Industrielle VT-Anwendungen auf Basis von Web-Technologien
Industrial VR/AR Applications based on Web Technologies

Andreas Harth[1], Tobias Käfer[1], Felix Leif Keppmann[1], Dmitri Rubinstein[2], René Schubotz[2], Christian Vogelgesang[2]
[1] AIFB, Karlsruher Institut für Technologie, *vorname.nachname*@kit.edu
[2] Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, *vorname.nachname*@dfki.de

Kurzfassung

Virtuelle Techniken (VT) sind weitgehend ausgereift; auf VT aufbauende Systeme sind in interaktiven Anwendungen bereits in der Industrie im Produktiveinsatz. Die Anforderungen in industriellen VT-Systemen ähneln denen von Industrie-4.0-Szenarien sowie denen im Internet der Dinge: Unterschiedlichste Komponenten mit inkompatiblen Datenformaten und Zugriffsprotokollen müssen bei gleichzeitig hohen Anforderungen an Latenz und Durchsatz zusammenarbeiten. Die in diesem Bereich traditionell vorherrschenden monolithischen Architekturen beschränken sowohl die Erweiterbarkeit als auch die Wiederverwendbarkeit einzelner Komponenten. In diesem Beitrag stellen wir die im Rahmen des Projekts ARVIDA entwickelte und auf Web-Technologien basierende Architektur für VT-Systeme vor, mit der einzelne semantisch ausgezeichnete Komponenten über das Netzwerk flexibel angesprochen werden können. Das Aufbrechen der monolithischen Strukturen erlaubt uns im Projekt die Komposition von komplexen Anwendungen aus heterogenen Komponenten. Zur Spezifikation der dafür notwendigen Anwendungslogik verwenden wir eine Regelsprache.

Abstract

Virtual Technologies (VT) are considered sufficiently mature; many interactive VT applications are already in productive use in industry. The requirements of these systems are quite similar to the industrial internet and the internet of things: Different components with proprietary protocols and data formats have to work together in an application without violating the requirements on latency and throughput. But the traditional monolithic VT systems restrict the extensibility and reusability of single components. In this paper, we present an architecture for VT systems based on semantic web technologies developed in the ARVIDA project. Each component is semantically annotated and accessible via network protocols. The modular design enables the composition of complex distributed applications from heterogeneous components. We use a rule language for the specification of the required application logic.

1 Einführung

Unter dem Begriff der Virtuellen Techniken (VT) werden interaktive Anwendungssysteme entlang des gesamten Realität-Virtualität-Kontinuums [18] der Mixed Reality verstanden. Der Einsatz der Virtuellen Techniken ist beispielsweise in den Anwendungsfeldern Entwurf und Entwicklung [20], Training und Schulung [9] sowie Werkerunterstützung [17] bei Industrieunternehmen schon seit einiger Zeit ein fester Bestandteil der Prozess- und IT-Systemlandschaft. Dabei erfordern industrielle VT-Anwendungssysteme, insbesondere jene unter Nutzung von Virtueller Realität (VR) und Erweiterter Realität (AR), ein reibungsloses Zusammenspiel verschiedener technologischer Komponenten. Hier seien virtuelle Menschmodelle, Motion-Capturing-Systeme, ingenieurwissenschaftliche Simulations- und Analysemodule oder haptische Ausgabegeräte exemplarisch genannt. Die funktionalen Anforderungen an solche VT-Anwendungssysteme steigen dabei stetig an, bis hin zu dem – für Industrie 4.0 zentralen [21] – Anspruch, die Prozess- und Entscheidungsschritte ausschließlich anhand virtueller Produktmodelle vornehmen zu können.

Die historische Entwicklung und die industriellen Anforderungen haben zu sehr funktionsreichen, aber proprietären VT-Anwendungssystemen geführt. Insbesondere VR-Systeme sind in vielen Fällen eng verzahnte Lösungen aus einem Grundsystem sowie spezialisierten Funktionskomponenten. Dabei setzen moderne VR-Systeme zwar häufig ein Modulkonzept um, jedoch sind diese Funktionsmodule meist nur innerhalb ihrer Produktfamilie interoperabel. Die Forderung nach einer durchgängigen, flexiblen und kostengünstigen VT-Unterstützung von industriellen Entwurfs-, Entwicklungs- und Produktionsprozessen in Industrie 4.0 [19] ist gleichsam eine Forderung nach verbesserter *Erweiterbarkeit*, *Wiederverwendbarkeit* und *Komponierbarkeit* von industriellen VT-Anwendungssystemen – und das über die Grenzen von proprietären Produktfamilien oder gar Organisationen hinweg.

Im Projekt ARVIDA[1] erarbeiten wir Konzepte, Technolo-

[1] ARVIDA (Angewandte Referenzarchitektur für virtuelle Dienste und Anwendungen) ist ein vom BMBF gefördertes (FKZ 01IM13001A, FKZ 01IM13001G) Verbundprojekt mit 21 Partnern aus Industrieunternehmen, kleinen und mittleren Unternehmen sowie akademischen Partnern. Das Konsortium wird von Prof. W. Schreiber von der Volkswagen AG geleitet. Das Projekt ist im September 2013 gestartet und läuft für drei Jahre `http://www.arvida.de/`

gien und Verfahren, um die Entwicklung von durchgängigen VT-Anwendungen für Industrie 4.0 zu ermöglichen durch einen konsequenten Einsatz von bereits etablierten sowie im Projekt ARVIDA erforschten Web-Standards und Technologien.
Die Beiträge und Struktur dieser Arbeit sind wie folgt: Wir beschreiben den verfolgten Ansatz sowie die zum Einsatz gebrachten Technologien (Abschnitt 2). Hierbei sind das REST-Paradigma [11], die damit verbundene Ressourcenzentrierte Dienstabstraktion sowie semantische Beschreibungen [13] von zentraler Bedeutung. Nachfolgend erläutern wir Konzepte und Technologien zur Bereitstellung von VT-Komponenten als Dienste (Abschnitt 3). Schließlich stellen wir einen regelbasierten Ansatz zur Erstellung und Ausführung von VT-Anwendungssystemen vor (Abschnitt 4) und skizzieren interaktive Demonstratoren, welche auf Basis der ARVIDA-Referenzarchitektur umgesetzt wurden (Abschnitt 5). Wir schließen mit einem Fazit (Abschnitt 6).

2 Ansatz und Technologien

Durch die Erforschung und Erarbeitung einer „Angewandten Referenzarchitektur für Virtuelle Dienste und Anwendungen“ sollen Web-Technologien und entsprechende Architekturstile für moderne dienstorientierte VT-Anwendungen nutzbar gemacht und so die Verbreitung von VT erleichtert werden. Für die Referenzarchitektur verwenden wir Technologien aus dem Web-Umfeld bzw. adaptieren diese Technologien, um den speziellen Anforderungen von VT-Anwendungen gerecht zu werden. Der Bau von VT-Anwendungen in einer solchen Architektur besteht dann aus zwei Schritten: Im ersten Schritt werden klar abgegrenzte eigenständige Komponenten erstellt, welche mittels Web-Technologien verfügbar gemacht werden und deren Daten semantisch beschrieben sind (siehe Abschnitt 3). Im zweiten Schritt werden die als Dienste ansprechbaren Komponenten zu Anwendungen zusammengestellt (siehe Abschnitt 4).
Im Bereich der Web-Technologien gibt es bei Diensten einen starken Trend hin zur Nutzung von Representational State Transfer (REST) [11] als Architekturparadigma für Anwendungen. REST ermöglicht die flexible, adaptive und robuste Erstellung von Anwendungen durch eine Komposition von Diensten. REST ist ebenfalls Grundlage für sogenannte Microservices, das sind kleinteilige und als Dienste formulierten Komponenten aus komplexen Anwendungen, die zu neuen Anwendungen zusammengeschaltet werden[2]. REST-basierte Systeme folgen bestimmten Einschränkungen; REST fordert, dass

- die Sicht auf die Datenelemente und Komponenten einer Anwendung in der Form von Ressourcen („Dinge“ im weitesten Sinne) bestehe;
- der Datenaustausch zwischen Komponenten durch die Manipulation von Zuständen von Ressourcen über den Austausch von Repräsentationen erfolge; und
- die Zustandsmaschine einer Anwendung exponiert werde in der Form von selbstbeschreibenden Ressourcen, die mit weiteren Ressourcen verlinkt sind (Hypermedia).

Diese Einschränkungen reduzieren die Freiheitsgrade bei der Diensterstellung. Die Einschränkungen erleichtern aber auch die Verwendung von Diensten, da Anwendungen, die auf REST-Ressourcen zugreifen, uniforme Schnittstellen voraussetzen können.
Konkret benötigt man das Folgende, um eine Netzwerkbasierte Anwendung aus REST-basierten Komponenten zu realisieren:

- einen Ansatz zur Benennung von Datenelementen und Komponenten (also von Ressourcen);
- ein Netzwerkprotokoll zum Zugriff auf Ressourcen und der Manipulation deren Zustände, ein gemeinsames Datenmodell und eine übereinstimmende Datenmodellierung; sowie
- die Möglichkeit zum Folgen von Hyperlinks sowie eine Methode zum Modellierung von Zustandsmaschinen, um Interaktionen mit mehreren Schritten zu behandeln.

REST-Architekturen werden meist auf Grundlage der Uniform Resource Identifiers (URIs) und des Hypertext Transfer Protocol (HTTP) umgesetzt. Dabei dienen URIs als Namen für Ressourcen; hauptsächlich verwenden wir URIs mit dem Protokollschema HTTP. Die Beschreibung von Ressourcen und deren Zuständen basieren wir auf das Resource Description Framework (RDF). Auf diese Technologien und ihren Beziehungen gehen wir nun genauer ein.
Zum Austausch von Repräsentationen von Zuständen von Ressourcen nutzen wir HTTP. In HTTP wird der Datenaustausch in Request/Response-Paaren aufgeteilt. Diese Aufteilung führt direkt zu zwei Kategorien von Komponenten: solche, die einkommende Requests verarbeiten (Server), und solche, die Requests abschicken (User Agents bzw. Clients). Auf einen Request antwortet ein Server einem User Agent mit einer Response. In HTTP können Zustände von Ressource sowohl ausgelesen (mithilfe der Operation GET) als auch gesetzt (mithilfe der Operation PUT) werden. Mittels der Operationen PUT oder POST können neue Ressourcen angelegt werden, und mittels der Operation DELETE können bestehende Ressourcen gelöscht werden. So werden in HTTP die sogenannten CRUD-Operationen (create-read-update-delete) umgesetzt, ähnlich zu den Operationen auf Dateien in einem Dateisystem. Mittels der Operation POST können auch beliebige Berechnungen auf entfernten Computern angefordert werden.
Zur Beschreibung von Ressourcen nutzen wir RDF. Dabei dienen URIs in RDF nur als Namen bzw. Bezeichner. In Kombination mit dem HTTP Protokoll können Clients den Zustand einer Ressource über das Netzwerk auslesen. Dabei wird der Zustand der Ressource in RDF beschrieben. Diese Kombination wurde in den Linked-Data-Prinzipien festgehalten. Die Befolgung dieser Prinzipien ermöglichte in den letzten Jahren große Erfolge bei der

[2] http://martinfowler.com/articles/microservices.html

Publikation und Integration von Daten [8]. RDF selbst definiert nur eine abstrakte Syntax [3] und eine einfache Semantik zur Beschreibung von graph-strukturierten Daten [4]. Das strukturelle und semantische Wissen für bestimmte Anwendungsbereiche wird im semantischen Web in Form von Vokabularen verwaltet. Zur konkreten Modellierung der Daten existiert eine Reihe von Ontologiesprachen wie das RDF Schema (RDFS) [5] und die Web Ontology Language (OWL) [1]. Während RDFS die Möglichkeiten zur Verfügung stellt, um Klassenhierarchien zu beschreiben, erweitert OWL diese Möglichkeiten um die Beschreibung komplexerer Beziehungen. Daten in RDF können aufgrund der Graphstruktur relativ einfach kombiniert werden. Daten, welche mit RDFS und OWL ausgezeichnet sind, können mittels logischer Ableitungen integriert und mittels Anfragesprachen konsultiert werden. Linked Data behandelt traditionell nur die GET-Operation von HTTP. Allerdings wurden in der Linked Data Platform (LDP) [6] auch die HTTP-Operationen PUT, POST und DELETE spezifiziert. Weiter wird in LDP zwischen RDF-Ressourcen und Nicht-RDF-Ressourcen (mit binären Repräsentationen) unterschieden, um die Behandlung beider im Kontext von Linked Data zu definieren.

3 Dienstschnittstellen

Im Folgenden gehen wir auf die Schnittstellen zu Diensten ein. ARVIDA-Dienste sind reine Server-Komponenten, welche HTTP-Requests entgegennehmen, mittels der die Clients den Zustand von Ressourcen des Dienstes anfragen bzw. ändern können. Ein Dienst stellt dabei eine Menge, durch URIs benannte, Ressourcen bereit.

Wir beginnen mit einer Beschreibung der entwickelten Vokabularen zum Auszeichnen von Daten, die in der Kommunikation mit Diensten verwendet werden. Danach beschreiben wir den ARVIDA-Präprozessor, mithilfe dessen C/C++-Quelltexte für die Deserialisierung und Serialisierung von Laufzeitobjekten von und zu RDF erzeugt werden kann. In Kombination mit dem in ARVIDA entwickelten RESTSDK für C/C++ lassen sich somit programmatisch Dienste erstellen. Des Weiteren beschreiben wir das Interaktionsmodell zum Zugriff auf die Zustände von Ressourcen und gehen auf verschiedene in VT-Anwendungen eingesetzte Arten von Diensten ein.

3.1 Domänenspezifische Vokabulare

Die gemeinsame Nutzung von definierten Vokabularen [12] ist einer der Grundbausteine zur Datenintegration in verteilten Web-Anwendungen. In ARVIDA wurde eine Reihe spezifischen Vokabularen für die Domäne der Virtuellen Techniken entwickelt (siehe Tabelle 1).

Als grundlegende Vokabulare dienen das Vocabulary of Metrology (VOM) und das Math-Vokabular. VOM setzt den für ARVIDA relevanten Teil des „International Vocabulary of Metrology“ [14] um und definiert alle notwendigen physikalischen Größen, Maßeinheiten und grundlegenden Begriffe zu deren Messung. Die mathematischen Grundkonzepte umfassen neben Skalaren, Vektoren, Tensoren und Matrizen beispielsweise auch Koordinatensysteme. Das Spatial-Vokabular nutzt diese beiden Vokabulare und definiert verschiedene räumliche Beziehungen zwischen Koordinatensystemen, die sowohl für die Verortung von virtuellen, als auch realen Objekten genutzt werden können.

Zu diesem Zweck wurde das Konzept eines Spatial Relationship-Graphen [10] umgesetzt, welches eine einfache Darstellung solcher Probleme ermöglicht. Ein typisches Anwendungsbeispiel dafür ist das Tracking von Objekten und Personen. Für personenbezogene Tracking-Anwendungen existieren wiederum spezielle Vokabulare, welche ein feingranulares Skelett definieren.

Für Visualisierungssysteme existieren mehrere Vokabulare, durch welche die einzelnen Bestandteile des gesamten Visualisierungssystems, des 3D-Renderers und der Anzeige des Bildes abgebildet werden.

Für Szenengraphen [22] stellen wiederum die Spatial Relationship-Graphen die Basis für eine Spezifikation räumlicher Beziehungen. Nicht-räumliche Aspekte eines Szenegraphen werden durch spezielle Vokabulare expliziert.

3.2 Der ARVIDA-Präprozessor

Zur Vereinfachung der Implementierung von ARVIDA-konformen VT-Komponenten steht mit dem ARVIDA-Präprozessor (ARVIDA-PP) ein Verfahren zur minimalintrusiven Annotation von Quelltexten in C/C++ sowie, aufbauend auf den Annotationen, ein Quelltext-Generator für die Deserialisierung und Serialisierung von Objekten von und zu RDF zur Verfügung. Eine Verwendung des ARVIDA-PP wird durch ein Web-Frontend mit integriertem Quelltextbetrachter sowie einer REST-Schnittstelle erleichtert.

Einfache und lesbare Annotationen erlauben die semantische Auszeichnung von Quelltexten unter Nutzung der ARVIDA-spezifischen Domänenvokabulare. Mit Hilfe des Clang-Compiler für C++ werden die Annotationen durch den ARVIDA-PP extrahiert und in geeignete Deserialisierungs- und Serialisierungsroutinen transformiert. Hierbei werden die weit verbreiteten RDF-Bibliotheken Redland und Serd/Sord unterstützt. Um weitere RDF-Bibliotheken einbinden zu können, nutzt der ARVIDA-PP zur Quelltexterzeugung eine Template-Engine. Somit können eigene Templates erstellt oder existierende Templates auf die eigenen Bedürfnisse angepasst werden. Die erzeugten Routinen können nun einem eingebetteten HTTP-Server zur Verarbeitung eintreffender Requests zur Verfügung gestellt und die Daten einer VT-Funktionskomponente als RDF ausgetauscht werden.

3.3 Das ARVIDA-RESTSDK

Das in der Architektur verwendete Request/Response-Interaktionparadigma und das semantisch mächtige Datenmodell basieren auf bewährten und spezifizierten Web-Standards, die eine breite Unterstützung vorhandener Entwicklungswerkzeuge sowie eine einfache Integration in bestehende Netzwerke gewährleisten. Für die Entwicklung von Komponenten besteht im Bereich Java bereits eine gu-

Tabelle 1 ARVIDA Vokabulare, verfügbar unter `http://vocab.arvida.de/`.

URI	Inhalt
`2015/06/vom`	Physikalische Größen, Maßeinheiten, grundlegende Metrologie-Begriffe
`2015/06/maths`	Mathematischen Grundkonzepte
`2015/06/spatial`	Projektionen, räumliche Beziehungen und Grundvokabular für Spatial Relationship Graphs
`2015/06/visService`	Grundlegende Eigenschaften von Visualisierungssystemen und Renderern
`2015/06/camera`	Beschreibung von realen und virtuellen Kameras
`2015/06/displayservice`	Grundlegende Eigenschaften von Displays und Viewports
`2015/06/scene`	Räumliche Verortung von Szenegraphen, Varianten-Management
`2015/06/scenegraph`	Generische Konzepte zur Beschreibung von Szenegraphen
`2015/06/skeleton`	Generische Konzepte zur Beschreibung von Skeletten zur Animation
`2015/06/skeleton-H-Anim`	Spezialisierung für H-Anim Skelette
`2015/06/ubitrack`	Vokabular für das Tracking Framework Ubitrack
`2015/06/opencv`	Vokabular für das AR Framework OpenCV
`2016/02/boundingvolume`	Generische Beschreibung von Bounding Volumes
`2016/04/workflow`	Beschreibung von Workflows, die in Trainings vermittelt werden

te Unterstützung, z.B. Jena im Bereich RDF oder Jersey im Bereich REST. In weiteren Programmiersprachen ist dieser Unterstützung zum Teil ebenfalls gut (z.B. Python oder Ruby) oder, wie im Falle von C/C++, weniger gut ausgeprägt.

Zur Erleichterung der Entwicklung unter C/C++, welche eine der vorherrschenden Programmiersprachen im Bereich VT darstellt, wurde im Rahmen des ARVIDA-Projektes das RESTSDK entwickelt. Es bietet integrierte Unterstützung für die Bereitstellung von Linked Data REST-Schnittstellen mittels Request-Handler, optional in Kombination mit dem ARVIDA-Präprozessor. Außerdem bietet das RESTSDK Unterstützung für den Zugriff auf derartige Schnittstellen.

3.4 Interaktionsmodell für Dienste

Die ARVIDA-Dienstschnittstellen bauen auf standardisierten Web-Technologien auf. Allerdings sollen die Dienste, im Gegensatz zum Web, in hochdynamischen VT-Umgebungen eingesetzt werden. In VT-Anwendungen können wir die angebundenen Komponenten in drei Gruppen einteilen, basierend auf den Änderungsraten der Ressourcenzustände:

Marginale Änderungsrate: Eine niedrige, fast statische, Änderungsrate, beispielsweise externe Daten aus Dateisystemen, Datenbanken oder dem Web.

Regelmäßige Änderungsrate: Eine hohe Änderungsrate mit annähernd gleicher Frequenz, beispielsweise das Lesen von Kamerabildern, Skelettpunkten oder das Schreiben von Bildern in Visualisierungssysteme.

Unregelmäßige Änderungsrate: Eine Änderungsrate ohne oder mit sehr schwankender Frequenz, beispielsweise Nutzereingaben.

In den ersten beiden Kategorien nutzen wir regelmäßige Zugriffe („pulling“ oder „polling“), um auf den Zustand der Ressourcen zuzugreifen. Dabei kennt die Datensenke (oft die Anwendung) die URI der Quelle, d.h. der Dienst braucht keine URIs der Anwendung zu kennen. In den Demonstratoren erzielen wir so Zugriffsraten von 30 Hz (33 ms pro Zyklus). Für die letzte Kategorie nutzen wir ein Event-Modell („push“). Dabei muss die Quelle die URI der Datensenke (meist die VT-Anwendung) kennen, d.h. der Komponente muss die URI der Senke mitgeteilt werden. Außerdem muss die Komponente, die Events bereitstellt, die Möglichkeit bekommen, HTTP-Requests abzusetzen.

4 VT-Anwendungen

Basierend auf der vorgestellten Dienstschnittstelle können Anwendungen gestaltet werden, die aus verschiedenen Diensten bestehen, welche sich auf unterschiedlichen physikalischen Systemen befinden und von unterschiedlichen Herstellern stammen. Jeder Dienst kapselt dabei spezifische Funktionalität sowie Daten und stellt diese über eine konforme Schnittstelle im Netzwerk zur Verfügung. Des Weiteren gibt es (aktive) Komponenten, welche mit (passiven) Diensten interagieren, d.h. Requests absetzen und Responses der Dienste verarbeiten. Im folgenden Text ist mit einer Anwendung immer eine aktive Komponente im Zusammenspiel mit Diensten gemeint.

Auf Basis der uniformen Dienstschnittstellen können verschiedenste Dienste pragmatisch zu Anwendungen zusammengestellt werden, z.B. mittels prozeduraler Programmiersprachen wie C/C++ oder Java. Dabei wird der Weltzustand in Objekten der Programmiersprache dargestellt und der aktuelle Weltzustand im Quelltext des Programms verwaltet. Die Anwendungslogik wird mittels Kontrollstrukturen der Programmiersprache umgesetzt. Es wird hierbei jedoch der Zugriff auf entsprechende Diensten bzw. deren Schnittstellendefinitionen zur Entwicklungszeit benötigt, um diese – unter Berücksichtigung der jeweils verwendeten Programmiersprachen und Technologien – individuell anzusprechen.

In einer weiteren Ausbaustufe kann die Anwendungslogik regelbasiert spezifiziert werden. Im Folgenden skizieren wir den Ansatz einer solchen regelbasierten Komposition von VT-Anwendungen in ARVIDA.

4.1 Regelbasierte Anwendungserstellung

Die uniformen Dienstschnittstellen, das Request/Response Kommunikationsparadigma und das semantisch mächtige Datenmodell ebenen den Weg für eine deklarative Komposition der Dienste, welche von den konkret für die Entwicklung der Diensten verwendeten Programmiersprachen und Technologien abstrahiert.

Die hierbei verwendete Kompositionssprache ermöglicht z.B. die Definition von Datentransformationen, Entscheidungen basierend auf dem aktuellen Weltzustand, oder auszuführenden Interaktionen mit Diensten. Sie bildet zum einen, durch ihr zugrundeliegendes formales Modell, die Basis für Autorenwerkzeuge, welche dem Fachanwender eine vereinfachte visuelle Oberfläche bieten können.

In einfacher Form der Komposition von VT-Anwendungen wird dabei der Weltzustand als Vereinigung der Zustände der relevanten Ressourcen in RDF gehalten und periodisch mittels HTTP GET Requests aktualisiert. Anwendungslogik wird durch Wenn-Dann Regeln umgesetzt. Diese Regeln setzen dabei den Zustand von Ressourcen (via PUT/-POST/DELETE Operationen) basierend auf dem aktuellen Weltzustand der Anwendung.

Eine so deklarierte Komposition von Komponenten, auch Programm genannt, kann zum durch eine passende Ausführungsumgebung zentral koordiniert und zu einer laufenden Anwendung verschaltet werden, ohne den Zugriff auf Dienste bzw. deren Schnittstellen zur Entwicklungszeit zu benötigen.

Während die zentrale Koordination mit einer aktiven Komponente für einige Szenarien ein valides Schema darstellt, können in anderen Szenarien die Anforderungen, z.B. an Redundanz, Latenz, oder Bandbreite, nicht mehr erfüllt werden. Darüber hinaus sind diese Anforderungen zur Entwicklungszeit der Komponenten nicht zwingen bekannt. In diesem Fall sind Ansätze erforderlich, die eine Komposition mit dezentraler Koordination der Komponenten ermöglichen.

4.2 Regelsprache und Ausführungsumgebung

Zur Unterstützung der deklarativen Komposition und Koordination wurde im Rahmen des ARVIDA-Projektes die Entwicklung der Linked Data-Fu Regelsprache [23] zur Komposition und Integration von Linked Data REST-Ressourcen vorangetrieben. Die auf der Syntax von Notation3 (N3) [7] basierende Regelsprache dient der deklarativen Definition von Programmen, die die anwendungsspezifische kontrollierende Logik zur Integration von verschiedenen, den gemeinsamen Paradigmen folgenden, Komponenten zu einer verteilten Anwendung enthalten.

Die Regelsprache erlaubt die Ableitung von Wissen durch Unterstützung von Schlussfolgerungen über Daten in RDF sowie RDFS und Teilen von OWL, die Interaktion mit Linked Data REST-Ressourcen durch Unterstützung der HTTP-Methoden GET, PUT, POST, und DELETE, sowie mathematische Berechnungen, unterstützt durch eingebaute Funktionen. Darüber hinaus wird die Bearbeitung von Abfragen über vorhandenes Wissen unterstützt.

Die Ausführungsumgebung für Programme der Regelsprache wurde im Rahmen des ARVIDA-Projektes weiterentwickelt. Diese übernimmt die Evaluation der Regeln und Abfragen, sowie die Ausführung der HTTP-basierten Interaktion mit Linked Data REST-Ressourcen. Außerdem ist die Ausführungsumgebung um eine REST-Schnittstelle erweitert worden. Damit wird die Nutzung des Interpreters als eigenständige Komponente sowie der Einsatz des Interpreters als intelligente Zugriffsschicht in Komponenten ermöglicht. Die Schnittstelle erlaubt die Erstellung und Modifikation von Interpreter-Instanzen, von Regelprogrammen, sowie von Linked Data REST-Ressourcen basierend auf Abfragen in SPARQL Protocol and RDF Query Language (SPARQL) [2] Syntax.

5 Demonstratoren

Im Rahmen der Entwicklung, der Evaluation sowie zur Veranschaulichung der Referenzarchitektur wurden verschiedene Proof-of-Concept-Implementationen, ein Architektur-Demonstrator sowie Teile eines industrielle Prototypen entwickelt.

Die grundsätzliche Eignung der aus dem Web-Umfeld stammenden Paradigmen und Technologien für eine Architektur zur Komposition von VT-Anwendungen wurden in zwei Proof-of-Concept-Implementationen [16, 15] evaluiert. Durch die Einbindung zusätzlicher, parallel aus dem Web eingebundener, Daten wurde das Integrationspotential gezeigt. In der simplen verteilten Anwendung werden die Skelett-Koordinaten der vor einem handelsüblichen Tiefensensor befindlichen Personen berechnet, über ein Linked Data REST-Interface in einem Netzwerk bereitgestellt, von einer Ausführungsumgebung mit Hilfe eines Regelprogramms an eine Visualisierung weitergeleitet und dort mit zusätzlichen Informationen aus dem Web dargestellt.

Der Demonstrator der Referenzarchitektur fokussiert sich in seinem Szenario auf den Arbeitsprozess eines Werkers, der diesen Prozess in einer überlagerten Realität an virtuellen Maschinen durchführt, um so die Auswirkung verschiedener Position dieser Maschinen auf den Prozess nachvollziehen zu können. Die Komponenten stellen verschiedenste Funktionalität zur Verfügung, unter anderem die überlagerte Realität, das Tracking zu überlagernder Objekte, die Überwachung der Schritte des Arbeitsprozesses, sowie eine Verwaltung für Arbeitsaufträge. Die Kommunikation zwischen den Komponenten respektiert die REST-Prinzipien und für die ausgetauschten Daten werden die im Projekt entwickelten Vokabularien verwendet. Die Überwachung des Werker-Workflows und damit verbundene Koordination von Komponenten ist in der Regelsprache deklariert und wird von der Ausführungsumgebung kontrolliert.

Im industriellen Umfeld sind, neben weiteren Anwendungsszenarien, unter anderem Teile eines ersten Prototypen für das Training von Produktionsmitarbeitern hinsichtlich der Abläufe bei Montageumfängen umgesetzt worden [9]. Die Mitarbeiter werden dabei in einem sogenannten „Profiraum“ an Aufbauten geschult, die dem realen Produktionsumfeld nachempfunden sind. In diesem Prototypen werden den Mitarbeitern mithilfe von Datenbril-

len Lerninhalte in Form einer erweiterten Realität präsentiert. Das Trainingssystem wurde unter Berücksichtigung der Referenzarchitektur und mit Hilfe der beschriebenen Technologien modelliert und implementiert.

6 Fazit

Durch Entwurf und Umsetzung domänenspezifischer Vokabulare, eines Verfahrens zur Abbildung von Laufzeitobjekten auf RDF-Graphen und einer integrierten Unterstützung für die Erstellung von REST-Schnittstellen gelingt im Projekt ARVIDA die Bereitstellung von VT-Funktionskomponenten in Form von semantisch beschriebenen VT-Diensten.

Diese VT-Dienste können mit Hilfe der regelbasierten ARVIDA-Kompositionssprache zu neuen VT-Anwendungen zusammengestellt werden. Die Fokussierung auf wenige Interaktionsprimitive der uniformen REST-Schnittstellen vereinfacht dabei die Komposition. Darüber hinaus erlaubt die ARVIDA-Regelsprache die gleichzeitige Verwendung von Techniken der Datenintegration. Die im Rahmen des Projekts weiterentwickelte Ausführungsumgebung ist für die Umsetzung von VT-Anwendungen im Kontext von Industrie 4.0 ausreichend performant. Unsere Industriepartner setzen die entwickelten Technologien in Prototypen bereits ein.

Heterogene Systeme, die zu komplexen Anwendungen komponiert werden sollen, sind die Bausteine des Internet der Dinge und von Industrie 4.0. ARVIDA deckt dabei einen Teil der in der „Referenzarchitektur für Industrie 4.0“ (RAMI 4.0) identifizierten Herausforderungen ab. Auch außerhalb von VT-Anwendungen, welche recht hohe Anforderungen an Durchsatz und Latenz stellen, sehen wir Einsatzmöglichkeiten. Wir hoffen daher, dass Web-Technologien, deren Einsatzfähigkeit wir im Projekt ARVIDA demonstriert haben, auch in weiteren Bereichen zum Einsatz kommen.

7 Literatur

[1] OWL 2 Web Ontology Language Document Overview (Second Edition), Nov. 2012. `http://www.w3.org/TR/owl2-overview/`.

[2] SPARQL 1.1 Query Language, Mar. 2013. `http://www.w3.org/TR/sparql11-query/`.

[3] RDF 1.1 Concepts and Abstract Syntax, Feb. 2014. `http://www.w3.org/TR/rdf11-concepts/`.

[4] RDF 1.1 Semantics, Feb. 2014. `http://www.w3.org/TR/rdf-mt/`.

[5] RDF Schema 1.1, Feb. 2014. `http://www.w3.org/TR/rdf-schema/`.

[6] Linked Data Platform 1.0, Feb. 2015. `http://www.w3.org/TR/ldp/`.

[7] T. Berners-Lee and D. Connolly. Notation3 (N3): A readable RDF syntax, 2011. `https://www.w3.org/TeamSubmission/n3/`.

[8] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[9] S. Brauns, D. Koriath, T. Käfer, and A. Harth. Individualisiertes Gruppentraining mit Datenbrillen für die Produktion. In *Tagungsband zum Workshop „Arbeitsplatz der Zukunft“ der 46. Jahrestagung der Gesellschaft für Informatik (INFORMATIK)*. GI, 2016. akzeptiert zur Publikation.

[10] F. Echtler, M. Huber, D. Pustka, P. Keitler, and G. Klinker. Splitting the Scene Graph. 2008.

[11] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, USA, 2000.

[12] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *Int'l journal of human-computer studies*, 43(5):907–928, 1995.

[13] P. Hitzler, M. Krötzsch, S. Rudolph, and Y. Sure. *Semantic Web: Grundlagen*. Springer-Verlag, 2007.

[14] JCGM 200, editor. *International vocabulary of metrology – Basic and general concepts and associated terms (VIM)*. 3rd edition edition, 2012.

[15] F. L. Keppmann, T. Käfer, S. Stadtmüller, R. Schubotz, and A. Harth. High performance linked data processing for virtual reality environments. In *Proceedings of the Posters & Demonstrations Track of the 13th International Semantic Web Conference, ISWC*, 2014.

[16] F. L. Keppmann, T. Käfer, S. Stadtmüller, R. Schubotz, and A. Harth. Integrating highly dynamic restful linked data apis in a virtual reality environment. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, ISMAR*, 2014.

[17] S. Lukosch, M. Billinghurst, L. Alem, and K. Kiyokawa. Collaboration in augmented reality. *Computer Supported Cooperative Work (CSCW)*, 24(6):515–525, 2015.

[18] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. Augmented reality: a class of displays on the reality-virtuality continuum, 1995.

[19] A. Roth and D. Siepmann. *Industrie 4.0 – Ausblick*, pages 247–260. Berlin, Heidelberg, 2016.

[20] W. Schreiber and P. Zimmermann, editors. *Virtuelle Techniken im industriellen Umfeld: Das AVILUS-Projekt-Technologien und Anwendungen*. Springer, 2011.

[21] D. Siepmann and N. Graef. *Industrie 4.0 – Grundlagen und Gesamtzusammenhang*, pages 17–82. Springer, 2016.

[22] H. Sowizral. Scene graphs in the new millennium. *IEEE Computer Graphics and Applications*, 20(1):56–57, Jan 2000.

[23] S. Stadtmüller, S. Speiser, A. Harth, and R. Studer. Data-Fu: A Language and an Interpreter for Interaction with Read/Write Linked Data. In *Proceedings of the International World Wide Web Conference*, 2013.