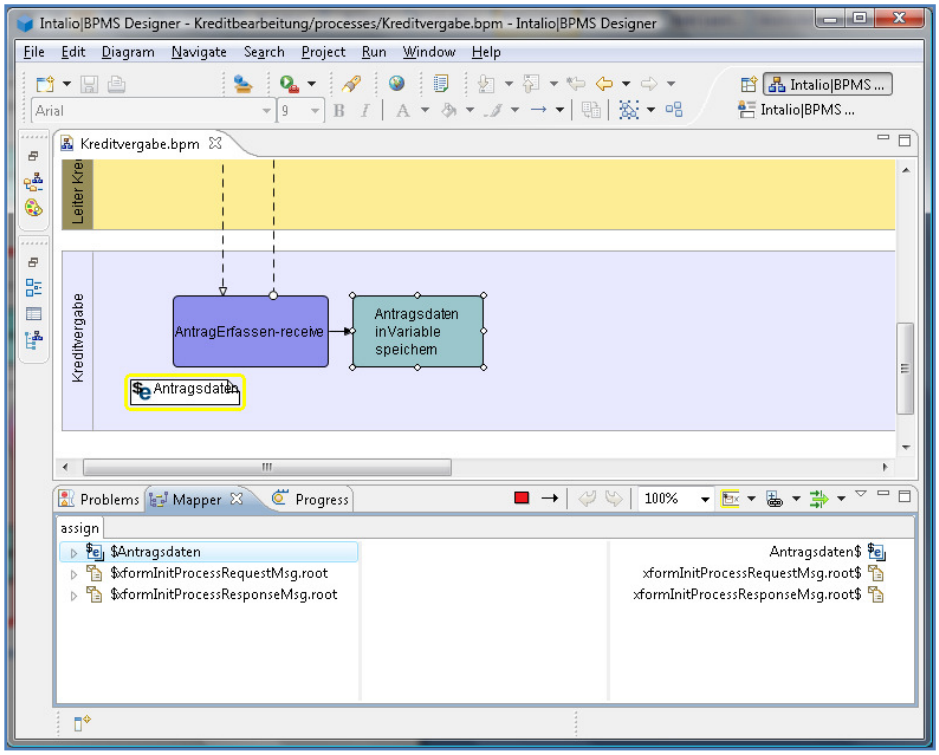

Abbildung 32: Aktivität für die Zuordnung der Daten zur Prozessvariablen

Es wird jeweils das Feld der linken Seite mit der Maus angeklickt, anschließend das entsprechende Feld auf der rechten Seite.

Es wird nun auch deutlich, warum die Bezeichnungen der Felder beim Anlegen des Dialogs wichtig waren. Sie erleichtern es im *Mapper*, die richtigen Felder zu finden. So können beispielsweise alle Felder, deren Bezeichnungen mit „*Label_“* beginnen, ignoriert werden. Die Verwendung der *Groups* hilft nun ebenfalls. Sie können auf- und zugeklappt werden und machen so die Darstellung übersichtlicher.

Das Antragsdatum soll automatisch ermittelt werden. Dies ist in Abbildung 34 gezeigt. Die Funktion „*current-date()“* liefert jeweils das momentane Datum. Wie auf S. 19 erläutert, sollen Datumsangaben wegen der besseren Ausgabemöglichkeit als *String* im deutschen Datumsformat gespeichert werden. Hierzu werden Tag, Monat und Jahr einzeln aus dem von *current-date*() gelieferten augenblicklichen Datum ermittelt. Die Funktion „*day-from-date*()“ liefert beispielsweise den Tag in Form einer Zahl. Die Funktion „*concat*()“ fügt diese einzelnen Angaben nun zu einem Datum im deutschen Format zusammen, wobei zur Trennung Punkte eingefügt werden.

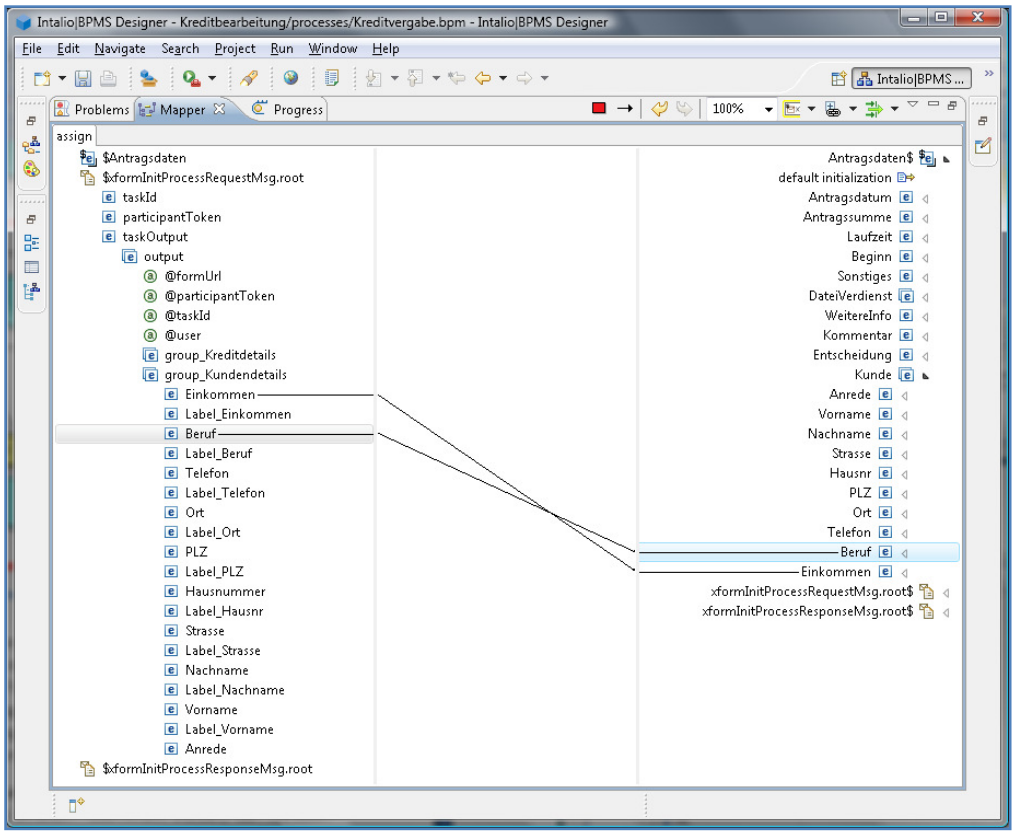

Abbildung 33: Zuordnung der Datenfelder

Im *Mapper* können diese und weitere *XPath*-Funktionen verwendet werden. Für weitere Informationen zu diesem Thema wird wiederum auf die zahlreich verfügbare XML-Einführungsliteratur verwiesen.

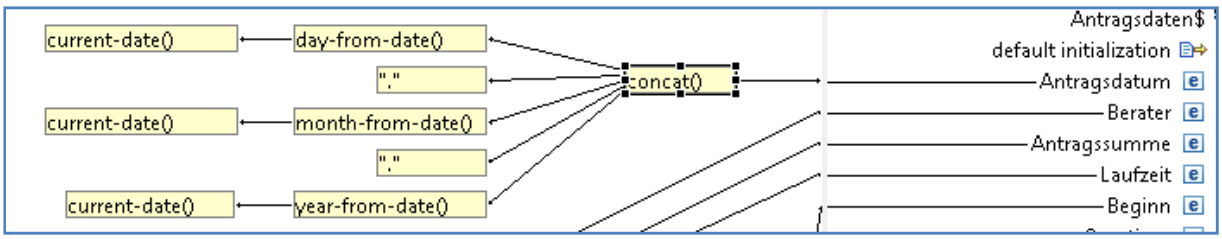

Abbildung 34: Eintragen des Antragsdatums und Umwandlung in einen *String*

Xpath-Funktionen werden mit Hilfe des Buttons  eingefügt.

Über „*Window*“ > „*Show View*“ > „*Mapper Palette*” lässt sich eine Übersicht der verfügbaren Funktionen anzeigen. Wenn man die *Mapper Palette* in ein anderes Fenster als den *Mapper* schiebt, dann kann man diese Funktionen auch per *Drag&Drop* in den *Mapper* ziehen.

Bei der Funktion *concat*() ist es wichtig, die zu verbindenden Elemente nacheinander in der richtigen Reihenfolge mit *concat*() zu verbinden. Die Verbindungen werden immer von links nach rechts gezogen.

Die verwendete Ermittlung des Antragsdatums hat übrigens einen kleinen Schönheitsfehler: Das augenblickliche Datum wird dreimal ermittelt. Da dies unmittelbar hintereinander geschieht, wird sich das Datum in der Zwischenzeit normalerweise nicht ändern. Tritt jedoch der unwahrscheinliche Fall auf, dass dazwischen ein Monatswechsel stattfindet (z. B. vom 31.1. auf den 1.2.), wird ein falscher Wert in Antragsdatum eingetragen, wenn beim Tag noch der alte Wert eingetragen wird (im Beispiel 31), beim Monat aber bereits der neue Wert (im Beispiel 2, so dass sich als Datum der 31.2. ergibt). Korrekt wäre es, das Datum nur einmal zu ermitteln. Allerdings lassen sich leider keine drei Verbindungen von „*current-date*()“ ziehen. Daher müsste man den Wert zunächst in einer Variablen zwischenspeichern und anschließend erst zerlegen. Darauf wurde wegen des zusätzlichen Aufwandes, und weil das Eintreten des Problems recht unwahrscheinlich ist, verzichtet.

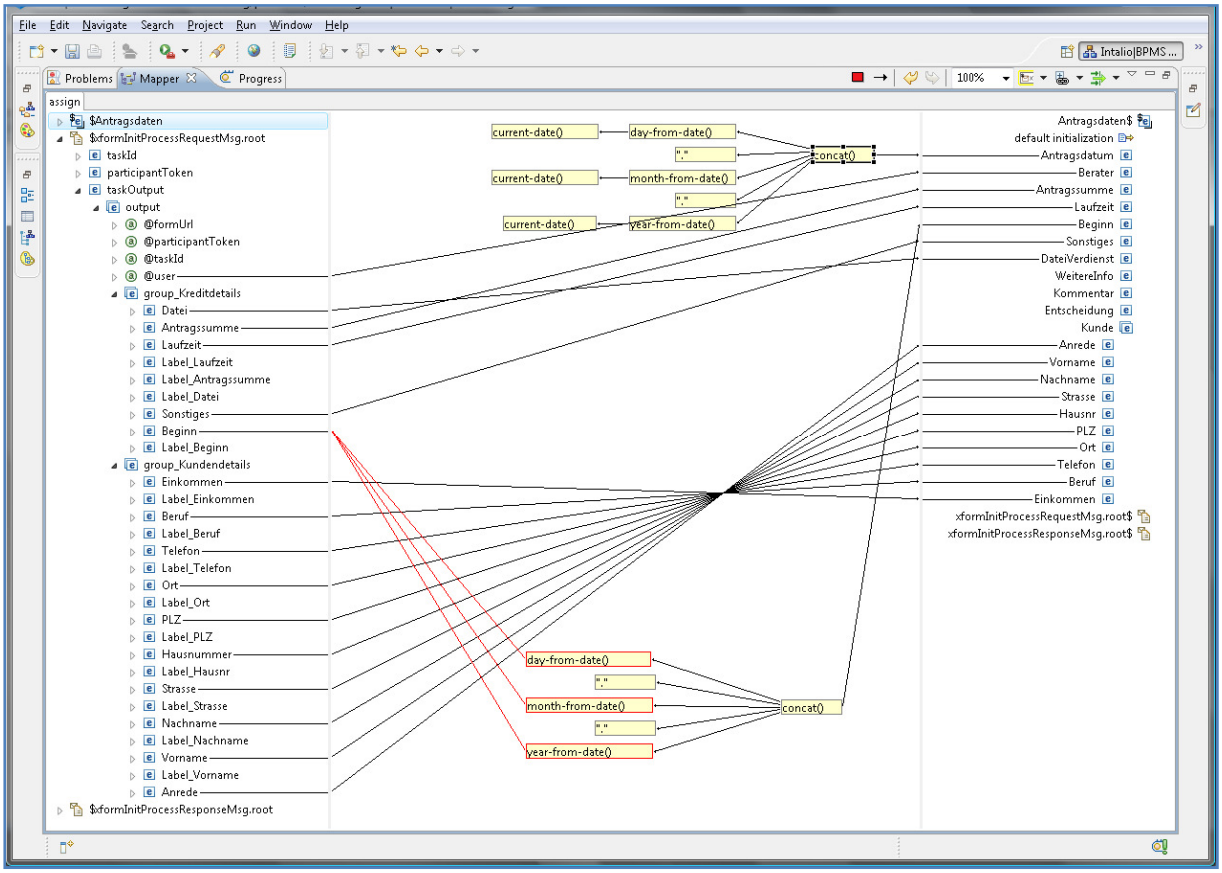

Abbildung 35: Das gesamte *Mapping*

Das gesamte Mapping zeigt Abbildung 35. Das Datum für den Beginn der Laufzeit wird auf die gleiche Weise wie das Antragsdatum in einen *String* umgewandelt.

Ausgehend von einem Variablenfeld können erfreulicherweise mehrere Verbindungen gezogen werden, so dass sich Tag, Monat und Jahr von „Beginn“ auf die in Abbildung 35 gezeigte Weise ermitteln lassen.

Als „Berater“ soll der Benutzer eingetragen werden, der den Antrag erfasst hat. Die Information über den Bearbeiter liefert das BPMS im Feld „*@user*“ mit. Dieser wird dem Feld „Berater“ in der Variablen zugeordnet.

Leider ist es nicht so ohne weiteres möglich, auf den gesamten Namen des Benutzers zuzugreifen, der in die Datei „*security.xml*“ eingetragen wurde (vgl. Abschnitt 3.2). Intalio|BPMS bietet zwar einen speziellen Webservice an, der einem derartige Informationen liefern kann, doch gibt es in der verwendeten Version Probleme beim Zugriff. Daher wird nur der Benutzername in das Feld „Berater“ eingetragen.

3.7.6 Modellierung der Schleife

Eine einfache Modellierung der Schleife wie in der EPK aus Abbildung 1, wo ein Pfad aus einer Verzweigung einfach weiter vorne wieder in eine Zusammenführung läuft, ist im BPM *Designer* nicht möglich. Stattdessen muss eine als Schleife durchlaufener Unterprozess („*Looping Subprocess*“) verwendet werden. Dieser ist durch ein kleines Schleifensymbol am unteren Rand gekennzeichnet, sowie durch ein Minuszeichen, über das man den Unterprozess auf- und zuklappen kann um die darin enthaltenen Aktivitäten ein- und auszublenden. Alle Aktivitäten innerhalb der Schleife werden in diesen Unterprozess hineingelegt.

Für eine Schleife muss festgelegt werden, unter welchen Bedingungen sie (noch einmal) durchlaufen wird und unter welchen Bedingungen sie beendet wird. Im vorliegenden Fall wird „Antrag prüfen“ auf jeden Fall einmal durchlaufen. Falls eine Rückfrage gestellt wird, wird anschließend „Antrag ergänzen“ durchgeführt, und die Schleife wird noch einmal ausgeführt. In allen anderen Fällen, in denen keine Rückfrage gestellt wird, wird die Schleife beendet, und der Ablauf geht mit „Über Antrag entscheiden“ weiter.

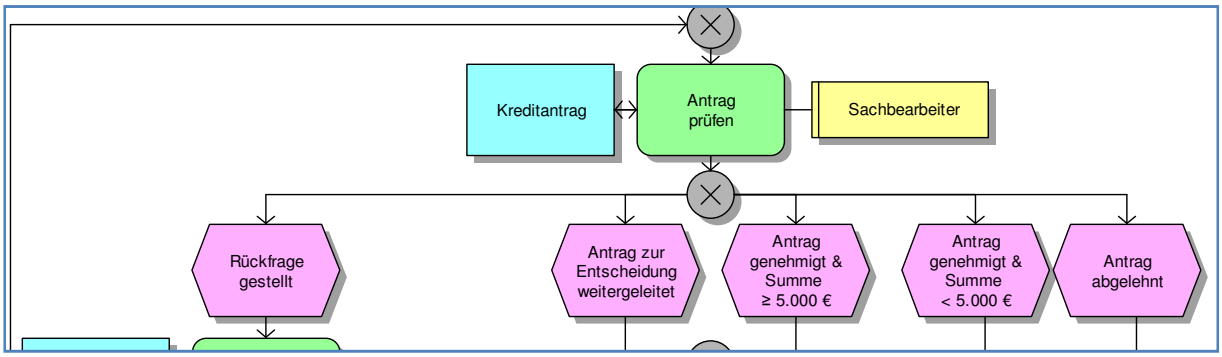

Abbildung 36: Mögliche Endereignisse der Funktion „Antrag prüfen“ (Ausschnitt aus Abbildung 1)

Es muss also eine Möglichkeit für die *Process Engine* geben, festzustellen, ob eine Rückfrage gestellt wurde. Ob eine Rückfrage erforderlich ist, entscheidet der Sachbearbeiter in der Funktion „Antrag prüfen“. Das Ergebnis der Entscheidung kann ebenfalls in der Variable „Antragsdaten“ gespeichert werden. Die Datenstruktur der Antragsdaten enthält bereits ein Feld „Entscheidung“, das hierfür mitverwendet werden kann. Bisher können dort nur die Werte „genehmigt“ und „abgelehnt“ zugeordnet werden (vgl. Abschnitt 3.5). Es muss daher noch ein Wert „rückfrage“ hinzugefügt werden.

Bei genauer Betrachtung der Funktion „Antrag prüfen“ und ihrer Endereignisse in Abbildung 36 stellt man fest, dass es noch eine vierte Entscheidungsmöglichkeit gibt: Der Antrag kann zur Entscheidung weitergeleitet werden. Der Wert „weitergeleitet“ wird daher im XML *Schema Editor* als vierter Wert zum Datenfeld „Entscheidung“ hinzugefügt (Abbildung 37).

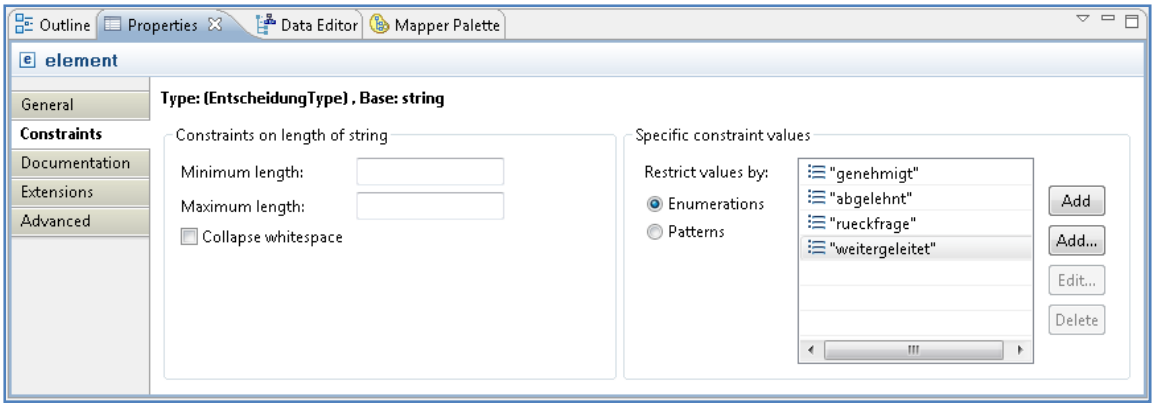

Abbildung 37: Erweiterung der erlaubten Werte für das Feld „Entscheidung“

Damit kann nun die Bedingung für den Schleifenabbruch formuliert werden. Die Schleife soll solange durchgeführt werden, bis das Feld „Entscheidung“ der Variablen „Antragsdaten“ einen anderen Wert als „rueckfrage“ enthält. Der Schleifentyp (*„Loop type“*) wird daher als *„Repeat Until“* („wiederhole bis“) festgelegt. Eine solche Schleife wird solange wiederholt, bis eine bestimmte Bedingung erfüllt ist. Diese Bedingung wird als logischer Ausdruck wiederum mit Hilfe von *XPath*-Funktionen modelliert (Abbildung 39).

Der *Loop type* wird im *Properties*-Fenster des *Loop Subprocess* festgelegt (Abbildung 38).

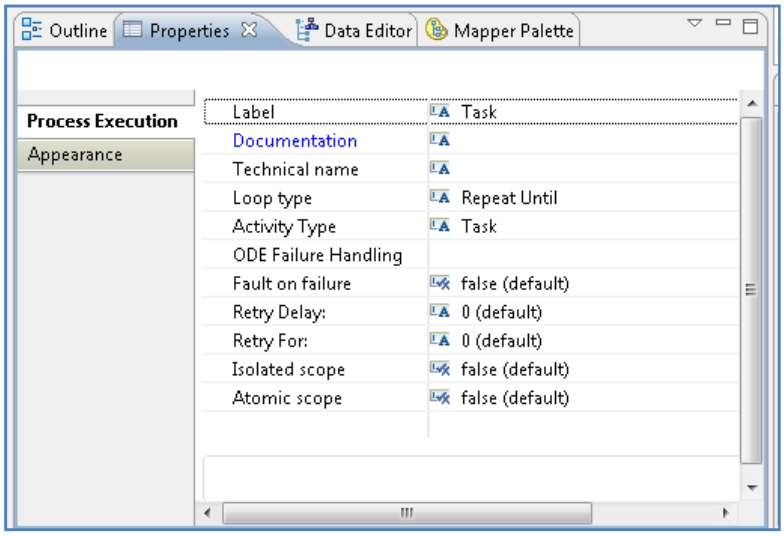

Abbildung 38: Festlegen des Schleifentyps

Bei der Formulierung der Bedingung im *Mapper* (der *Loop Subprocess* muss selektiert sein) wird zunächst der Inhalt des Feldes „Entscheidung“ mit dem Wert „rückfrage“ verglichen. Ein direkter Vergleich auf Gleichheit erwies sich als problematisch. Daher wird mit *„contains*()“ überprüft, ob der Wert „rueckfrage“ in dem Inhalt von „Entscheidung“ enthalten ist. Hierbei ist wichtig, dass zuerst „Entscheidung“ und anschließend der *String* „rueckfrage“ mit *„contains*()“ verbunden wird.

Da die Schleife wiederholt werden soll, bis „Entscheidung“ den Wert „rueckfrage“ *nicht* enthält, wird die Funktion *„not“* verwendet. Wichtig ist jeweils, dass die Verbindungen von links nach rechts gezogen werden.

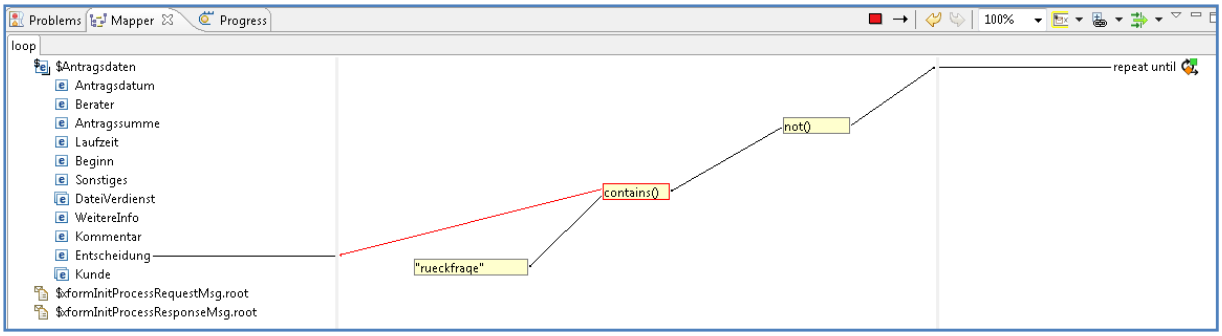

Abbildung 39: Bedingung zum Abbruch der Schleife

Damit diese modellierte Logik später funktioniert, muss bei der Modellierung der Funktion „Antrag prüfen“ sichergestellt werden, dass in das Feld „Entscheidung“ die richtigen Werte eingetragen werden.

3.7.7 Dialog Antrag prüfen

Im Benutzer-Dialog für die Funktion „Antrag prüfen“ sollten sicherlich die Antragsdaten angezeigt werden. Der Dialog ist daher dem Dialog der Funktion „Antrag erfassen“ recht ähnlich. Eine einfache Kopie eines Dialogs ist aber aufgrund der verschiedenen generierten Dateien und der verschiedenen Abhängigkeiten darin, nicht möglich.

Es gibt aber folgenden Weg, einen Dialog zu kopieren. Die beiden Dateien „AntragErfassen.xform“ und „AntragErfassen.xform.xsd“ im Verzeichnis „*forms*“ werden ***außerhalb*** des BPM Designers im *Windows Explorer* kopiert und in „AntragPruefen.xform“ bzw „AntragPruefen.xform.xsd“ umbenannt. Anschließend werden diese neuen Dateien mit einem *Editor* geöffnet. Mit dem *Editor* werden alle Vorkommen von „AntragErfassen“ in den beiden Dateien durch „AntragPruefen“ ersetzt, und die Dateien werden gespeichert.

Die beschriebenen Schritte sollten nicht innerhalb des BPM *Designers* durchgeführt werden, da dort sofort verschiedene Dateien im *Build*-Verzeichnis angelegt werden, die unter Umständen nicht korrekt an die Änderungen angepasst werden.

Nun wechselt man wieder in den BPM *Designer*. Über das Konetxtmenü der rechten Maustaste auf dem Projekt „Kreditbearbeitung“ wird „*Refresh*“ („Auffrischen“) ausgewählt. Anschließend wird der neue Dialog „AntragPruefen“ im *Process Explorer* angezeigt, und er kann im *Form Editor* geöffnet und weiter bearbeitet werden.

In diesen Dialog sollen die Antragsdaten übernommen, dort jedoch nicht mehr geändert werden. Daher wird für die Eingabefelder im *Properties*-Fenster jeweils der *Input/Output*-Wert in „*in*“ geändert, und der Wert für „*Read-only*“ (nur Lesen, kein Ändern der Werte) wird auf „*true*()“ gesetzt.

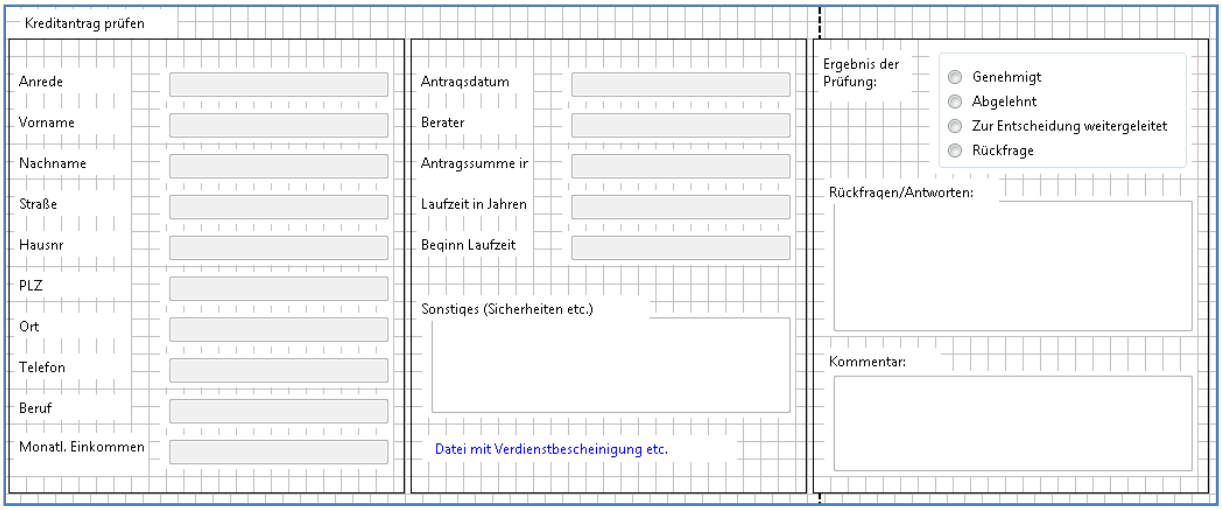

Abbildung 40: Entwurf des Dialogs „Kreditantrag prüfen“

Abbildung 40 zeigt den fertigen Dialog. Es sind folgende weitere Änderungen gegenüber dem Dialog „Antrag erfassen“ hinzugekommen:

- Überschrift in „Kreditantrag prüfen“ geändert.
- Anrede: Da keine Auswahl mehr zu treffen ist, wird die Anrede als *string* angezeigt.
- Antragsdatum vom Typ *string.*
- Berater vom Typ *string.*
- Beginn Laufzeit ist nun ebenfalls vom Typ *string.*
- Eine *group,* als „*group*_Antragsbearbeitung“ bezeichnet, die die folgenden Elemente enthält.
- Datei mit Verdienstbescheinigung etc.: Hier soll kein Hochladen mehr erfolgen, sondern eine hochgeladene Datei zum Ansehen angeboten werden, daher wird ein Dialog-Element vom Typ „*link*“ verwendet.
- Ergebnis der Prüfung: *Radio Buttons*. *Input/Output* ist hier *out, Required* ist *true*().
- Rückfragen/Antworten und Kommentar: *Input/Output* ist *in-out*, so dass bei späteren Schleifendurchläufen die vorher gemachten Eingaben bestehen bleiben und ergänzt oder verändert werden können.

3.7.8 Integration des Dialogs „Antrag prüfen“ in den Workflow

„AntragPruefen.xform“ wird nun mit der Maus aus dem *Process Explorer* in das Diagramm auf den *Pool* „Sachbearbeiter“ gezogen. Im angezeigten Auswahlmenü ist der Punkt „*Use* AntragPruefen *for People Activity (create and complete)*“ auszuwählen.

Im Gegensatz zu „Antrag erfassen“, mit dem der Prozess gestartet wird, werden für diesen Dialog automatisch zwei Aktivitäten angelegt. Die eine, „AntragPruefen-*create*“, dient dazu, eine Aufgabe für den Benutzer zu erzeugen und in die Liste seiner Aufgaben einzutragen. Die zweite, „AntragPruefen-*complete*“ dient zum Ausfüllen und Abschließen des Dialogs.

Entsprechend müssen im System-*Pool* „Kreditbearbeitung“ auch zwei Aktivitäten angelegt werden: Eine zur Übermittlung des Dialogs an die Benutzungsoberfläche des Sachbearbeiters, die andere zum Empfang der Daten aus dem fertig bearbeiteten Dialog. Entsprechend den vom BPMS *Designer*

automatisch vergebenen Namen „AntragPruefen-*create*“ und „AntragPruefen-*complete*“ wurden die analogen Bezeichnungen „AntragPruefen-*send*“ und „AntragPruefen-*receive*“ gewählt.

Wichtig ist wieder die Richtung und Reihenfolge der Nachrichtenflüsse: Bei der ersten Aktivität geht die Initiative vom System aus. Hier geht der Nachrichtenfluss zuerst vom System-*Pool* zum Benutzer-*Pool* (in unserer Darstellung von unten nach oben), anschließend in die Gegenrichtung. Bei der zweiten Aktivität ist es umgekehrt. Hier geht die Initiative vom Sachbearbeiter aus (wenn er den Dialog ausfüllt), entsprechend geht der erste Nachrichtenfluss vom Benutzer-*Pool* zum System-*Pool* (von oben nach unten) und der zweite zurück. Abbildung 41 zeigt das erweiterte Prozessmodell.

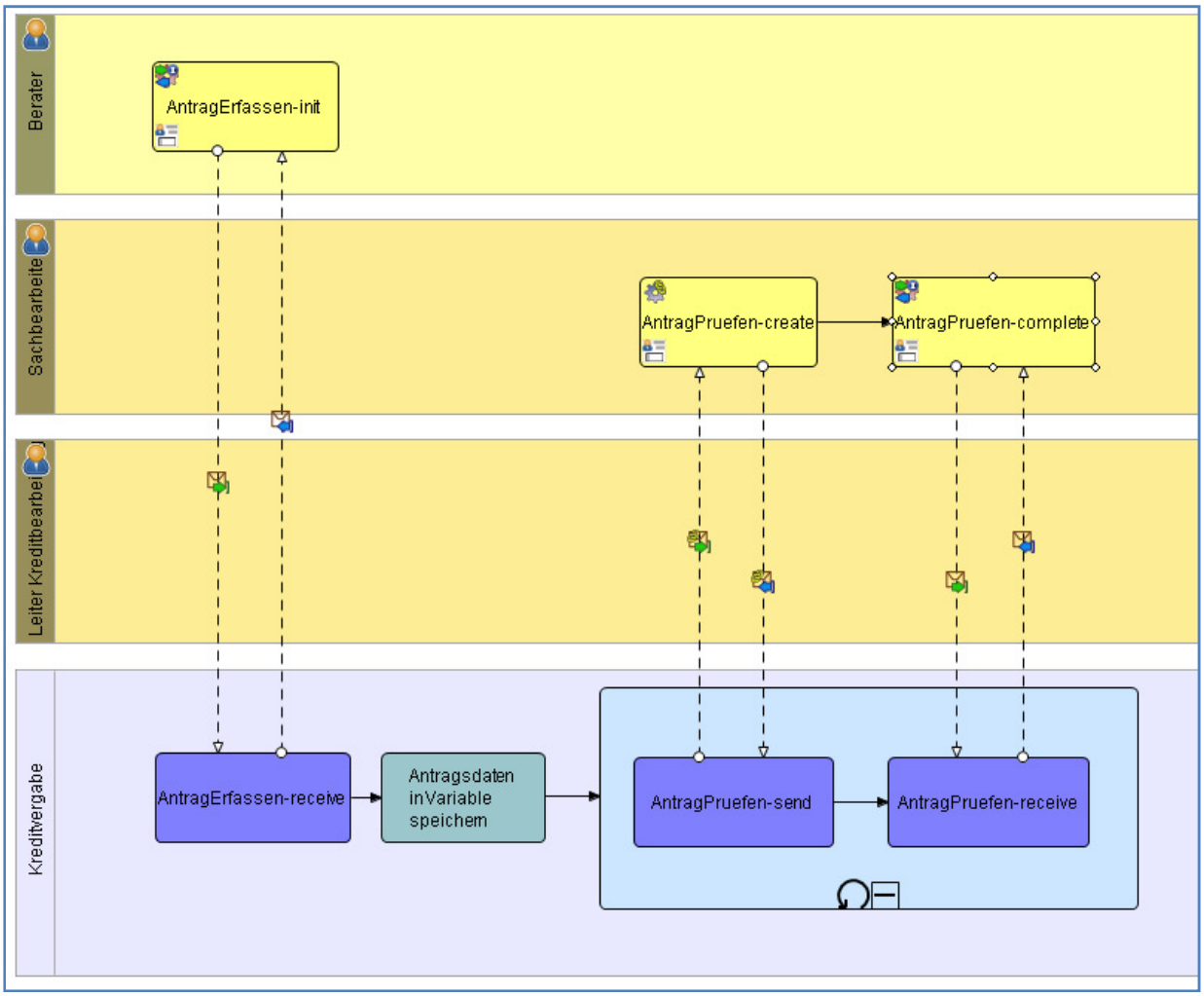

Abbildung 41: Integration des Dialogs „Antrag pruefen“ in den *Workflow*

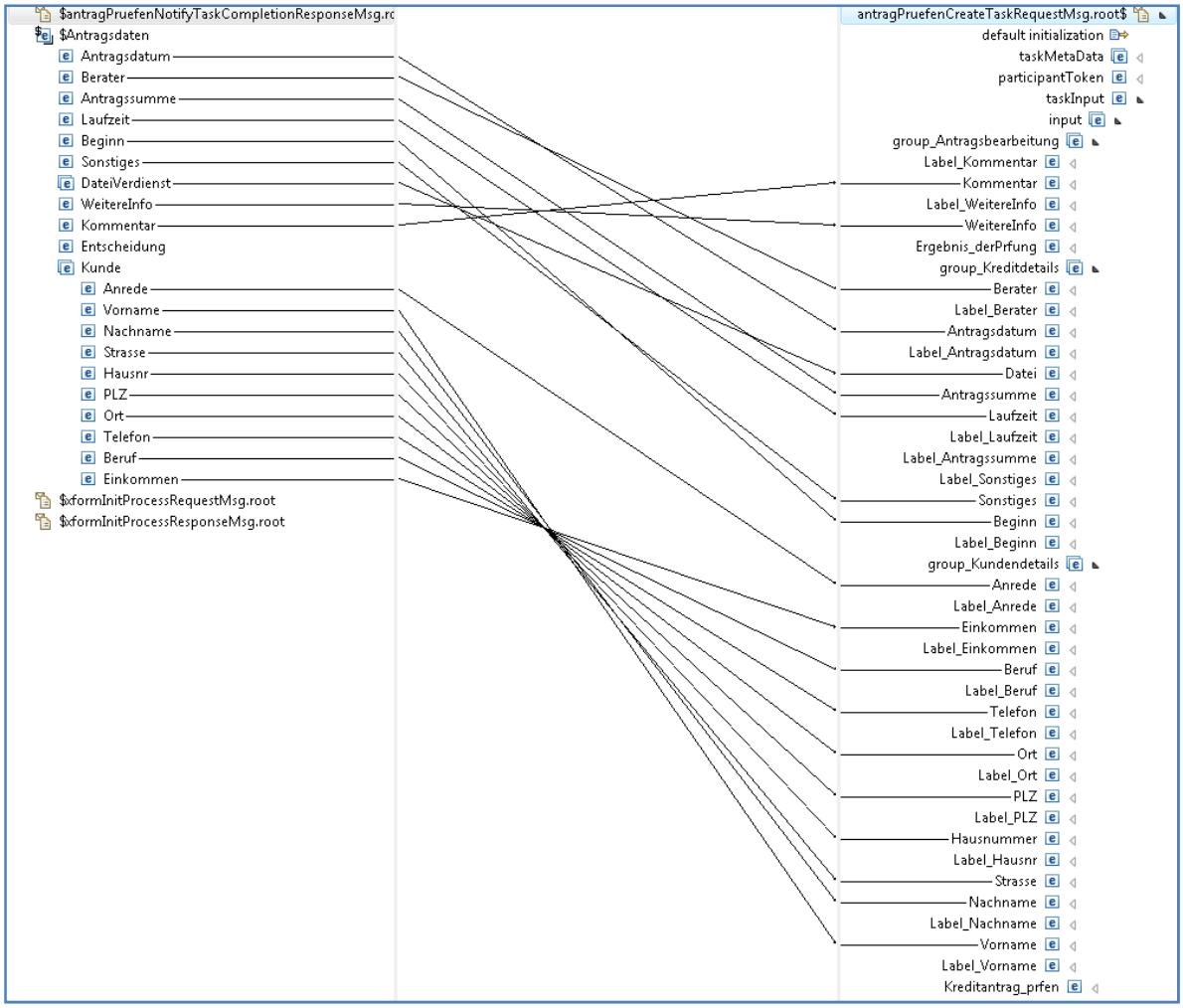

Abbildung 42: *Mapping* der *Input*-Daten für „Antrag prüfen“

Nun müssen noch die Daten der Prozessvariablen an den Dialog übergeben werden. Dies wird in der Aktivität „AntragPruefen-*send*“ mit Hilfe des *Mappers* definiert. Wie Abbildung 42 zeigt, werden im Wesentlichen die gleichnamigen Felder aufeinander abgebildet. Lediglich die Entscheidung wird nicht zugeordnet, da diese bei jedem Schleifendurchlauf neu getroffen werden muss.

Für den ersten Dialog „Antrag erfassen“ wurde in den *Properties* unter „*Workflow*“ eine Bezeichnung eingetragen, die später in der Benutzer-Oberfläche angezeigt wird (Abschnitt 3.7.2, S. 28). Dies wäre für „Antrag prüfen“ ebenfalls möglich. Wenn man jedoch sehr viele Anträge vorliegen hat, ist es wenig hilfreich, fünfzig Mal den Eintrag „Antrag prüfen“ in seiner Aufgabenliste zu sehen. Nützlicher wäre es, darzustellen, um welchen Antrag es sich jeweils handelt. Dies kann ebenfalls mit Hilfe des *Mappings* erreicht werden. Abbildung 43 zeigt, wie ein entsprechender Text mit der Funktion „*concat*()“ zusammengesetzt wird, der das Antragsdatum und den Nachnamen des Kunden enthält. In der Aufgabenliste wird dann später ein Text der Form „Kreditantrag vom 1.2.2008 für Müller prüfen“ angezeigt.

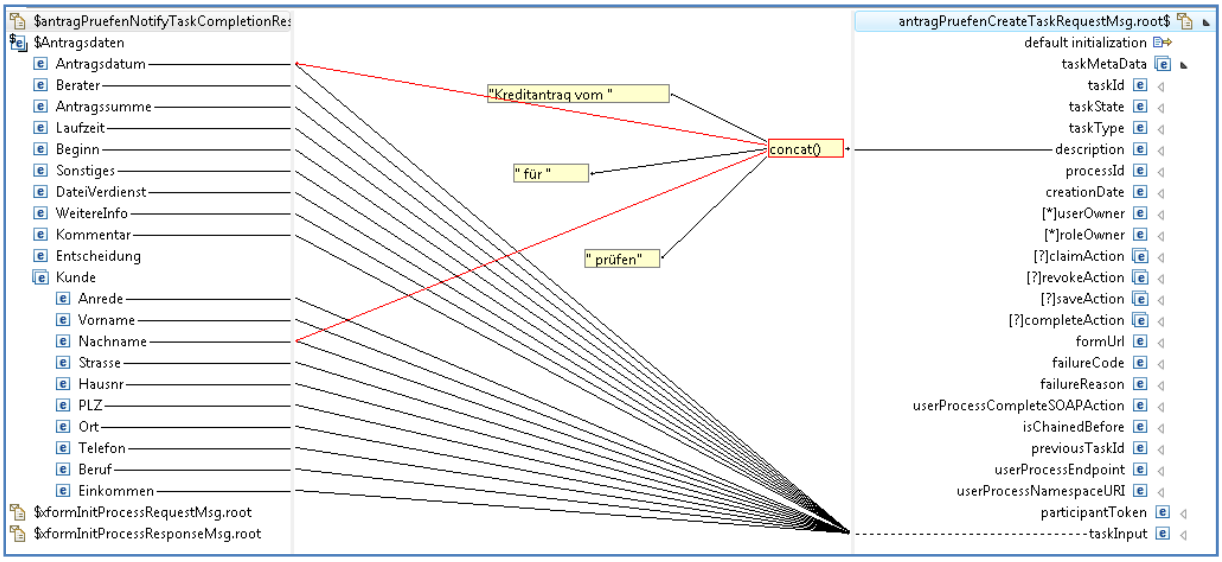

Abbildung 43: Zusammenfügen einer Beschreibung für die Aktivität „Antrag prüfen“

Schließlich müssen noch die neu erfassten oder evtl. geänderten Daten der Prozessvariablen zugeordnet werden. Dies geschieht in einem weiteren Task (Abbildung 44).

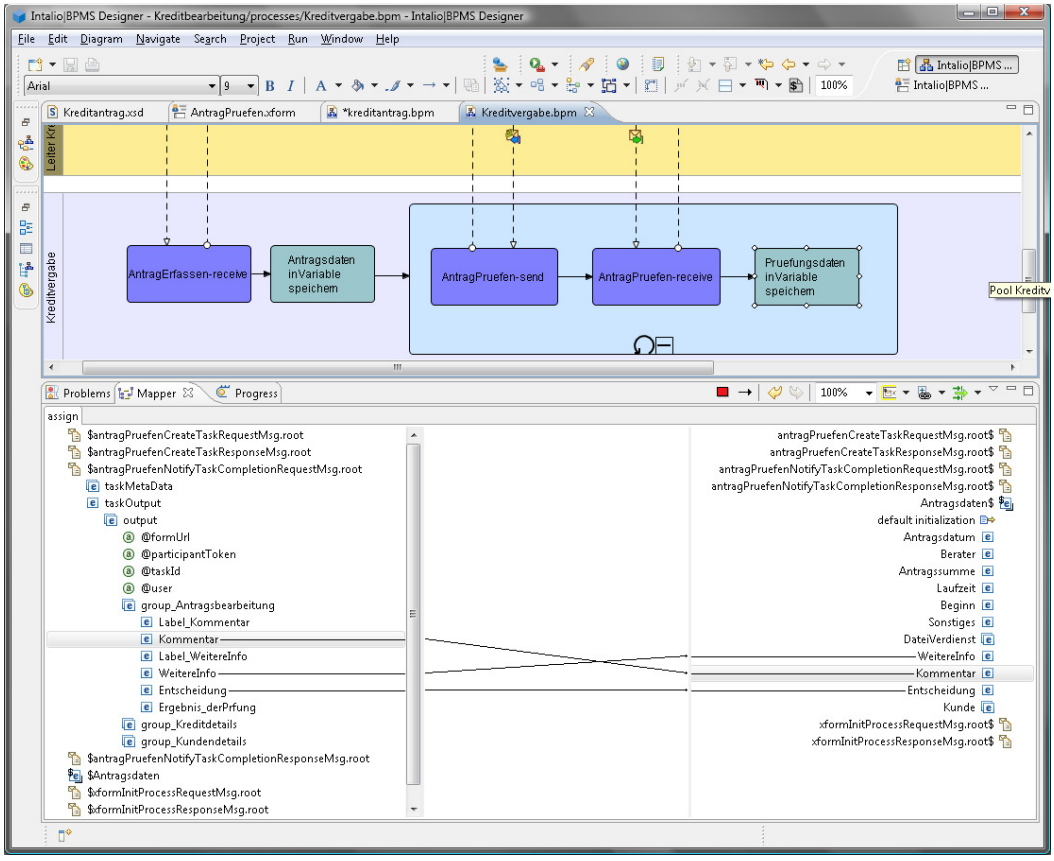

Abbildung 44: Zuordnung der Prüfungsdaten zur Prozessvariablen

3.7.9 Testen der Antragsprüfung

An dieser Stelle bietet es sich an, den *Workflow* erneut zu *deployen* und zu testen. Beim *Deployen* müssen die Dateien des neu erstellten Dialogs mit selektiert werden. Nach dem Erfassen des Antrags

findet sich in der Benutzungs-Oberfläche der Sachbearbeiter unter „*Tasks*“ (Aufgaben) ein Eintrag mit dem zu prüfenden Antrag. Eventuell muss zuvor auf „*Refresh*“ gedrückt werden. Wählt man im Dialog dann den Eintrag „Rückfrage“ aus, so wird die Schleife erneut durchlaufen und man bekommt denselben Antrag sofort wieder zur Prüfung in die *Tasks*-Liste. Wählt man einen anderen Eintrag, so ist der Ablauf beendet.

3.7.10 Dialog Antrag ergänzen

Den Dialog „Antrag ergänzen“ kann man durch Kopieren des Dialogs „Antrag prüfen“ außerhalb des BPM *Designers* erstellen, wie dies in Abschnitt 3.7.7 beschrieben wurde.

Abbildung 45 zeigt den Entwurf des Dialogs. Gegenüber dem Dialog „Antrag prüfen“ hat sich Folgendes geändert:

- Die Überschrift wurde in „Kreditantrag ergänzen“ geändert.
- Die *Radio Buttons* für die Entscheidung entfallen.
- Das Dialogelement für Antworten und Rückfragen wurde nach oben gerückt und der Text des *Labels* geändert
- Der Kommentar wurde entfernt, da er nicht für den Berater bestimmt ist
- Es wurde ein *Upload* eingefügt (Name „Neue Datei“), um eine geänderte Datei hochzuladen.

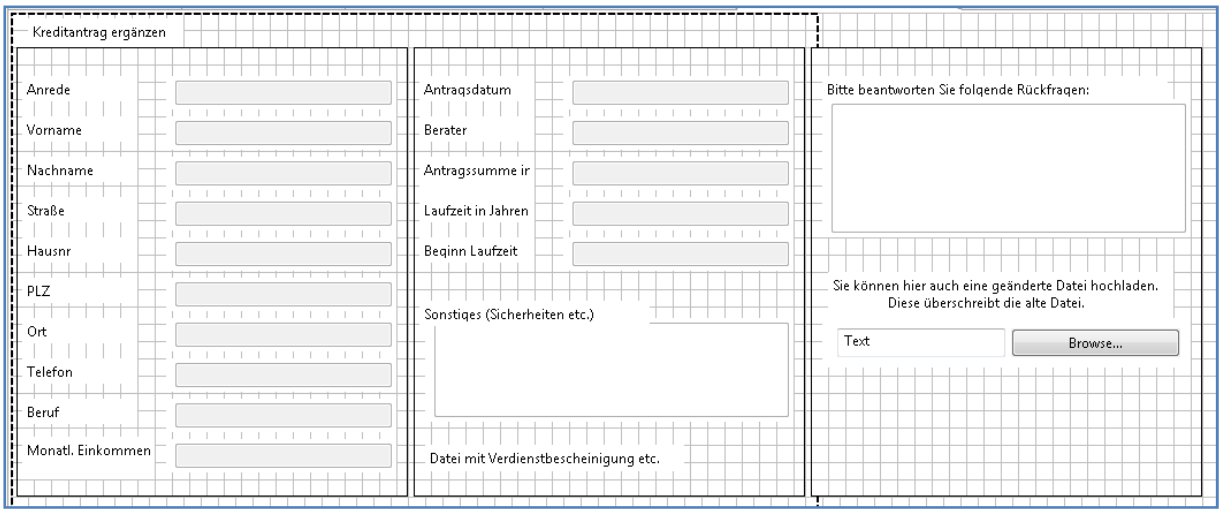

Abbildung 45: Entwurf des Dialogs „Antrag ergänzen“

Aufgrund der Schleife kann dieser Dialog sowie der Dialog „Antrag prüfen“ mehrfach innerhalb eines Prozessdurchlaufs aufgerufen werden. Hierbei tritt die Frage auf, wie mit den in einem vorherigen Schleifendurchlauf eingegebenen Informationen umgegangen werden soll. Soll es beispielsweise möglich sein, die Antragsdaten zu ändern? Wurde ein Fehler bei der Eingabe gemacht, wäre dies wünschenswert. Dies könnte man erreichen, indem man bei den diversen Eingabe-Elementen im obigen Dialog den *Input/Output*-Wert zu „*in-out*“ ändert. Da dies bei der Anrede sowie dem Beginn der Laufzeit wieder andere Eingabe-Elemente mit entsprechenden Typ-Konvertierungen erfordern würde, wurde für das vorliegende Beispiel darauf verzichtet.

Auch die Eingabe von Rückfragen und Antworten muss geregelt werden. Würde man festlegen, dass der Antrag nur einmal zur Ergänzung zurück gegeben werden kann, so könnte man ein Feld für die

Frage und eines für die Antwort verwenden. Kann die Schleife hingegen – wie im vorliegenden Fall – beliebig oft durchlaufen werden, so müssten hierfür bei jedem Schleifendurchlauf ein weiteres Frage- und ein weiteres Antwortfeld hinzugefügt werden. Dies ist einerseits aufwändig zur realisieren, andererseits nicht so einfach im Dialog darzustellen. Daher wird der Einfachheit halber nur ein einziges Textfeld verwendet. Bei jedem Durchlauf der Schleife wird der Text darin von Sachbearbeiter und Berater ergänzt. Nachteil dieser Lösung ist, dass der Inhalt versehentlich gelöscht werden kann.

3.7.11 Integration des Dialogs „Antrag ergänzen“ in den Workflow

Der Dialog wird auf den Berater-*Pool* gezogen und als „*People Activity*“ eingefügt (vgl. Abschnitt 3.7.8). Die beiden im System-*Pool* „Kreditvergabe“ erforderlichen Aktivitäten müssen wiederum in den Schleifen-Unterprozess eingefügt werden. Die Nachrichtenflüsse werden ebenso wie bei „Antrag prüfen“ modelliert.

Wie die EPK in Abbildung 1 zeigt, soll die Ergänzung des Antrags nur durchgeführt werden, wenn eine Rückfrage besteht. Ansonsten soll die Schleife beendet werden. Dies lässt sich in BPMN mit einem *Gateway* modellieren. Es existieren unterschiedliche *Gateways* für exklusive und inklusive Verzweigungen sowie für parallele Abläufe. Im vorliegenden Fall wird eine exklusive Entscheidung benötigt.

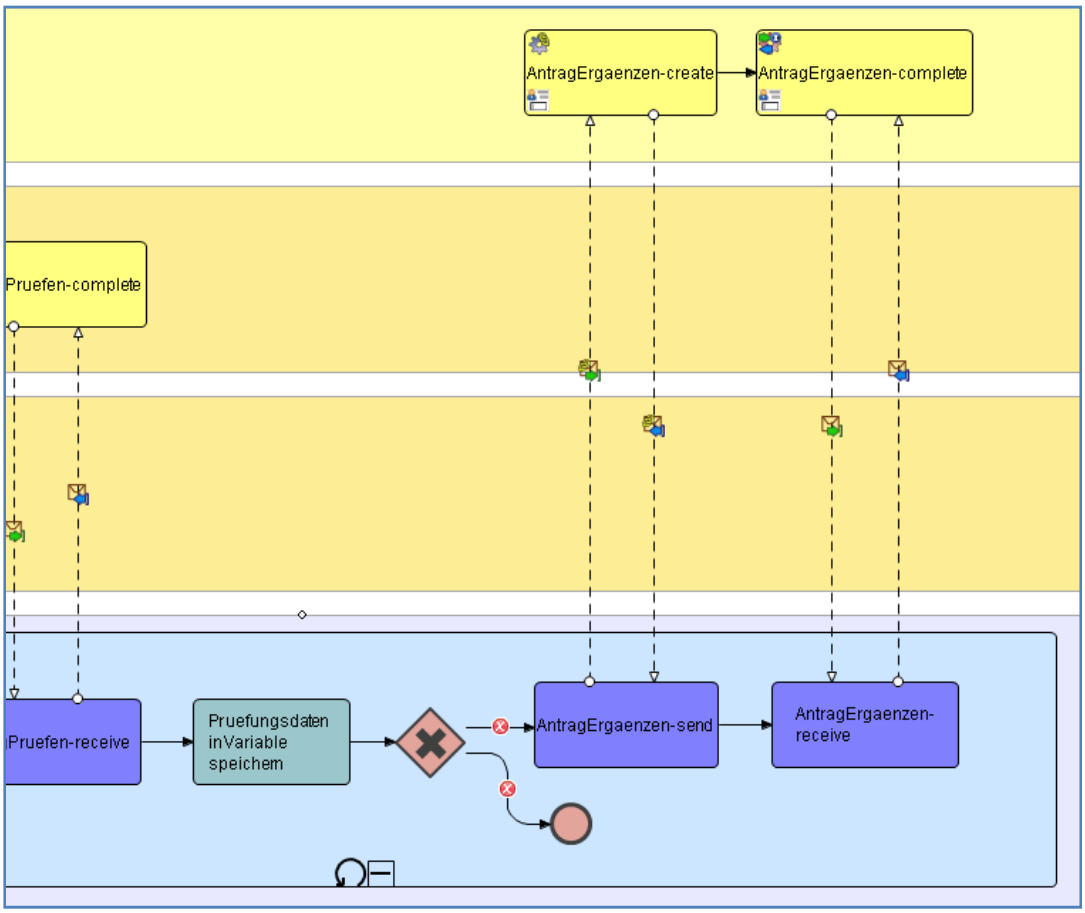

Abbildung 46: Verzweigung und Dialog „Antrag ergänzen“ im *Workflow*

Abbildung 46 zeigt die Verzweigung im Modell. Es sind zwei Ausgangspfade modelliert: Der eine führt zum Ergänzen des Antrags, der andere zu einem End-Ereignis. Wird ein End-Ereignis innerhalb eines Unterprozesses erreicht, so wird der augenblickliche Durchlauf des Unterprozesses beendet, und der

Ablauf wird mit der nächsten Aktivität nach dem Unterprozess fortgesetzt (oder es folgt der nächste Schleifendurchlauf, wenn die Abbruchbedingung noch nicht eingetreten ist).

An den Verbindungen, die aus dem *Gateway* herausführen sind zwei Fehlersymbole angezeigt. Fährt man mit dem Mauszeige auf ein solches Symbol, so wird der Grund des Fehlers angezeigt, in diesem Fall „*A case must define a condition*“ (Für einen Fall muss eine Bedingung formuliert sein). Diese Bedingungen müssen noch formuliert werden.

Der Antrag soll ergänzt werden, wenn das Feld „Entscheidung“ den Wert „rueckfrage“ enthält. Selektiert man den *Gateway*, so kann man für jede ausgehende Verbindung die hierfür erforderliche Bedingung formulieren. Die Bedingung ist genau die umgekehrte Bedingung, wie sie für den Abbruch der Schleife formuliert wurde (S.41), es braucht nur die Funktion „*not*()“ weg gelassen zu werden.

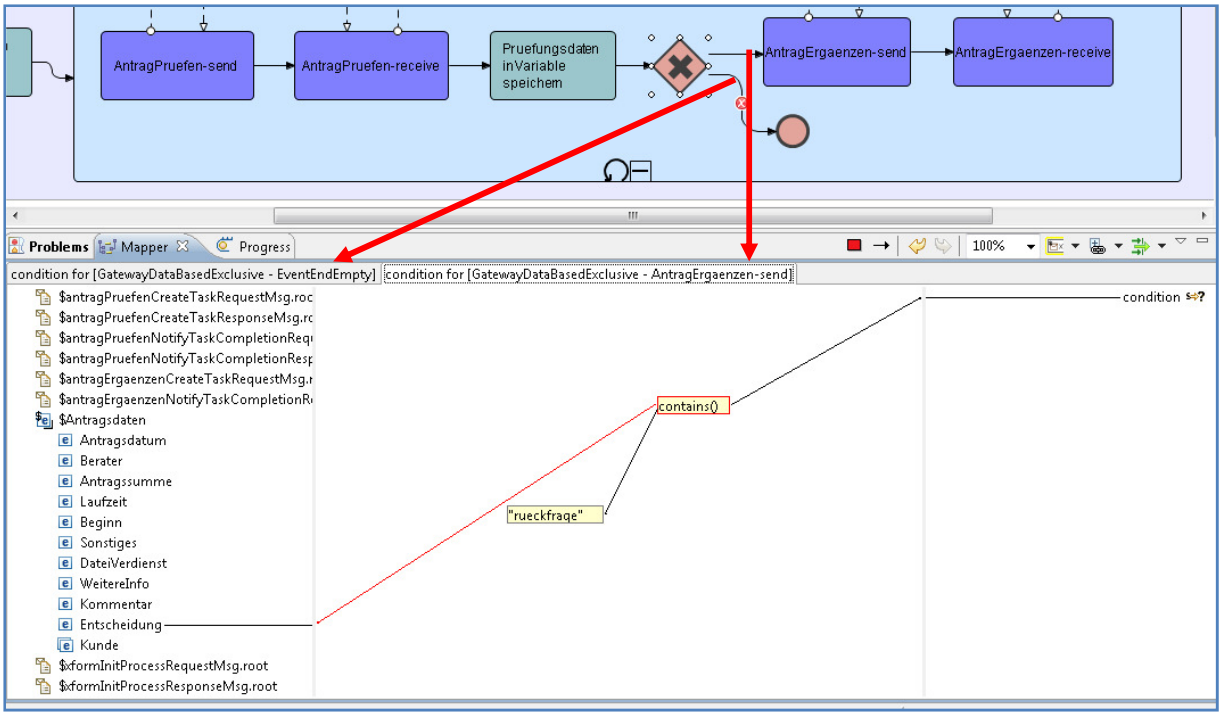

Abbildung 47: Exklusives *Gateway* – Bedingungen (hier für den oberen Pfad dargestellt)

Für den Pfad zum End-Ereignis könnte man nun ebenfalls eine Bedingung formulieren. Dies wäre dieselbe wie für den Schleifenabbruch. Stattdessen kann man diesen Pfad auch als Standardpfad setzen, indem man über das Kontextmenü der rechten Maustaste „*Set as default choice*“ auswählt. Dieser Pfad wird nun immer dann durchlaufen, wenn keine Bedingung eines anderen aus dem *Gateway* herausgehenden Pfades zutrifft. Der *Default*-Pfad wird mit einem kleinen Schrägstrich markiert (Abbildung 48).

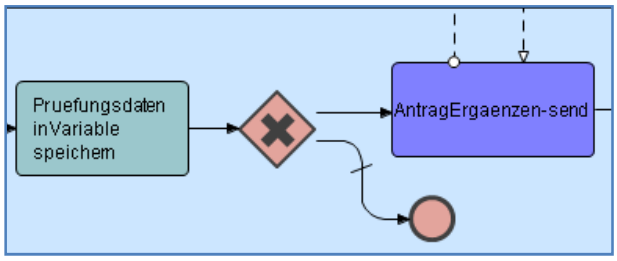

Abbildung 48: Kennzeichnung des *Default*-Pfades mit einem kleinen Schrägstrich

Im *Mapping* der Aktivität „AntragErgaenzen-*send*“ sind wiederum die Antragsdaten dem Dialog als *Input* zuzuordnen (Abbildung 49).

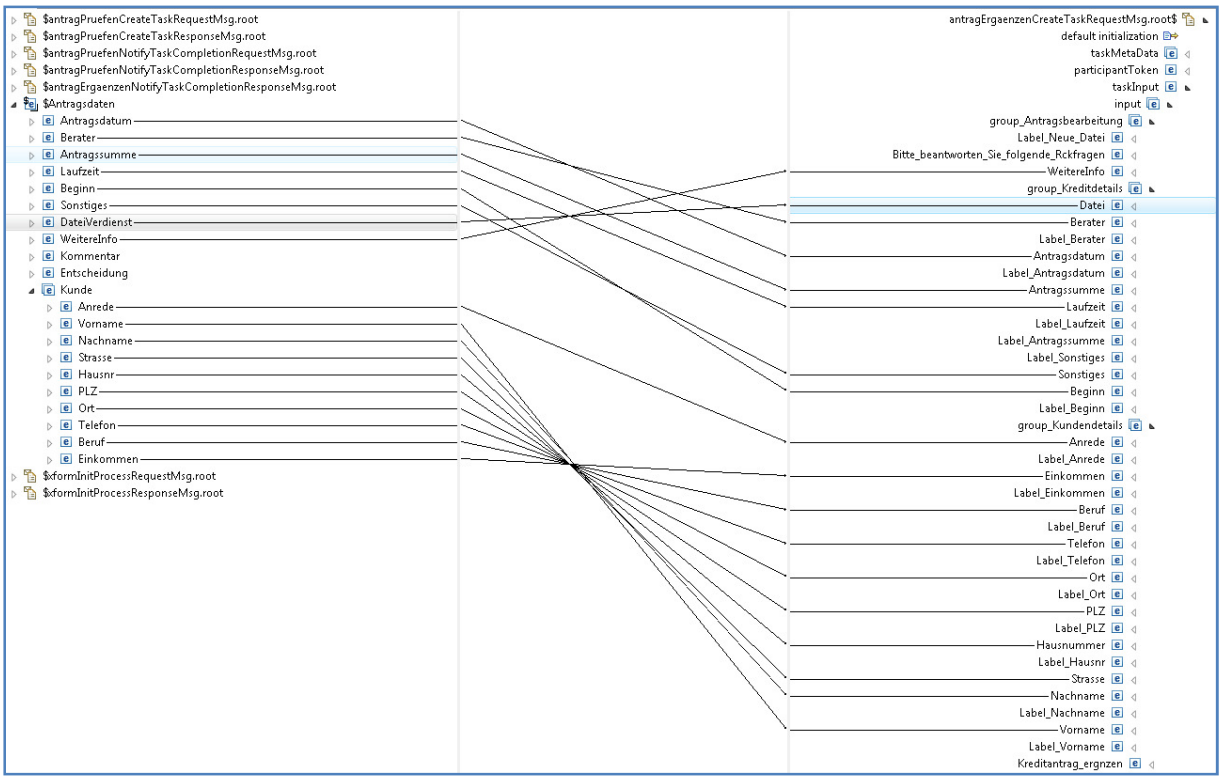

Abbildung 49: *Mapping* der Aktivität „AntragErgaenzen-*send*“

3.7.12 Dynamische Zuordnung des Benutzers

Dadurch, dass der Dialog in den *Pool* „Berater“ gelegt wurde, wird erreicht, dass die Aufgabe „Antrag ergänzen“ nur bei Benutzern mit der Rolle Berater in der *Task*-Liste erscheint (Abbildung 50). Allerdings erscheint sie bei *allen* Beratern. Dies ist nicht gewünscht, richtet sich die Rückfrage doch nur an einen Berater, nämlich den, der vorher den betreffenden Antrag erfasst hat.

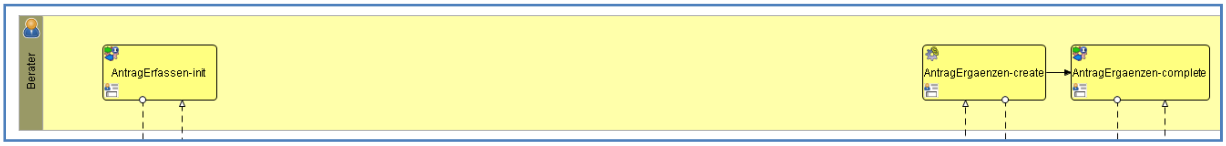

Abbildung 50: *Pool* „Berater“

Dies lässt sich auf die folgende Weise erreichen. Zunächst wird die Zuordnung der Rolle Berater zum Pool gelöscht. Anschließend wird die Rolle Berater der Aktivität „AntragErfassen-*init*“ direkt zugeordnet. Diese soll ja bei allen Beratern gleichermaßen erscheinen.

Die Rollenzuordnung sowohl von *Pools* als auch von Aktivitäten findet sich in den *Properties* unter *Workflow* im Feld „*Role(s)*“. Beim *Pool* ist der Eintrag dieses Feldes zu löschen, bei der Aktivität ist „kreditbank/berater“ hinzuzufügen.

Die Zuordnung der Rolle zur Ergänzung des Antrags erfolgt ebenfalls im *Mapping* der Aktivität „AntragErgaenzen-*send*“. Hier wird das Feld „Berater“ aus den Antragsdaten dem Feld „*userOwner*“

(der Benutzer, der die Aktivität „besitzt“) zugeordnet (Abbildung 51). Das Feld „Berater“ enthält ja den Benutzernamen des Beraters, der den Prozess angestoßen hat.

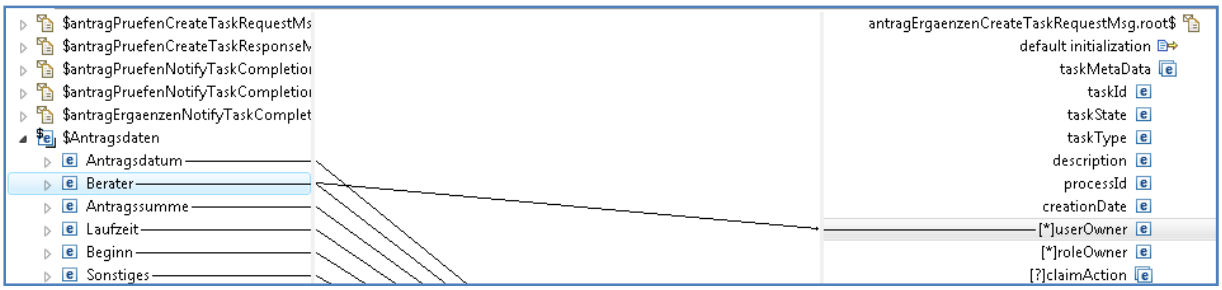

Abbildung 51: Zuordnung des Besitzers zu „Antrag ergänzen“

An den Tasks „AntragErgaenzen-*create*“ und „AntragErgaenzen-*complete*“ erscheint nun ein kleines Warnsymbol. Geht man mit dem Mauszeiger darauf, so erhält man den Hinweis „*The workflow task ist not associated with a user or a role.*“ (Die *Workflow*-Aktivität ist weder einem Benutzer noch einer Rolle zugeordnet). Da die Rollenzuordnung aber über das *Mapping* definiert wurde, stellt dies kein Problem dar, und die Warnung kann ignoriert werden.

Im Gegensatz zu „Antrag ergänzen“ ist es für die Funktion „Antrag prüfen“ durchaus gewollt, dass diese in den *Task*-Listen aller Sachbearbeiter erscheint. Sie wird dann jeweils von dem nächsten verfügbaren Sachbearbeiter ausgeführt.

Schließlich wird wieder eine aussagekräftige Beschreibung für den Eintrag der Aktivität in die *Task*-Liste definiert (Abbildung 52).

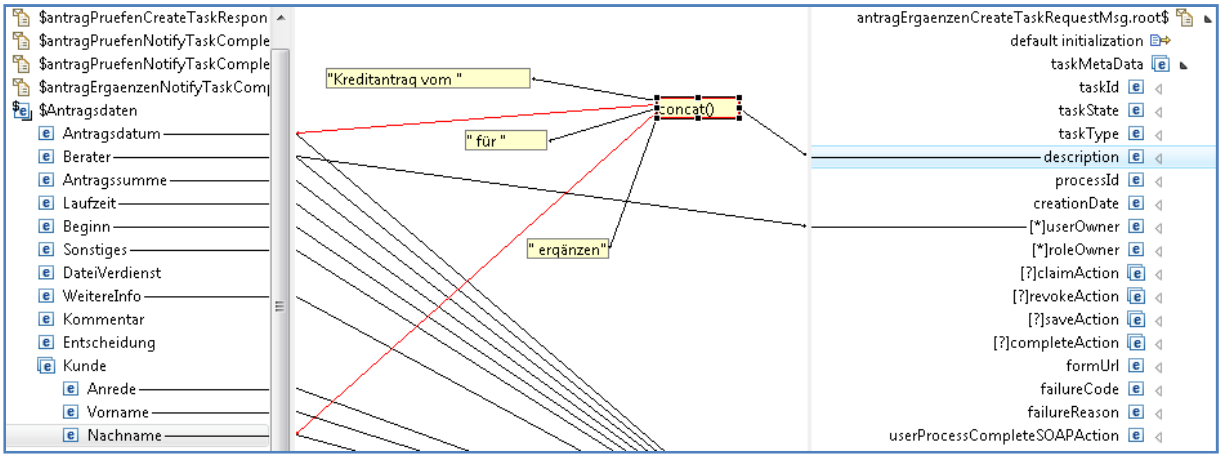

Abbildung 52: Modellierung der Beschreibung für die Aktivität „Antrag ergänzen“

3.7.13 Zuordnung der ergänzenden Daten zur Prozessvariablen

Beim Ergänzen eines Antrags können nur zwei Datenelemente geändert werden: Das Feld „WeitereInfo“, das die Rückfragen und Antworten enthält, und die hochgeladene Datei. Um nicht mit vielen Dateien umgehen zu müssen, soll die neue Datei die alte überschreiben. Wird allerdings keine neue Datei hochgeladen, soll die alte Datei erhalten bleiben. Würde man im *Mapping* einfach das Feld „NeueDatei“ dem Feld „DateiVerdienst“ in den Antragsdaten zuordnen, würde die alte Datei gelöscht, wenn keine neue Datei hochgeladen wurde. Daher muss zunächst geprüft werden, ob

tatsächlich eine neue Datei hochgeladen wurde. Nur in diesem Fall soll eine Zuordnung zum Feld „DateiVerdienst“ erfolgen.

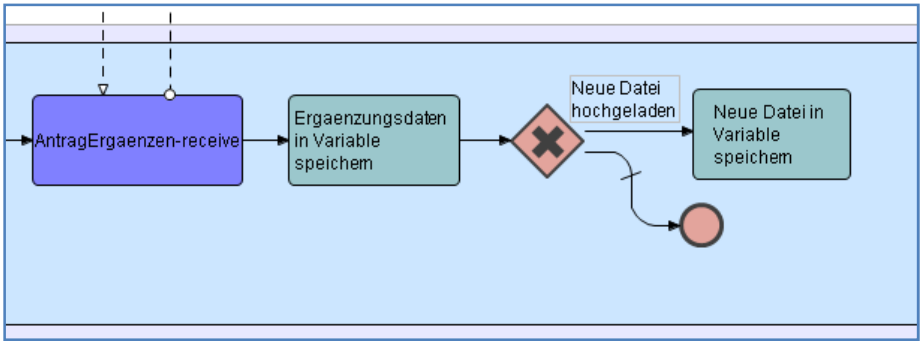

Abbildung 53: Zuordnung der ergänzenden Daten zur Prozessvariable

Dies ist in Abbildung 53 modelliert. Im *Mapping* der Aktivität „Ergänzungsdaten in Variable speichern“ findet lediglich die Zuordnung von „WeitereInfo“ statt (Abbildung 54).

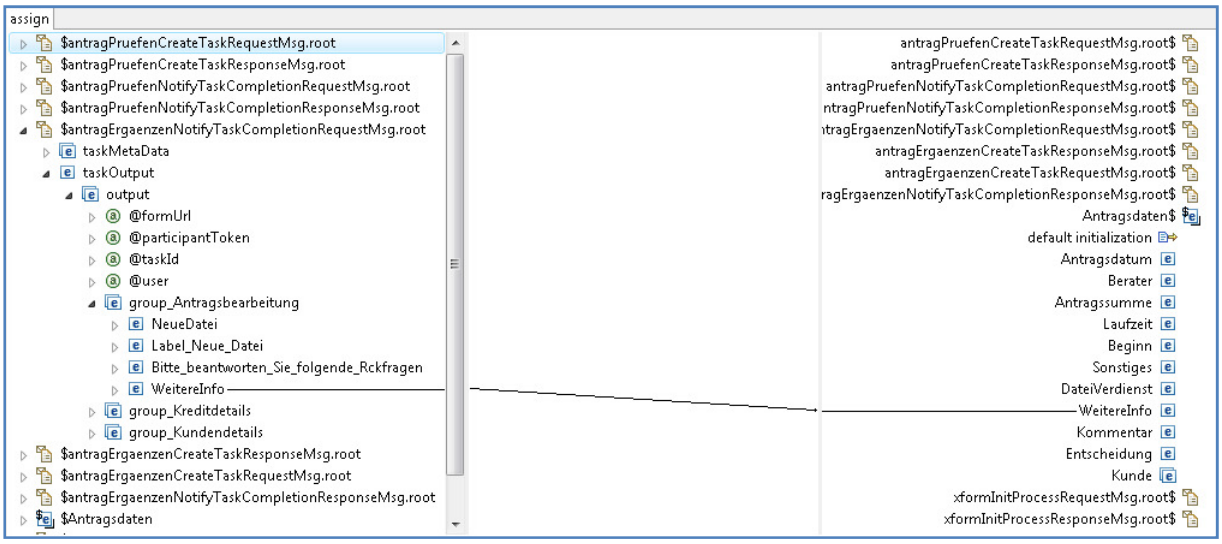

Abbildung 54: *Mapping* der Aktivität „Ergänzungsdaten in Variable speichern“

Um festzustellen, ob eine neue Datei hochgeladen wurde, wird für den oberen aus dem *Gateway* herausführenden Pfad überprüft, ob der Dateiname länger als null Zeichen ist (Abbildung 55). In diesem Fall wird die Aktivität „Neue Datei in Variable speichern“ ausgeführt. Ihr Mapping wird in Abbildung 56 gezeigt. Zur besseren Lesbarkeit wird der Kante im Diagramm die Beschriftung „Neue Datei hochgeladen“ hinzugefügt, die aber keine Auswirkung auf den Kontrollfluss hat (Abbildung 53).

An dieser Stelle empfiehlt es sich, den Prozess erneut zu *deployen* und zu testen. Insbesondere sollte getestet werden, ob die Zuordnung des Beraters, das Eintragen von Fragen und Antworten und das Hochladen einer neuen Datei richtig klappen.

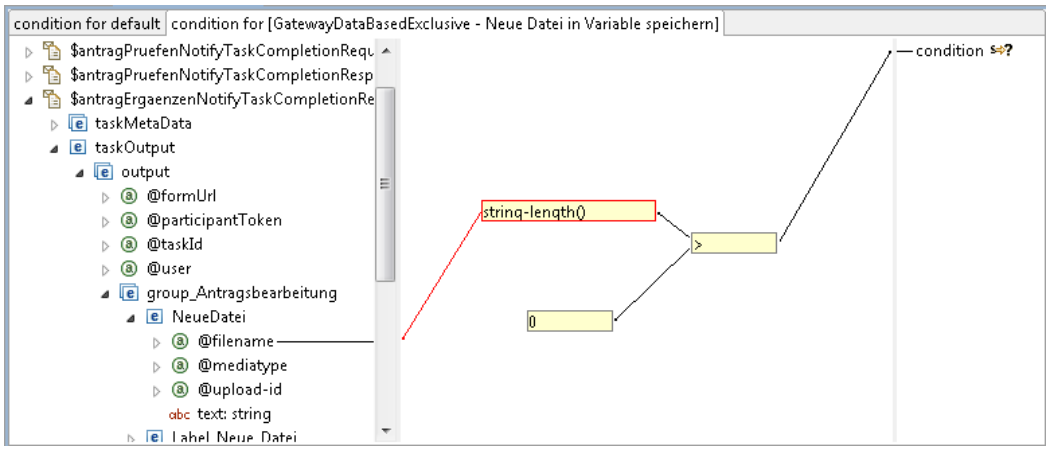

Abbildung 55: Überprüfung, ob eine neue Datei hochgeladen wurde

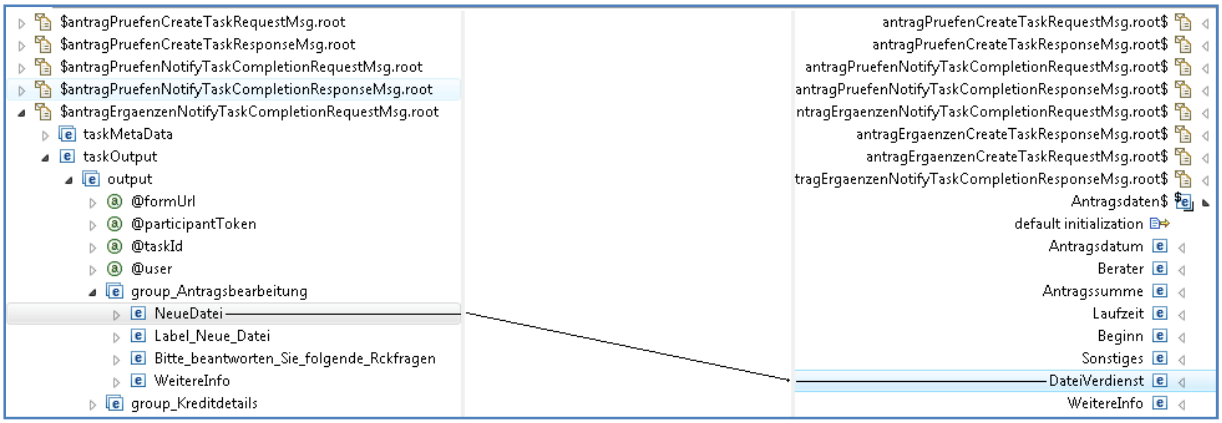

Abbildung 56: Zuordnung der neu hochgeladenen Datei zur Prozessvariablen

3.7.14 Dialog „Antrag entscheiden“

Dieser Dialog wird wiederum durch Kopieren des Dialogs „Antrag ergänzen“ erzeugt (außerhalb des BPM *Designers*), wie dies in Abschnitt 3.7.7 beschrieben wurde.

Abbildung 57 zeigt den Entwurf des Dialogs. Gegenüber dem Dialog „Antrag ergänzen“ wurden folgende Änderungen vorgenommen:

- Die Überschrift wurde geändert zu „Über Kreditantrag entscheiden“
- Das Feld „Weitere Info“ bekam den *Input/Output*-Wert „*in*“, der Wert für „*Read-only*“ ist nun „*true()*“. Das *Label* wurde zu „Ergänzungen“ geändert.
- Es wurde ein Feld mit dem Kommentar des Sachbearbeiters eingefügt (ebenfalls *in* und *Read-only*).
- Das *Upload*-Element zum Hochladen einer Datei wurde entfernt.
- Es wurde ein *Radio Button*-Element „Entscheidung“ eingefügt. Es liefert als mögliche Werte „genehmigt“ oder „abgelehnt“.

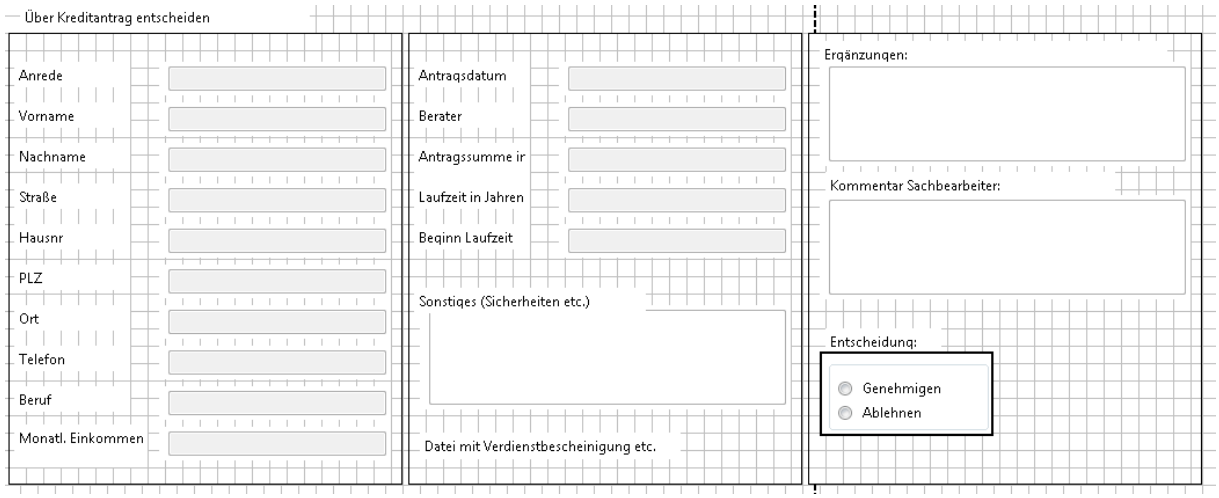

Abbildung 57: Entwurf des Dialogs „Antrag entscheiden“

Der Dialog wird wie bekannt durch Ziehen der Datei „AntragEntscheiden.xform“ aus dem *Process Explorer* auf den *Pool* „Leiter Kreditbearbeitung“ und Anlegen von zwei Aktivitäten im System-*Pool* „Kreditvergabe“ mit den entsprechenden Nachrichtenflüssen in den *Workflow* integriert.

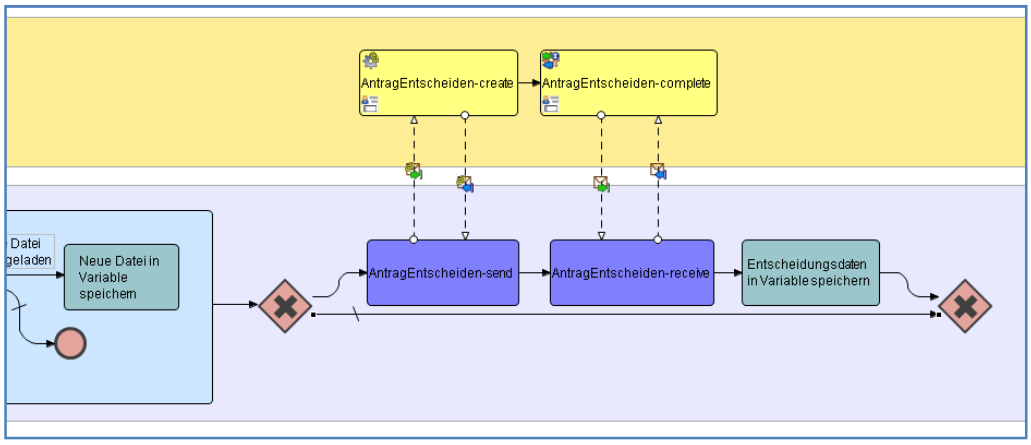

Abbildung 58: Integration des Dialogs „Antrag entscheiden“ in den *Workflow*

Allerdings wird dieser Dialog nicht in jedem Fall aufgerufen, weshalb in Abbildung 58 ein *Gateway* mit einer exklusiven Verzweigung modelliert wurde. Im unteren *Default*-Pfad wird der Dialog nicht aufgerufen.

Für den oberen Pfad muss eine Bedingung formuliert werden. Diese lässt sich aus der EPK nach Abbildung 1 ableiten. Die Entscheidung des Leiters der Kreditbearbeitung ist demnach notwendig, wenn:

1. die Antragssumme größer oder gleich 5000 ist und der Antrag vom Sachbearbeiter genehmigt wurde, oder
2. der Antrag vom Sachbearbeiter zur Entscheidung weitergeleitet wurde.

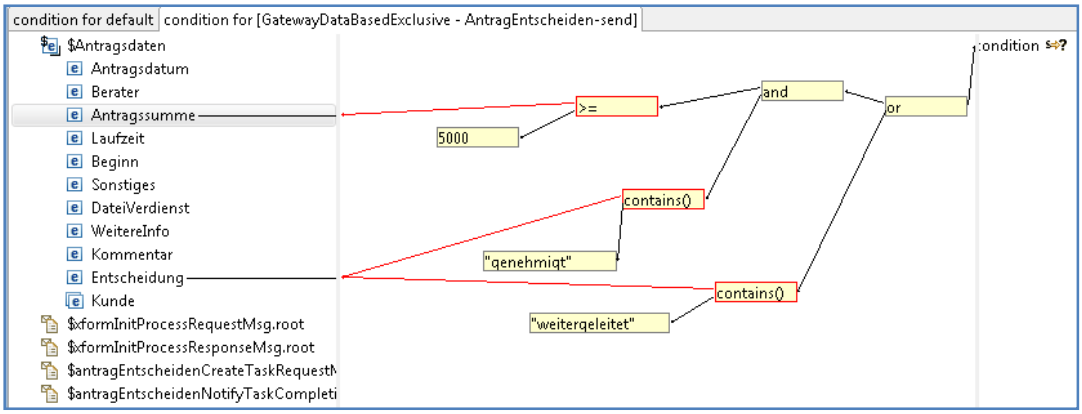

Abbildung 59: Formulierung der Bedingung, unter der eine Entscheidung des Leiters der Kreditbearbeitung erforderlich ist.

Abbildung 59 zeigt, wie diese Bedingung im *Mapper* modelliert wird. Schließlich müssen wie üblich die *Input*-Daten für den Dialog definiert werden (Abbildung 60).

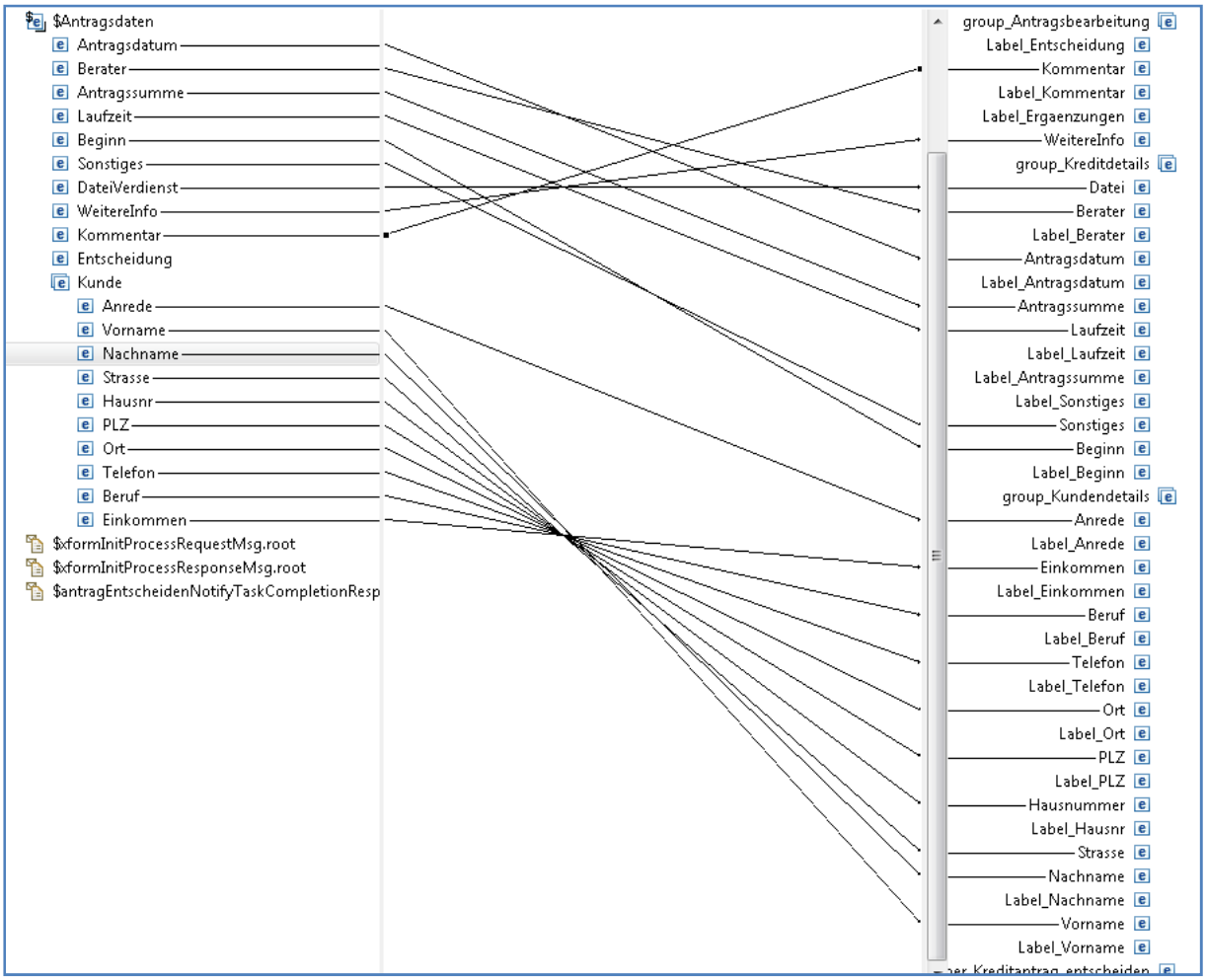

Abbildung 60: *Mapping* der Aktivität „AntragEntscheiden-*send*“

Ergänzend kommt wieder die Definition einer aussagekräftigen Beschreibung hinzu (Abbildung 61).

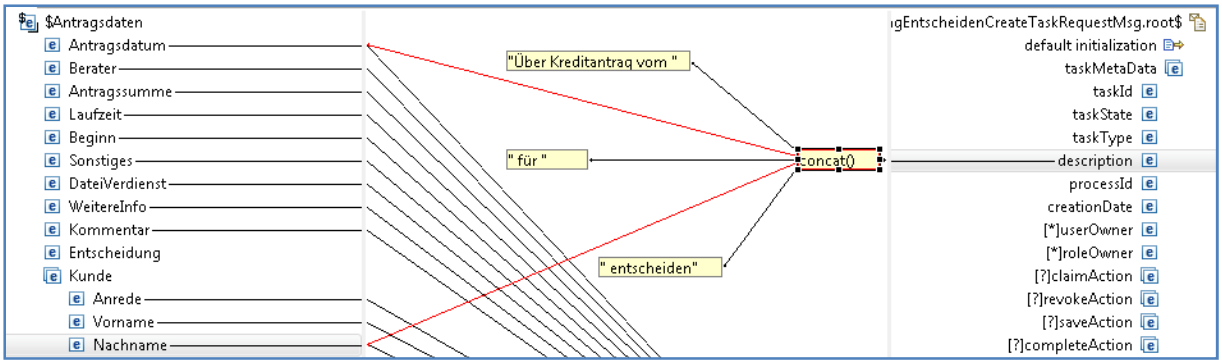

Abbildung 61: Beschreibung für die Aktivität „Antrag entscheiden“

Schließlich muss die getroffene Entscheidung auch wieder der Prozessvariablen zugeordnet werden. Dies erfolgt im *Mapping* der Aktivität „Entscheidungsdaten in Variable speichern“ (Abbildung 62).

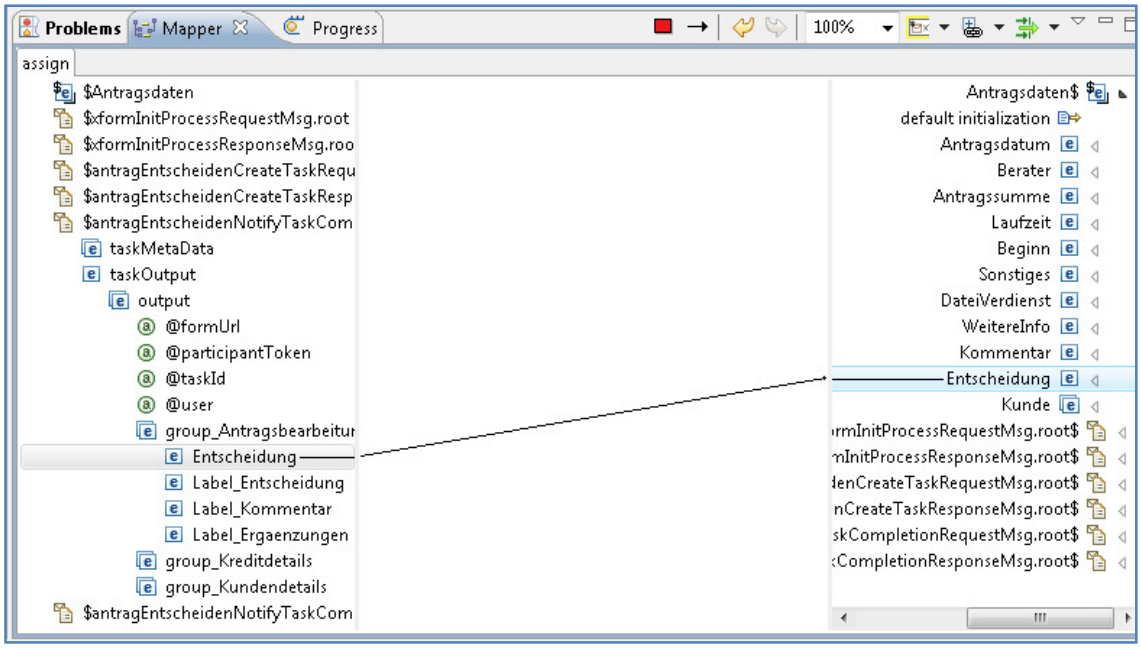

Abbildung 62: Zuordnung der Entscheidung zur Prozessvariable

3.7.15 Über Ergebnis der Antragsprüfung informieren

Der Dialog „Entscheidung mitteilen“ wird ebenfalls durch Kopieren eines vorhandenen Dialogs, „Antrag entscheiden“ (außerhalb des BPM *Designers*) erstellt. Es wurden folgende Änderungen durchgeführt:

- Die Überschrift wurde geändert in „Mitteilung über Kreditantrag“
- Die *Groups* mit Kunden- und Antragsdaten wurden nach rechts verschoben.
- Es wurde eine Reihe von Feldern mit Kommentaren, Berater, zusätzlichen Informationen, der Entscheidung und der hochgeladenen Datei gelöscht.
- Links wurde eine *Group* angelegt. Darin befinden sich zwei *Label*: Das eine, benannt als „Mitteilungstext“ dient zur Aufnahme des Textes, mit dem die Annahme oder Ablehnung des Antrags mitgeteilt wird. Der Wert für *Input/Output* ist dementsprechend „*in*“. Das zweite Label enthält die Unterschrift.

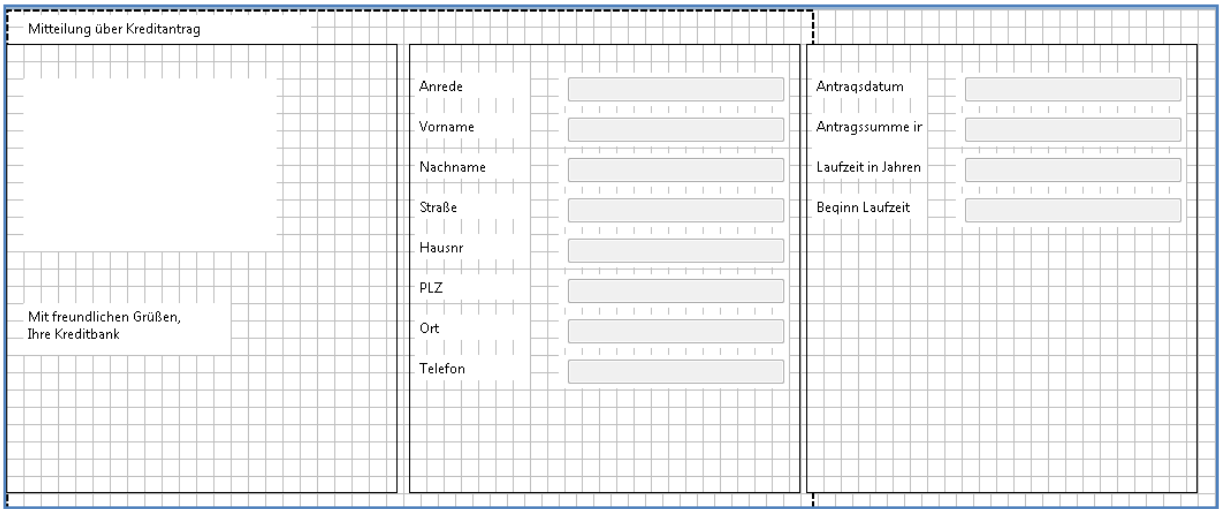

Abbildung 63: Dialog Entscheidung mitteilen

Der Dialog wird in den *Pool* „Berater“ gezogen. Im Gegensatz zu den anderen Dialogen sollen diesmal keine Benutzereingaben im Prozess verarbeitet werden, er dient nur zur Benachrichtigung des Beraters. Beim Einfügen in den *Pool* ist daher „*Use* EntscheidungMitteilen *for Notification (notify)*“ auszuwählen. Daraufhin wird nur eine Aktivität angelegt, entsprechen muss auch nur eine Aktivität im System-*Pool* „Kreditbearbeitung“ angelegt und durch Nachrichtenflüsse verbunden werden, wobei der erste Nachrichtenfluss vom System-*Pool* zum Dialog geht (von unten nach oben), der zweite in Gegenrichtung.

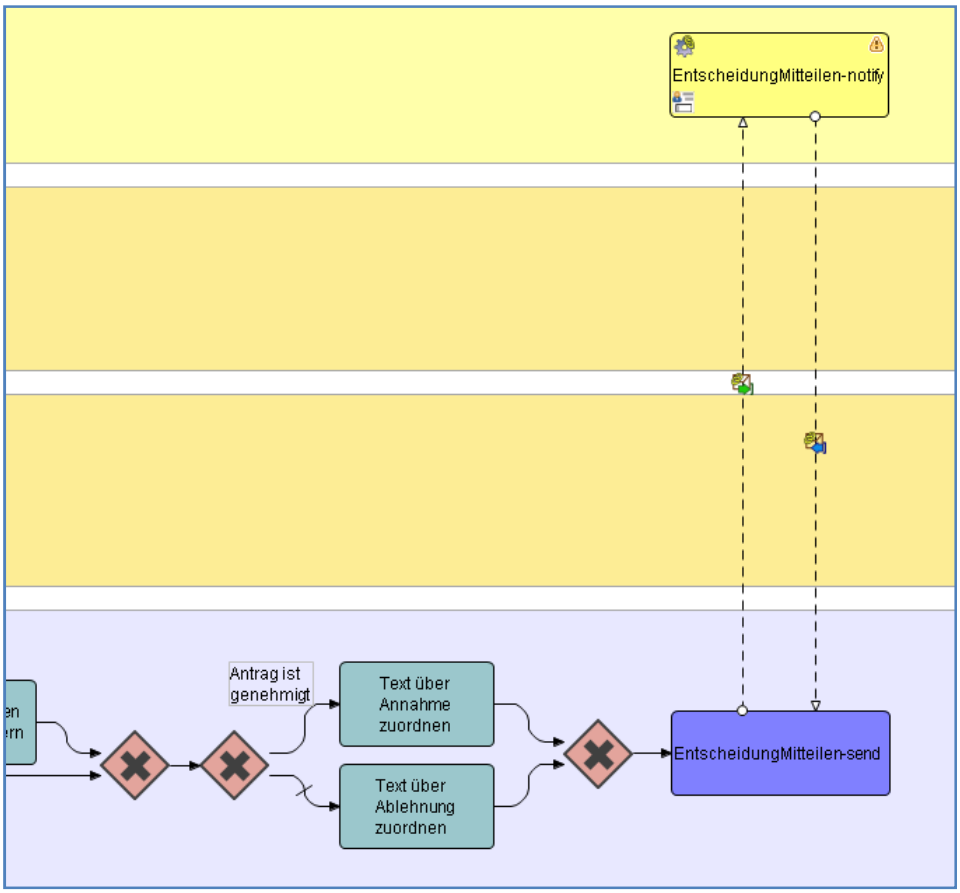

Abbildung 64: Einbindung des Dialogs „Entscheidung mitteilen“ in den *Workflow*

Da in der Mitteilung je nach Entscheidung ein anderer Text stehen soll, ist in Abbildung 64 eine Verzweigung modelliert. Je nach Pfad wird ein Text zugeordnet, der die Annahme oder die Ablehnung des Antrags mitteilt. Die Annahme wird nur mitgeteilt, wenn der Wert des Feldes „Entscheidung“ „genehmigt“ ist. Dies ist als Bedingung für die obere Kante definiert (Abbildung 65). Ansonsten wird dem unteren *Default*-Pfad gefolgt, und es wird der Mitteilung ein ablehnender Text zugeordnet.

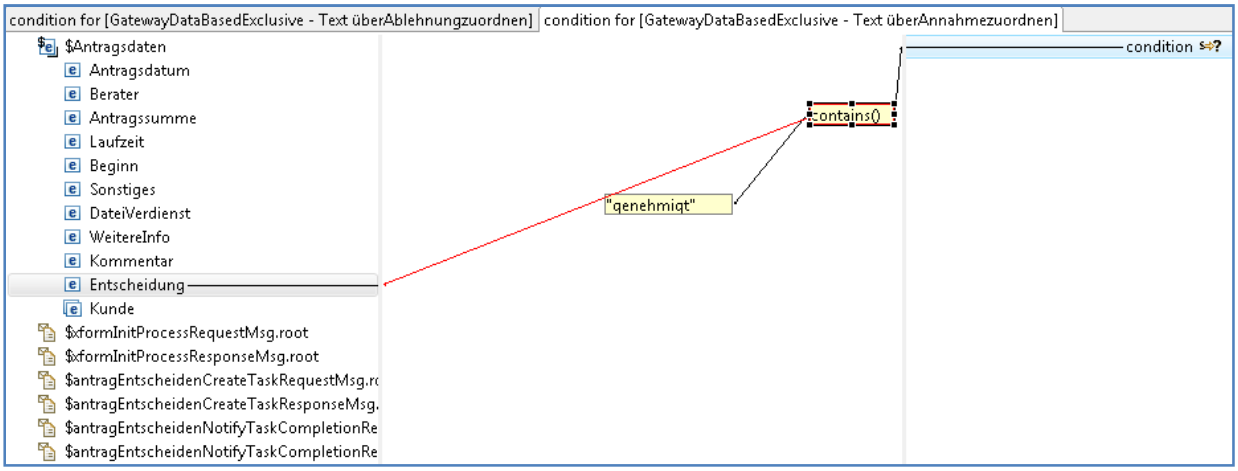

Abbildung 65: Bedingung für die Aktivität „Text über Annahme zuordnen“

Abbildung 66 zeigt das *Mapping* der Aktivität „Text über Annahme zuordnen“. Der Text wird direkt dem *Input*-Datenfeld „Mitteilungstext“ des Dialogs „EntscheidungMitteilen“ zugewiesen (d. h. er wird nicht erst in einer Variablen zwischengespeichert).

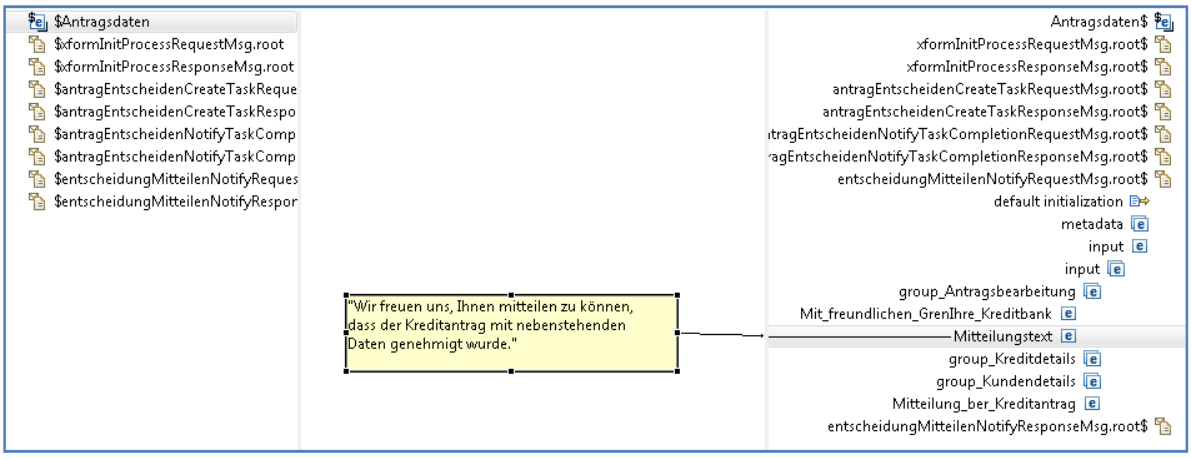

Abbildung 66: *Mapping* der Aktivität „Text über Annahme zuordnen“

Das in Abbildung 67 gezeigte *Mapping* der Aktivität „Text über Ablehnung zuordnen“ unterscheidet sich lediglich dadurch, dass eben ein anderer Text zugeordnet wird.

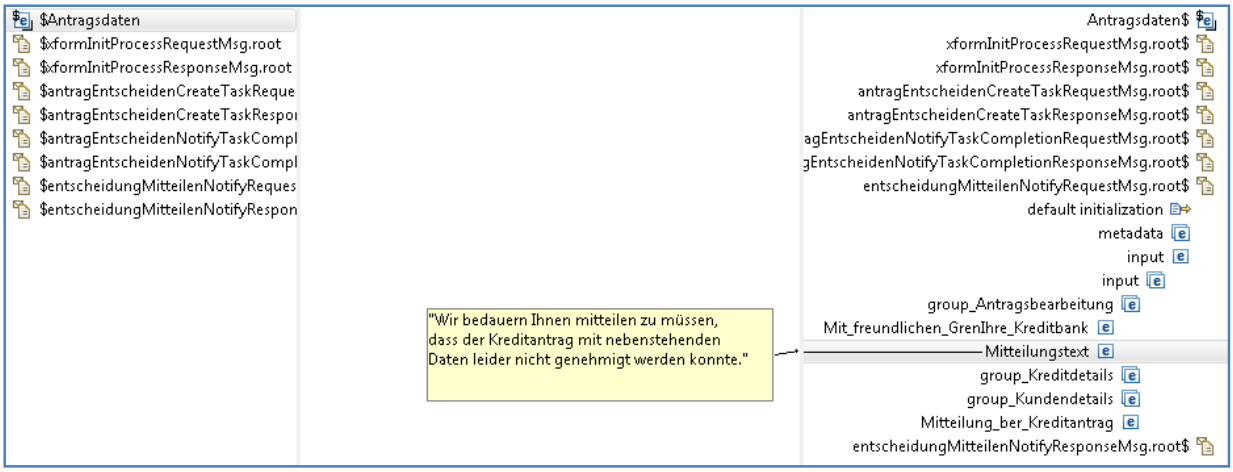

Abbildung 67: *Mapping* der Aktivität „Text über Ablehnung zuordnen“

Schließlich ist wieder die Zuordnung der benötigten Antragsdaten-Werte zu den Dialogfeldern zu treffen (Abbildung 68). Auch die Mitteilung soll wieder nur dem Berater zugestellt werden, der den Antrag erfasst hat. Daher wird – wie bereits bei der Aktivität „Antrag ergänzen“ – der Wert von „Berater“ dem Feld „*userOwner*“ zugeordnet.

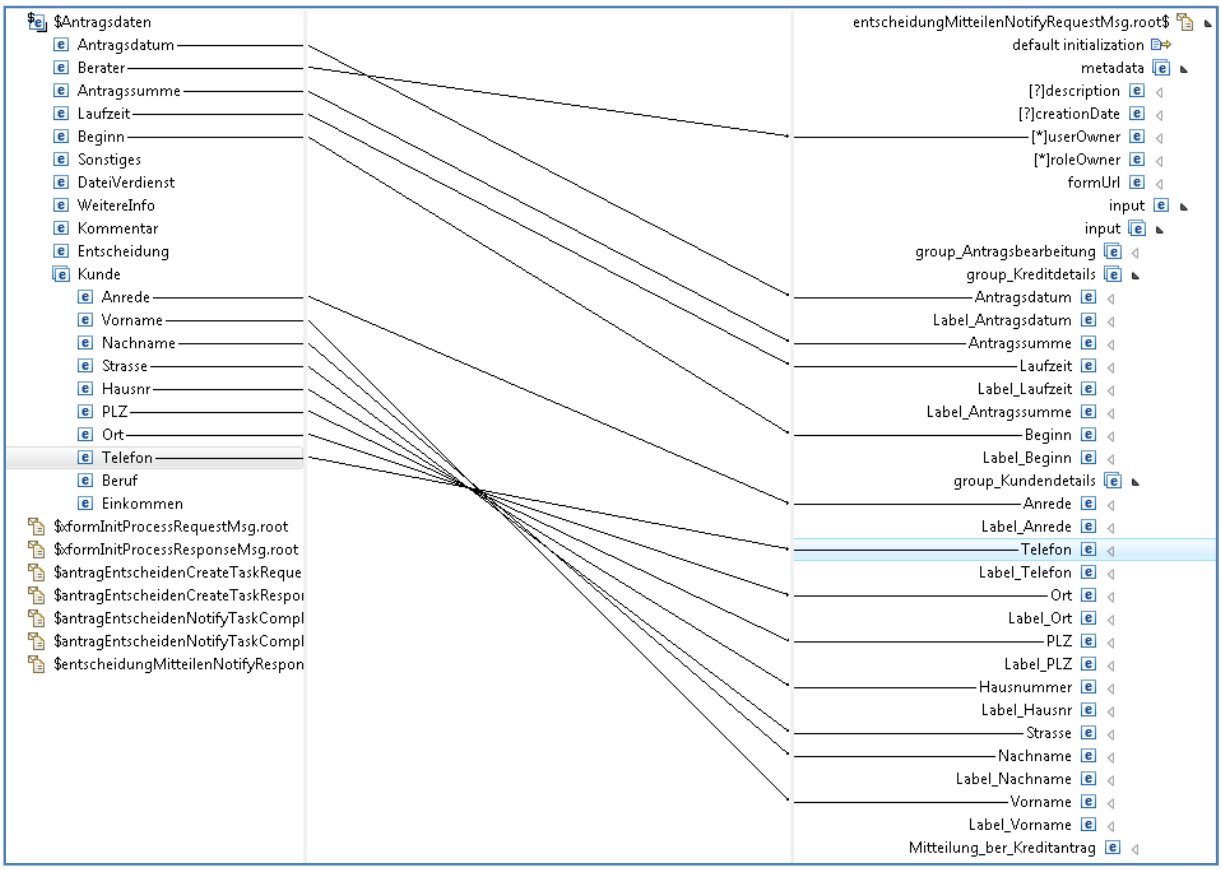

Abbildung 68: *Mapping* der Aktivität „Entscheidung mitteilen-*send*“

Und auch in dieses Mapping wird noch die Definition einer aussagekräftigen Beschreibung eingefügt (Abbildung 69).

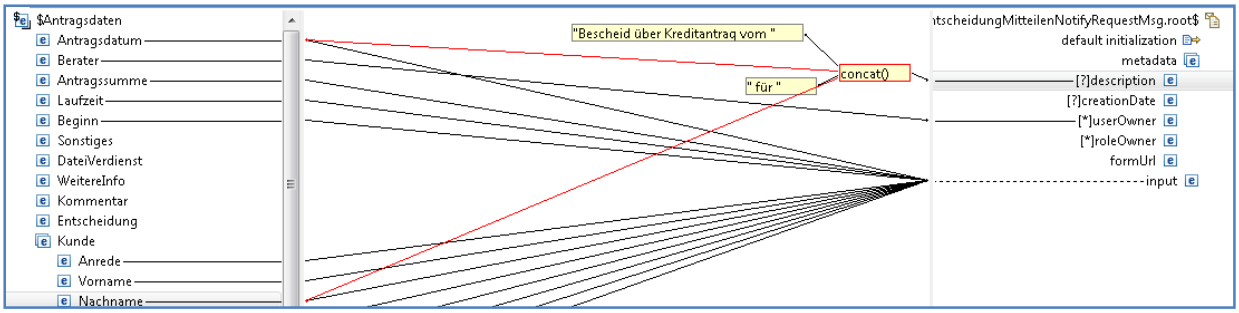

Abbildung 69: Beschreibung der Mitteilung

Damit sind die Modellierung des *Workflows*, die Erstellung der Dialoge und der Daten-*Mappings* abgeschlossen. Der fertige Prozess muss nun wieder *deployed* und getestet werden. Im folgenden Abschnitt wird das komplette *Workflow*-Modell noch einmal im Zusammenhang dargestellt.

3.7.16 Das komplette Workflow-Modell

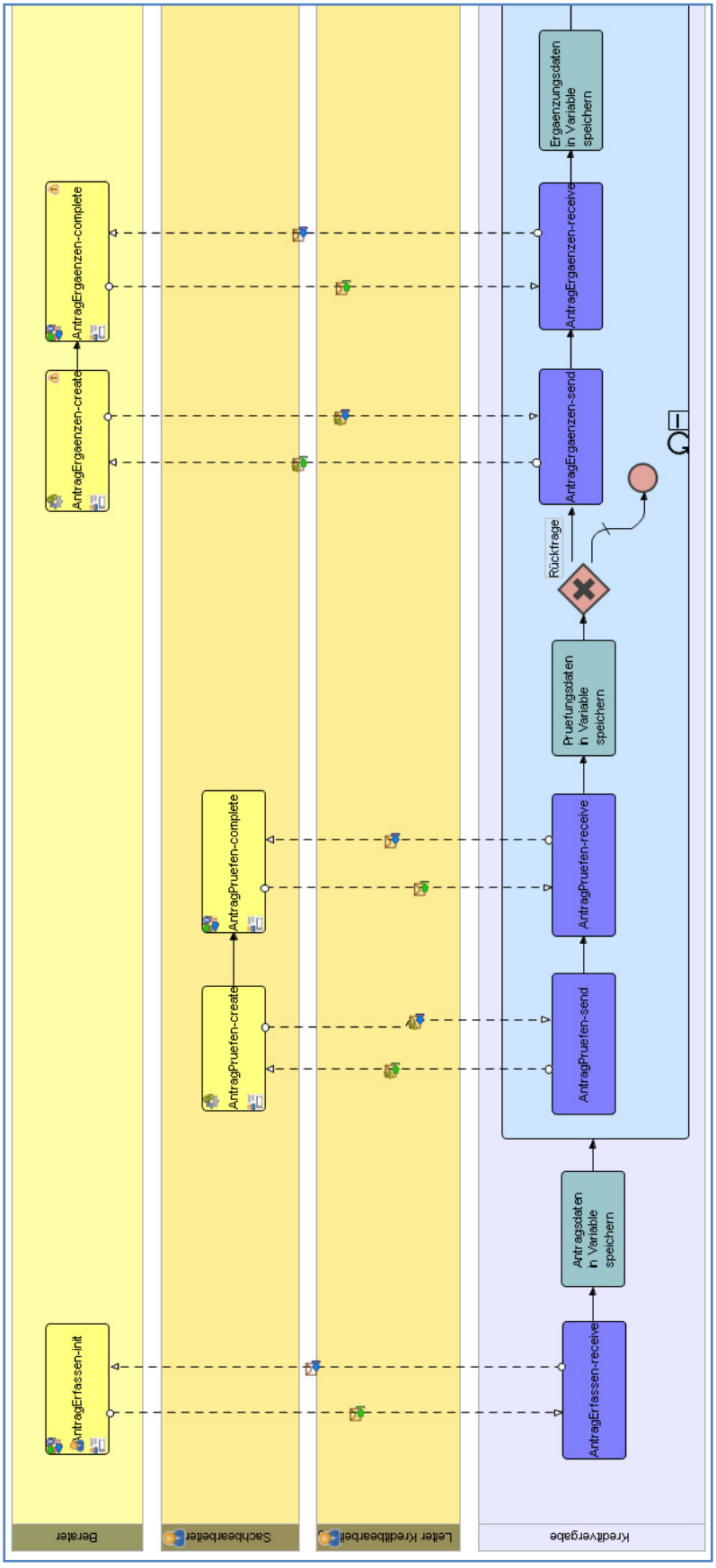

Abbildung 70: Das komplette *Workflow*-Modell (Teil 1)

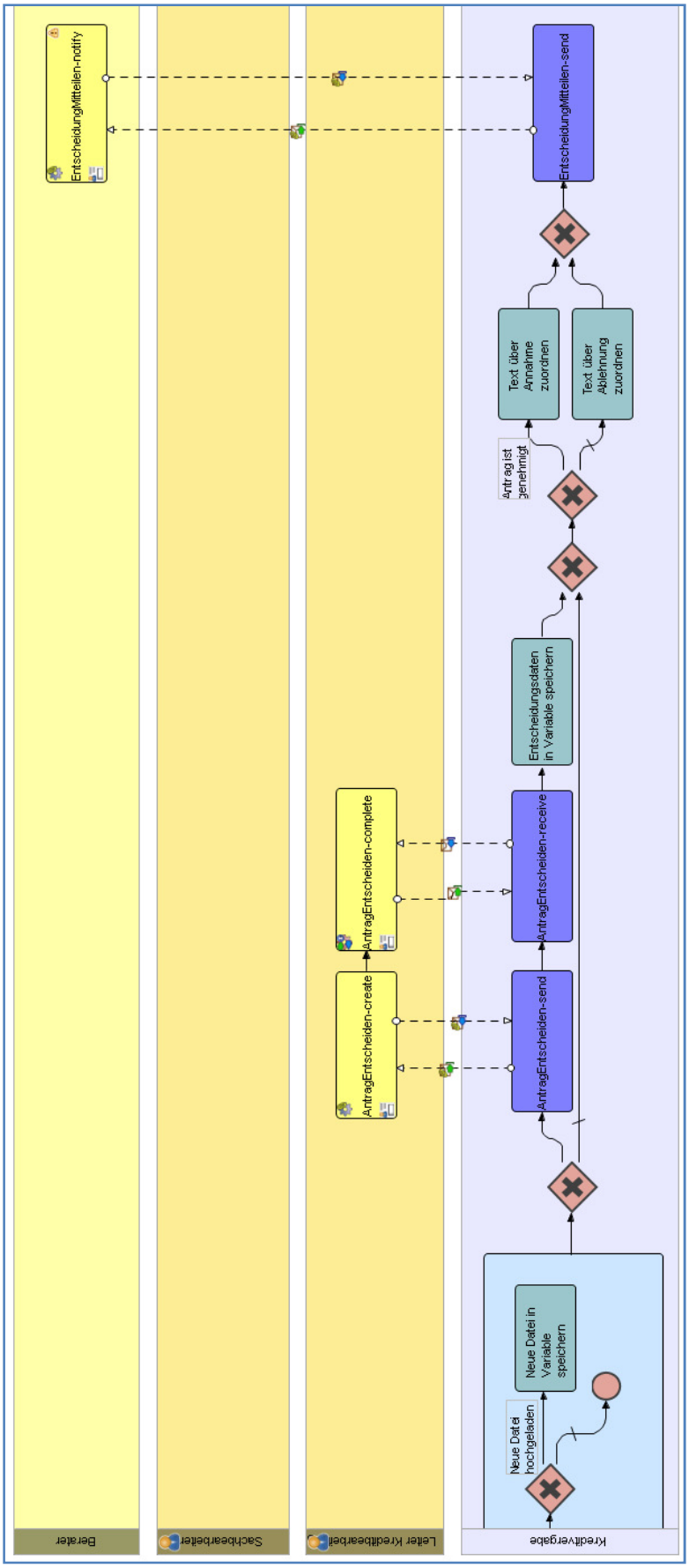

Abbildung 71: Das komplette *Workflow*-Modell (Teil 2)

4 Ein Durchlauf des Workflows

Im Folgenden wird ein beispielhafter Durchlauf aus Sicht der beteiligten Mitarbeiter dargestellt. Voraussetzung ist, dass das in Kapitel 3 entwickelte *Workflow*-Modell auf dem Server *deployed* wurde und der Server läuft. Wurde die Standard-Installation nicht verändert und wird vom Rechner, auf dem der Server läuft, darauf zugegriffen, so erreicht man die Benutzungs-Oberfläche, indem man in einen *Web-Browser* die Adresse http://localhost:8080/ui-fw eingibt.

4.1 Antrag erfassen

Der Berater Bernd Becker meldet sich mit seinem Benutzernamen „becker“ an.

Alle beteiligten Benutzer in dieser Demo haben das Passwort „password“.

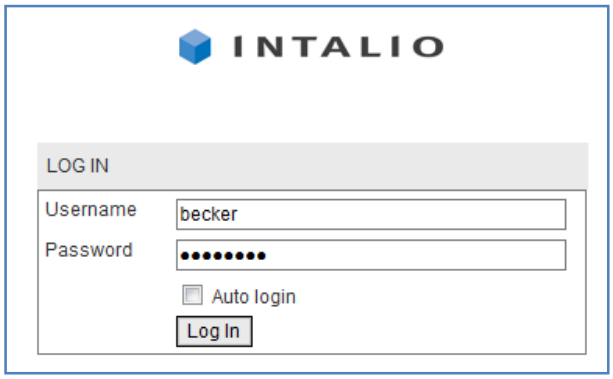

Abbildung 72: Anmeldung

In der Benutzeroberfläche findet er unter „*Processes*“ den Eintrag „Kreditantrag erfassen“.

Abbildung 73: Prozessauswahl

Er startet diesen Prozess, worauf sich der Eingabedialog zum Erfassen der Kreditantragsdaten öffnet (Abbildung 74). Muss-Felder sind rot hinterlegt. Felder, die besondere Eingaben verlangen, sind mit einem roten Ausrufezeichen gekennzeichnet. Fährt er mit der Maus auf das Ausrufezeichen so erhält er einen Hinweis auf die möglichen Eingabewerte (z. B. dass die Postleitzahl fünfstellig sein muss).

Abbildung 74: Dialog „Kreditantrag erfassen“

Er füllt den Dialog korrekt aus, worauf die Markierungen verschwinden. Außerdem wählt er mit „Durchsuchen“ eine Datei aus, die als Anlage zu dem Antrag hochgeladen wird.

Abbildung 75: Ausgefüllter Dialog „Kreditantrag erfassen“

Über die Schaltfläche „*Start process*“ schließt er die Eingabe ab und startet den *Workflow*.

4.2 Antrag prüfen

Sachbearbeiter Siegfried Schmidt (Benutzername: „schmidt“), erhält nun einen Eintrag in der Liste seiner *Tasks* (Aufgaben). Eventuell muss „*Refresh*“ (Auffrischen) gedrückt werden, damit die *Task*-Liste aktualisiert und die neu hinzu gekommene Aufgabe angezeigt wird. Auch alle anderen Benutzer mit der Rolle „Sachbearbeiter“ erhalten diesen Eintrag in ihrer *Task*-List.

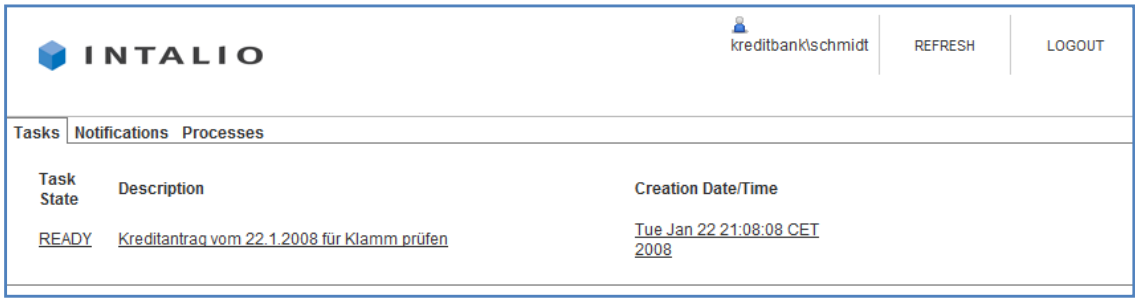

Abbildung 76: Aufgabe „Kreditantrag prüfen“ in der *Task*-Liste des Sachbearbeiters

Herr Schmidt öffnet den *Task* und bekommt den Dialog „Kreditantrag prüfen“ angezeigt (Abbildung 77). Er liest die Einträge und klickt auf den *Link* „Datei mit Verdienstbescheinigung etc.“, worauf sich das vom Berater hochgeladene Dokument in einem separaten Fenster öffnet. Die grau angezeigten Einträge können nicht geändert werden.

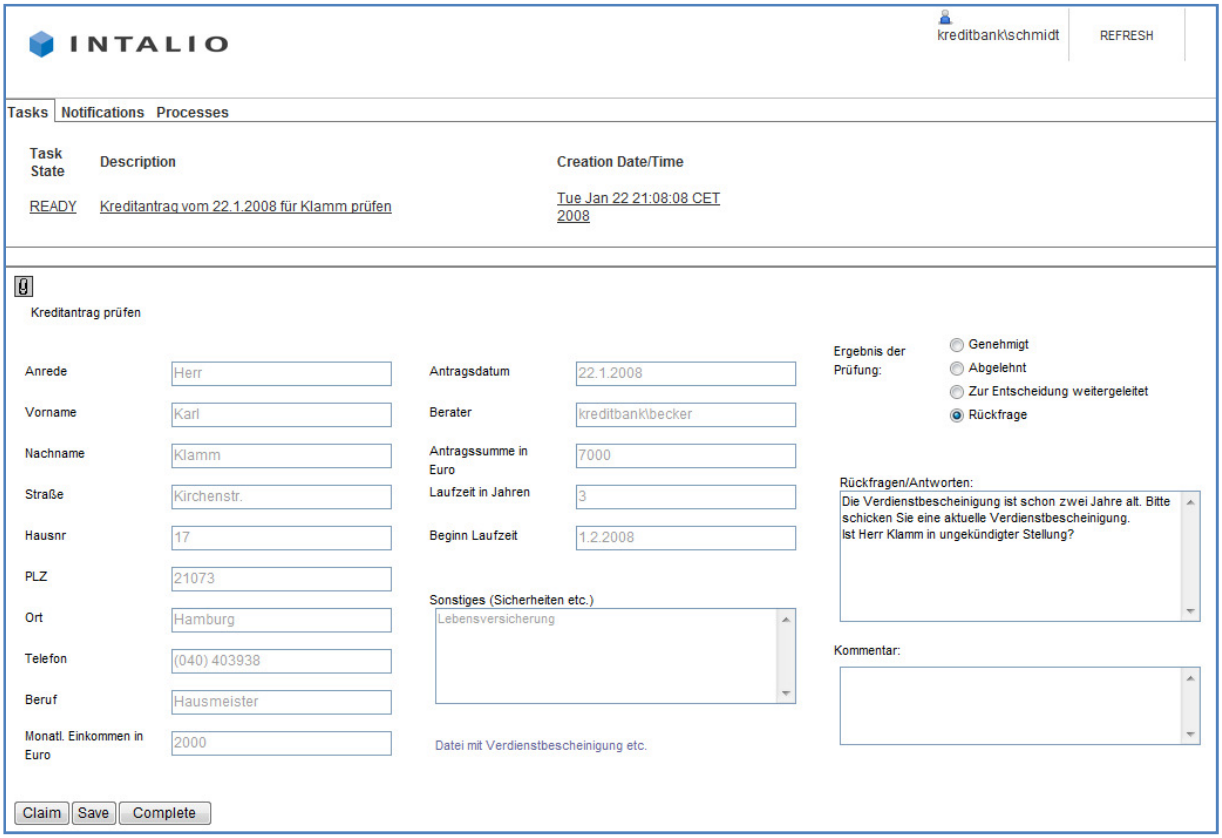

Abbildung 77: Dialog „Kreditantrag prüfen“

Da er noch einige Rückfragen hat, wählt er bei „Ergebnis der Prüfung“ den Eintrag „Rückfrage“ aus und trägt seine Fragen in das Feld Rückfragen/Antworten ein. Über die Schaltfläche „*Complete*“ schließt er die Antragsprüfung ab. Bei den anderen Benutzern mit der Rolle „Sachbearbeiter“ wird der Eintrag automatisch aus der *Task*-Liste entfernt.

4.3 Antrag ergänzen

Aufgrund der Rückfrage muss der Antrag vom Berater ergänzt werden. Herr Becker findet in seiner *Task*-Liste hierfür einen Eintrag. Andere Benutzer mit der Rolle „Berater“ erhalten diesen Eintrag nicht. So ist sichergestellt, dass der Berater den Antrag zur Ergänzung bekommt, der ihn auch gestellt hat. Öffnet er ihn, so erhält er den in Abbildung 78 gezeigten Dialog.

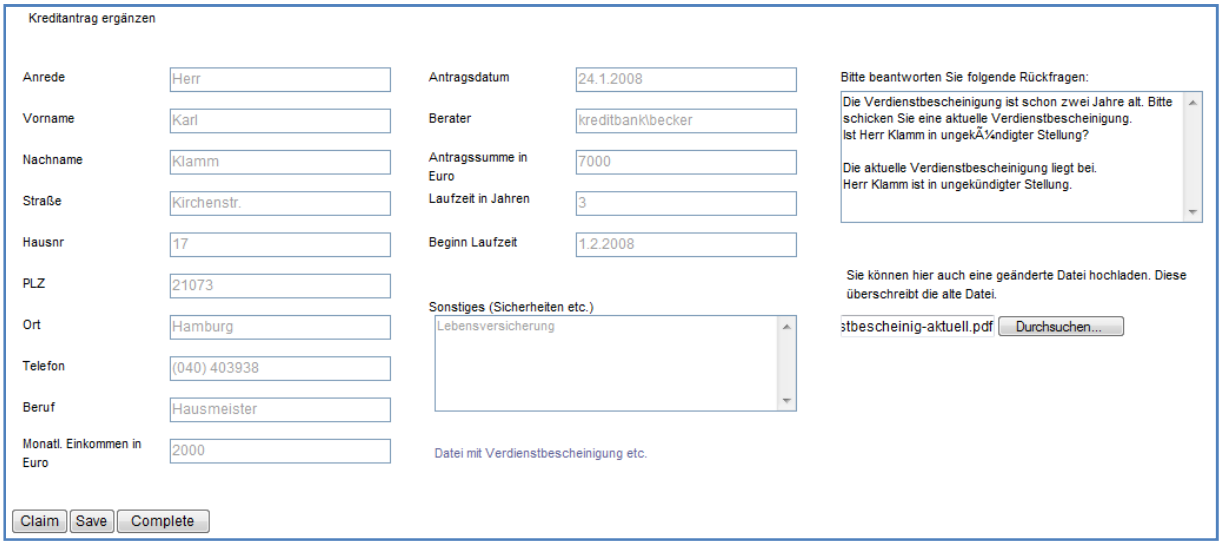

Abbildung 78: Dialog „Antrag ergänzen“

Hier beantwortet er die Rückfrage, lädt eine neue Datei hoch und schließt die Bearbeitung mit „*Complete*“ ab.

4.4 Zweiter Zyklus aus Prüfung, Ergänzung und neuer Prüfung

Nun erhalten wieder alle Benutzer mit der Rolle „Sachbearbeiter“ einen neuen *Task* zur Antragsprüfung. Die zweite Prüfung muss also nicht unbedingt von demjenigen Sachbearbeiter erledigt werden, der die auch die erste Prüfung durchgeführt hat.

Einer der Sachbearbeiter öffnet den Task. Da er noch eine weitere Rückfrage hat, wählt er wiederum die Option „Rückfrage“ aus und fügt seine neue Frage zu dem Text im Feld „Rückfragen/Antworten“ hinzu. Dies ist in Abbildung 79 dargestellt, wo auch deutlich wird, dass deutsche Umlaute aus den Eingaben leider nicht richtig verarbeitet werden.

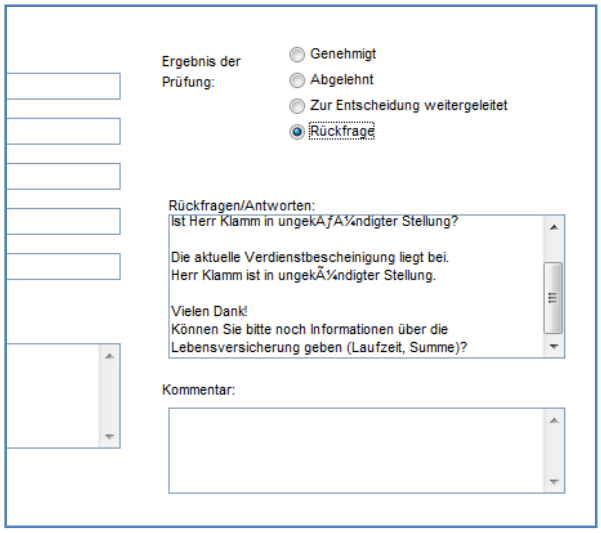

Abbildung 79: Erneute Prüfung des Antrags (Ausschnitt)

Als nächstes folgt eine neue Ergänzung durch den Berater, Herrn Becker (Abbildung 80). Diesmal lädt er keine neue Datei hoch.

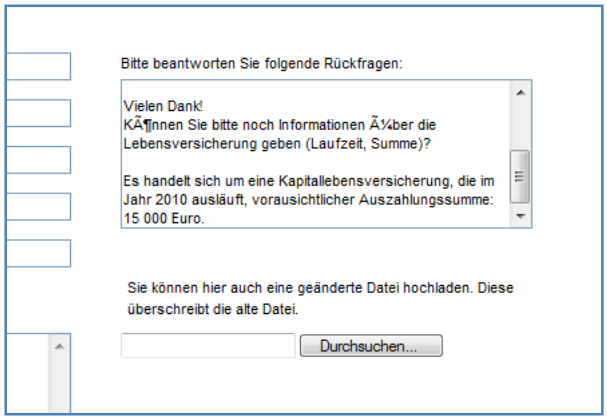

Abbildung 80: Erneute Ergänzung des Antrags (Ausschnitt)

Gemäß dem festgelegten Prozess ist nun noch einmal eine Prüfung erforderlich. Da der prüfende Sachbearbeiter diesmal keine weiteren Fragen mehr hat und zur Überzeugung gekommen ist, dass der Kreditantrag genehmigt werden soll, wählt er die Option „Genehmigt“ und trägt einen Kommentar in das entsprechende Feld ein (Abbildung 81).

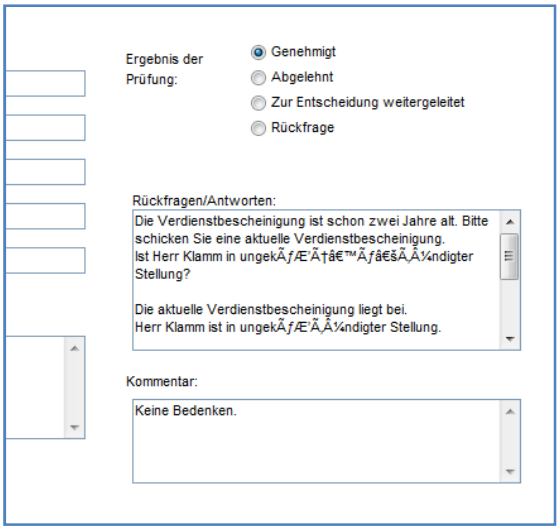

Abbildung 81: Dritte Prüfung des Antrags (Ausschnitt)

4.5 Genehmigung

Zwar hat der Sachbearbeiter bereits „Genehmigt“ ausgewählt, doch muss laut Prozessmodell ein Kreditantrag mit einer Summe von über fünftausend Euro von einem Leiter einer Kreditabteilung genehmigt werden.

Den entsprechenden Dialog für die Entscheidung über den Kreditantrag zeigt Abbildung 82. Der Leiter der Kreditabteilung sieht sich alle Daten an und wählt dann den Eintrag „Genehmigen“ aus.

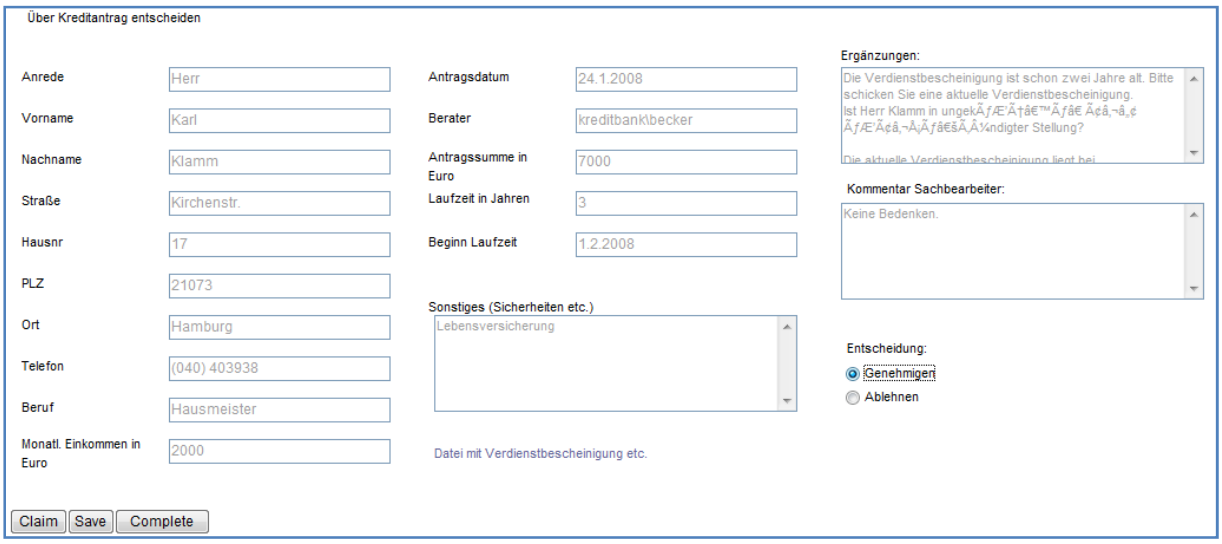

Abbildung 82: Entscheidung über den Antrag

4.6 Mitteilung über das Ergebnis des Antrags

Herr Becker erhält als Berater, der den Antrag gestellt hat, schließlich eine Mitteilung über das Ergebnis der Entscheidung. Da der Antrag genehmigt ist, enthält diese einen entsprechenden Text, in dem der positive Bescheid mitgeteilt wird. Damit ist der Ablauf beendet.

Herr Becker findet in seiner Benutzungsoberfläche einen Eintrag unter „*Notifications*“ (Benachrichtigungen). Hierüber lässt sich die in Abbildung 83 gezeigte Benachrichtigung öffnen. Mit „*Dismiss*“ (Verwerfen) wird der Dialog verlassen.

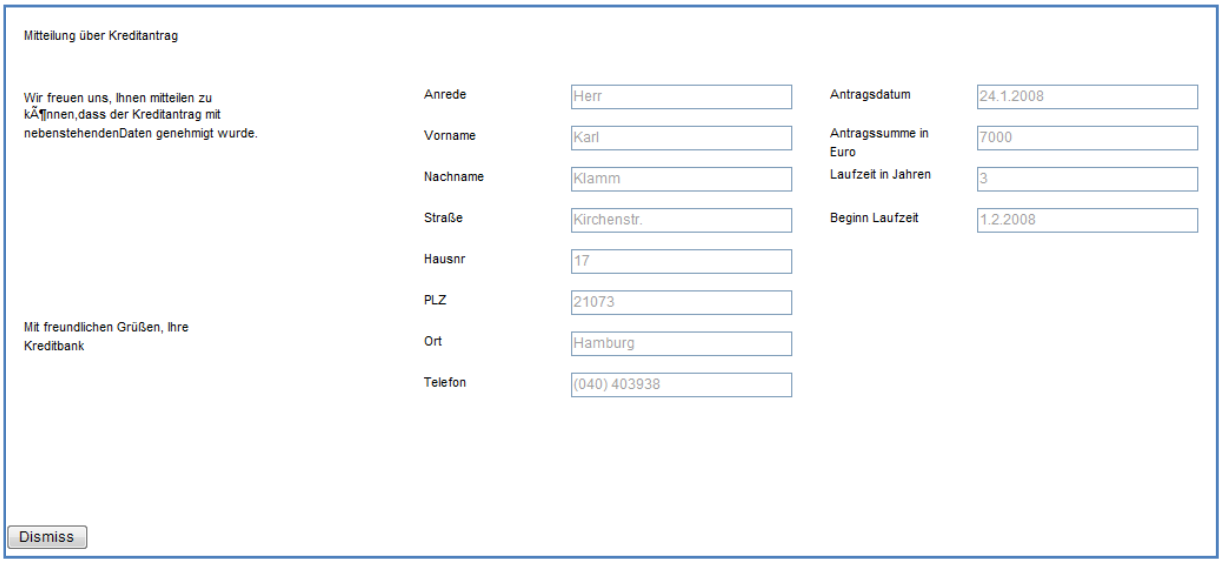

Abbildung 83: Mitteilung über das Ergebnis des Antrags

5 Zusammenfassung und Schlussfolgerungen

Im vorliegenden Arbeitspapier wurde ein möglicher Weg von fachlichen Modellen zum ausführbaren *Workflow* beschrieben. Die fachlichen Modelle wurden in ARIS erstellt, als *Workflow Management*-System wurde Intalio|BPMS eingesetzt. Hierbei wurden sämtliche durchgeführten Schritte detailliert beschrieben. Sie können mit Hilfe der frei verfügbaren *Community Edition* des Intalio|BPMS von jedem Interessierten nachvollzogen werden.

Das gewählte Beispiel der Bearbeitung eines Kreditantrages ist einerseits aus Komplexitäts- und Nachvollziehbarkeitsgründen gegenüber einem realen Prozess noch deutlich vereinfacht. Andererseits ist es in der umgesetzten Form wesentlich vollständiger und umfassender als die typischen Demonstrationsbeispiele zum Ausprobieren eines BPMS. So war es erforderlich, eine Reihe von Detailproblemen zu lösen, wie sie in realen Projekten ebenfalls auftreten. Oftmals ist zwar die grundlegende Funktionalität rasch realisiert, aber die Lösung von Problemen und die technische Umsetzung erforderlicher Details erzeugen einen hohen Aufwand. Hierdurch vermittelt das Beispiel zumindest einen Eindruck, wie sich die Entwicklung eines *Workflows* für reale Prozesse gestalten würde.

Für das Beispiel wurde darauf geachtet, komplett die Standard-Modellierungsfunktionalität von Intalio|BPMS zu verwenden, so dass keinerlei Programmierung im herkömmlichen Sinne erfolgte. Der *Workflow* sollte ausschließlich Aktivitäten umfassen, die von menschlichen Benutzern mit Hilfe der von Intalio bereit gestellten Benutzungsoberfläche durchgeführt werden („*Human Centered Workflow*“), d. h. es fand beispielsweise keine Integration externer *Web Services* o. ä. statt.

Die wesentlichen Erfahrungen bei der Entwicklung und Durchführung des beschriebenen Vorgehens unter Verwendung von ARIS-Modellen und Intalio|BPMS lassen sich folgendermaßen zusammenfassen:

- Das Vorgehen erwies sich insgesamt als geeignet. Allerdings überstieg der Zeitaufwand für die Realisierung die ursprünglichen Schätzungen um ein Vielfaches. Zu einem Teil lag dies an den mangelnden Kenntnissen von Intalio|BPMS. Leider liegt auch keine durchgängige Dokumentation dieses Systems vor. Zwar findet sich ein Großteil der benötigten Informationen in diversen Tutorials und Beispielen auf der Intalio Webseite, doch ist es im Bedarfsfall oftmals schwierig, diese zu finden. Doch auch bei guter Kenntnis des Systems ist der Entwicklungsaufwand nicht zu unterschätzen.
- Die verwendeten Modelle (EPK, Organigramm, Klassendiagramm) bildeten zusammen mit den eingesetzten Attributen eine gute Grundlage für die Entwicklung eines Workflow-Modells. Allerdings konnten nicht alle für die Umsetzung benötigten fachlichen Informationen komplett mit den verwendeten Modelltypen dargestellt werden. So ist etwa die Forderung, dass eine Rückfrage nicht an jeden beliebigen Benutzer der Rolle „Berater“ weiter geleitet wird, sondern nur genau an den, der den betreffenden Antrag gestellt hat, nicht explizit in den Modellen enthalten. Auch lässt sich den Modellen nicht entnehmen, was es genau bedeutet, wenn eine Aktivität ein zweites Mal durchgeführt wird: Werden die Daten noch einmal komplett neu erfasst, oder die vorhandenen Daten geändert oder ergänzt?
- Zum Teil haben sich derartige Fragen von fachlicher Relevanz erst während der technischen *Workflow*-Modellierung ergeben. Es ist daher eine kontinuierliche Abstimmung mit den

Fachexperten erforderlich, auch wenn bereits ein gutes fachliches Modell vorhanden ist. Wenn ausreichend Erfahrung mit der *Workflow*-Entwicklung vorliegt, können die Konventionen für die fachliche Modellierung ggf. entsprechend erweitert werden, um die Zahl der erforderlichen Rückfragen zumindest bei typischen Problemstellungen zu reduzieren.

- Die angewandte Art der fachlichen Modellierung nutzt zwar ausschließlich häufig in der Praxis eingesetzte Modellierungsmethoden und –konstrukte, doch fand die Auswahl der benötigten Konstrukte und insbesondere der verwendeten Attribute bereits im Hinblick auf die spätere Umsetzung in das *Workflow*-System statt. Die für die fachliche Modellierung angewandten Modellierungskonventionen sind daher auf die *Workflow*-Modellierung abzustimmen.
- Die inkrementelle Vorgehensweise hat sich bewährt. Hierbei wurde zuerst ein kurzer, aber bereits funktionierender *Workflow* entwickelt, getestet und dann sukzessive weiter entwickelt. Die getesteten Teil-*Workflows* eignen sich einerseits gut zur Validierung durch Fachanwender, andererseits lassen sich Fehler schneller entdecken und besser lokalisieren als wenn der *Workflow* zunächst komplett entwickelt wird.
- Die *Workflow*-Modellierung im Intalio|BPMS Designer ist recht intuitiv. Dennoch eignet sich die Modellierungskomponente eher für zumindest IT-affine Modellierer als für reine Fachexperten, da man recht schnell auch mit technischen Details in Berührung kommt.
- Das Rollenmodell ließ sich – aufgrund der einfachen Struktur – direkt umsetzen. Allerdings sind hierzu XML-Kenntnisse erforderlich.
- Das Datenmodell ließ sich ebenfalls sehr einfach in eine entsprechende Datenstruktur umsetzen. Hierfür sind zumindest grundlegende XML-Kenntnisse erforderlich. Zwar vereinfacht der XML *Schema Editor* die Erstellung eines Schemas und macht den direkten Kontakt mit dem XML-*Code* weitgehend überflüssig, doch ist zum Verständnis der verwendeten Konzepte XML-Basiswissen notwendig. Zur Klärung spezieller Fragen war an einigen Stellen trotz allem ein Blick in den XML-*Code* oder gar ein direkter Eintrag im *Code* notwendig. Die Umsetzung spezieller Restriktionen (z. B. zum korrekten Aufbau von Postleitzahlen, Telefonnummern oder Geldsummen) verlangt umfassendes XML-Detailwissen und die Beherrschung regulärer Ausdrücke.
- Die Umsetzung der EPK in das BPMN-*Workflow*-Modell war im Gegensatz zu anderen Modellen nicht so direkt möglich. Bei ausreichender Kenntnis von Intalio|BPMS ist dies zwar nicht besonders schwierig, doch unterscheiden sich EPK und BPMN-*Workflow*-Modell von ihrer Struktur her deutlich voneinander. So müssen einzelne fachliche Aktivitäten je nach Art der Benutzerinteraktion in bis zu fünf BPMN-Aktivitäten umgewandelt werden. Die Darstellung der Kontrollflusslogik unterscheidet sich grundlegend (z. B. Schleifen, Formulierung von Bedingungen).
- Die Daten-*Mappings* lassen sich mit dem *Mapper* recht einfach erstellen, solange eine unveränderte Abbildung eines Attributs auf ein anderes erforderlich ist. Berechnungen und Transformationen unter Verwendung von *XPath*-Funktionen erwiesen sich hingegen als recht kompliziert und erfordern sehr fundiertes XML-Wissen. Z. T. ist auch recht viel Kreativität gefragt, um bestimmte Transformationen zu realisieren.
- Die Modellierung der Dialoge im *Form Editor* ist ebenfalls sehr intuitiv, allerdings sind die angebotenen Möglichkeiten der Oberflächengestaltung recht eingeschränkt. Auch erfordert die Erstellung optisch akzeptabler Dialoge (bei denen z. B. alle Beschriftungen und alle Eingabefelder linksbündig untereinander ausgerichtet sind) einiges Herumexperimentieren. Die

Formulierung von Bedingungen zur Eingabeüberprüfung erfordert wiederum fundierte XML-Kenntnisse.

- Die reine Umsetzung und Modellierung der Kontrollflusslogik war vergleichsweise wenig aufwändig. Die beträchtlichen Aufwände entstanden bei der Erstellung der Dialoge mit den erforderlichen Eingabeüberprüfungen und bei der Definition der *Mappings*, insbesondere bei der Formulierung komplexer Transformationsregeln.