Verteilte Datenhaltung in der Landwirtschaft auf Basis von agroXML

Daniel Martini, Mario Schmitz, Jürgen Frisch, Martin Kunisch

Kuratorium für Technik und Bauwesen in der Landwirtschaft e. V. (KTBL)
d.martini@ktbl.de, m.schmitz@ktbl.de, j.frisch@ktbl.de, m.kunisch@ktbl.de

Abstract: Die eXtensible Markup Language wurde entworfen, um semantisch reichhaltige, verknüpfte Dokumente zu erstellen. Durch ein Zusammenspiel verschiedener standardisierter Internettechnologien, wie HTTP und XLink können Daten in lokale Datenbestände eingebunden werden. Dienste können dann mit Hilfe des Representational State Transfer (ReST) ohne weitere komplexe Protokolle auf einfache Art und Weise aufgebaut werden. Mit agroXML ist die Umsetzung dieser ressourcenorientierten Vorgehensweise bei der Erstellung von Diensten möglich. Im vorliegenden Fall wurde die Methode genutzt, um beispielhaft die Zusammenstellung von Schlachtchargen in der Schweinehaltung abzubilden.

1 Einleitung

In den letzten Jahren wurde agroXML – entworfen und entwickelt vom Kuratorium für Technik und Bauwesen in der Landwirtschaft auf Basis der eXtensible Markup Language (XML) – mehr und mehr zur Standard-Auszeichnungssprache für landwirtschaftliche Daten. Für weite Bereiche der Domäne Landwirtschaft wurden Dokumentenstrukturen mit Hilfe von XML Schema festgelegt. Inzwischen wurden auch erfolgreich Anwendungen zum bilateralen Datenaustausch unter Nutzung von agroXML umgesetzt. Entworfen wurde XML aber, um semantisch reichhaltige, miteinander verknüpfte Dokumente erstellen und ablegen zu können. Das heißt, der Netzwerkgedanke ist in den vom W3C gesehenen Anwendungsfällen für XML deutlich betont und hier liegt auch das Alleinstellungsmerkmal der Technologie. Sie hat das Potenzial, verteilte Ablage von Daten – wenn nötig auch weltweit – sowohl auf dem Betrieb selbst als auch zwischen Betrieben und externen Partnern zu ermöglichen.

3 Material und Methoden

Der grundlegende Aufbau des Systems orientiert sich an der von Jacobs und Walsh [Ja04] beschriebenen Architektur. Durch das Zusammenspiel mehrerer standardisierter Komponenten entsteht ein einfacher, sogenannter ReSTful Webservice.

Die fachlich-inhaltliche Ebene wird dabei durch agroXML abgedeckt. Sie liefert die notwendigen Elemente und Datentypen um die Sachverhalte der Landwirtschaft in

XML-Dokumenten darzustellen. agroXML ist durch einen Satz von XML Schemas und Inhaltslisten definiert, die unter http://www.agroxml.de/schema/ bzw. unter http://www.agroxml.de/content/ verfügbar sind. Nähere Informationen zur Konzeption und zu den Inhalten können den Publikationen von Doluschitz et al. [Do04], Kunisch et al. [KBF07] und Martini et al. [MFK07] entnommen werden.

XLink [De01] wird verwendet als Technologie, mit der die Verknüpfung von SGML-Dokumenten ermöglicht wird. Als eindeutige Bezeichner einer Ressource werden dabei Uniform Ressource Identifier oder – in der Praxis verbreiteter – deren Untermenge, die Uniform Ressource Locator eingesetzt. XLink stellt mehrere Typen von Verknüpfungen bereit: „simple“, „extended“, „locator“, „arc“, „resource“, „title“ und „none“. Im Folgenden wird lediglich der Typ „simple“ betrachtet. Im Aufbau ähnelt XLink den <a>(anchor)-Elementen mit ihren href-Attributen in HTML. Auch in XLink existiert ein href-Attribut, das in einem XML Schema beliebigen Elementen zugewiesen werden kann. Es gibt aber noch eine Reihe anderer Attribute, die es ermöglichen die Verknüpfung näher zu beschreiben oder einem Programm Hinweise zur Verarbeitung zu geben. Mit Hilfe der Technologie wird im Grunde genommen ein Graph aufgebaut. Knoten („nodes“) dieses Graphen sind dabei die Dokumente, die XLinks sind die sogenannten „arcs“, die Verbindungen zwischen den Knoten. Dementsprechend gibt es beispielsweise ein arcrole-Attribut, das die erstellte Relation näher beschreibt.

Der gezeigte Service wurde nach dem Paradigma des Representational State Transfer (ReST) [Fi00] aufgebaut. ReST geht von der Annahme aus, dass mit wenigen einfachen Operationen zum Lesen und Schreiben von Daten und einem System, das seinen Zustand in Abhängigkeit von diesen Operationen ändert, die meisten Anwendungsfälle im Bereich der Kommunikation abgebildet werden können. Dieses Konzept zieht sich in jeweils leicht abgewandelter Form wie ein roter Faden durch die Geschichte der Informatik, angefangen bei der Turing-Maschine [Tu36] über Datei- und Datenbanksysteme im Allgemeinen, über das sogenannte Create-Read-Update-Delete-pattern (CRUD, [Ki90]) bis zum Standard des Hypertext Transfer Protocol (HTTP) [FGM99]. Diesem Muster folgend können auch Webservices aufgebaut werden. Die Datenübertragung erfolgt dabei lediglich über HTTP. Optional können weitere Funktionalitäten über bewährte und bereits breit im Einsatz befindliche Erweiterungen des Protokolls wie beispielsweise SSL/TLS zur Verschlüsselung oder HTTP AUTH zur Authentifizierung aufgesetzt werden. Auf komplexe weitere Ebenen wie z. B. SOAP kann verzichtet werden. Funktionen werden nur durch Lesen oder Schreiben einer bestimmten Ressource aufgerufen.

4 Prototyp

Ein zur Demonstration der Möglichkeiten entwickelter Prototyp geht von folgendem Beispielanwendungsfall aus: Die Mast ist für eine Gruppe von Tieren beendet. Der Landwirt stellt infolgedessen eine Schlachtcharge zusammen. Daten über die Tiere existieren bereits. Auch Daten über den Verladeort sind vorhanden. Aufgabe ist es, diese Informationen für eine Charge zusammenzufassen und in einer über das Netzwerk maschinenlesbaren Struktur abzubilden. Eine wichtige Anforderung dabei ist es, die Daten so bereitzustellen, dass sie mit möglichst geringem Aufwand in eine Infrastruktur zur

Rückverfolgbarkeit und Qualitätssicherung, an der weitere Partner innerhalb der Lebensmittelkette beteiligt sind, eingebunden werden können.

Die für diesen und andere Anwendungsfälle aus der Tierhaltung notwendigen Datentypen und Elemente wurden in agroXML erstellt. Nach Möglichkeit wurde auf die für den Pflanzenbau bereits vorhandenen Datentypen und Elemente zurückgegriffen. Das ist bei den Betriebsstammdaten der Fall, aber auch grundlegende Datentypen beispielsweise für physikalische Größen konnten wiederverwendet werden. Für ein Schwein können zum Beispiel Daten zum Geschlecht, zur Ohrmarke, sowie zu Ereignissen, die sich auf das Tier beziehen, wie z. B. eine Wägung in XML-Instanzen abgelegt werden. Im Rahmen des Forschungsprojektes konnte in dem Zusammenhang auf Einzeltiererkennung zurückgegriffen werden. Anstattdessen können auch Tiergruppen erfasst werden. Es ist dann aber nicht möglich, die aufgeführten, einzeltierbezogene Daten anzugeben.

Ressourcen werden in ReSTful Webservices URLs zugeordnet, über die sie abrufbar sind. An diesem Ort kann sich dann die XML-Instanz befinden, die z. B. ein Objekt näher beschreibt. Um nun eine Charge zusammenzustellen, müssen die Objekte, aus denen sie sich zusammensetzt, verknüpft werden, so dass eine Beispielinstanz (verkürzt und vereinfacht dargestellt ohne XML-Prolog, XLink-Typ- und namespace-Deklarationen und Schema-Locations) so aussehen kann:

```
<Charge id="ID_1">
  <LoadLocation xlink:href="http://example.com/farms/farm_01.xml"/>
  <ChargedObject xlink:href="http://example.com/animals/pig_2348.xml"/>
  <ChargedObject xlink:href="http://example.com/animals/pig_1975.xml"/>
</Charge>
```

Im vorliegenden Beispiel wird also eine Charge aus zwei Schweinen gebildet, deren Daten sich über den Aufruf der im `xlink:href`-Attribut gegebenen URLs holen lassen. Im Sinne einer generischen Modellierung kann das `<ChargedObject>`-Element z. B. auch auf ein Rind oder einen Sack Getreide zeigen. Das Element `<LoadLocation>` enthält den Verweis auf den Ort, an dem die Beladung erfolgte, auch hier ist der mögliche Bezug nicht auf den Betrieb beschränkt. Auch Ställe oder andere Orte könnten hier eingebunden werden. Insgesamt ergibt sich als URL-Struktur für den Dienst:

Betriebsstammdaten: http://example.com/farms/*
Daten zu Tieren: http://example.com/animals/*
Daten zu Chargen: http://example.com/charges/*

Eine Clientanwendung kann automatisiert und selbständig durch die sich durch die Verknüpfungen ergebende Netzwerkstruktur navigieren, und Daten abrufen. Zu beachten ist hierbei, dass die XLinks an beliebige Orte verweisen können. In weiteren Beispielszenarien können so auch Daten externer Informationsanbieter eingebunden werden.

4 Diskussion

Die gezeigte Anwendung hätte auch nach message-orientierten Paradigmen, z. B. unter Nutzung von SOAP, aufgebaut werden können. Dabei würden Methoden zum Hinzufügen eines Schweines zu einer Charge oder zum Abruf von Chargendaten usw. erstellt.

Diese Vorgehensweise stellt zunächst geringere Ansprüche an die Entwicklung eines funktionalen Designs. Sobald aber weitere Datenanbieter in die Infrastruktur eingebunden werden sollen, zeigen sich eklatante Nachteile. So müssen die möglichen Methodenaufrufe und deren Parameter für jede Anwendung, die angebunden werden soll, bekannt gemacht werden. Ein Client, der auf eine große Anzahl an Diensten, beispielsweise entlang der gesamten Lebensmittelkette zugreifen will, muss sämtliche Details der Dienste im Voraus kennen. Das im Rahmen von serviceorientierten Architekturen propagierte, dynamische Anbinden des Client zur Laufzeit ist in der Praxis derzeit noch kaum möglich. Die oben beschriebene Dienstearchitektur mit ihrem beschränkten Methodenumfang hingegen erlaubt durch einen beliebig großen Satz von Ressourcen ohne Kenntnis des kompletten Satzes zu navigieren. Zur Umsetzung wird lediglich ein XML-Parser sowie eine Implementation des HTTP benötigt. Diese sind für ein breites Spektrum an Hardwareplattformen von stromsparenden, handlichen Mobilgeräten bis zu leistungsstarken Servern und für nahezu jede Programmierumgebung erhältlich. Inzwischen existieren auch abstrahierende APIs und Frameworks für ReST, die das Aufsetzen eines solchen Dienstes vereinfachen. Aber auch die Implementation in Programmiersprachen mit weniger hohem Abstraktionsgrad, z. B. als Modul zum Apache Webserver in C, ist mit vertretbarem Aufwand möglich.

Wir danken dem Bundesministerium für Bildung und Forschung (BMBF) für die Förderung der Arbeiten im Rahmen des IT FoodTrace-Verbundprojektes (FKZ 0330761). Eine beispielhafte Umsetzung erfolgte gemeinsam mit dem Teilprojekt „Farming Cell“. Nähere Informationen hierzu finden sich in Herd et al. [He08]. Wir bedanken uns an dieser Stelle für die gute Zusammenarbeit.

Literaturverzeichnis

[De01] DeRose, S., Maler, E., Orchard, D. : XML Linking Language (XLink) Version 1.0. World Wide Web Consortium, 2001. http://www.w3.org/TR/xlink/

[Do04] Doluschitz, R., Kunisch, M.: agroXML - ein standardisiertes Datenformat für den Informationsfluss entlang der Produktions- und Lieferkette. In: Zeitschrift für Agrarinformatik, Ausgabe 12/4, Jahrgang 2004.

[FGM99] Fielding, R. T., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: RFC2616: Hypertext Transfer Protocol – HTTP 1.1. 1999.

[Fi00] Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures. Dissertationsschrift University of California, Irvine, 2000.

[He08] Herd, D., Kuhlmann, A., Martini, D., Kunisch, M., Friedrichs, E.: Technische Möglichkeiten zur Verbesserung der Prozessdokumentation und Rückverfolgbarkeit in der Schweinehaltung. In: KTBL-Schrift 469, 2008.

[Ja04] Jacobs, I., Walsh, N.: Architecture of the World Wide Web, Volume One. World Wide Web Consortium, 2004, http://www.w3.org/TR/webarch/.

[KBF07] Kunisch, M., Böttinger, S., Frisch, J.: agroXML – der Standard für den Datenaustausch in der Landwirtschaft. In: KTBL-Schrift 454, Darmstadt, 2007.

[Ki90] Kilov, H.: From semantic to object-oriented data modeling. In: Proceedings of the First International Conference on Systems Integration, 1990.

[MFK07] Martini, D., Frisch, J., Kunisch, M.: agroXML Inhaltslisten – Konzeption und Inhalte. In: Lecture Notes in Informatics - Proceedings 27. GIL-Jahrestagung, Stuttgart, 2007.

[Tu36] Turing, A. M.: On Computable Numbers, with an Application to the Entscheidungsproblem. In: Proceedings of the London Mathematical Society, 2 42, 1936.