Raphael Zender (Hrsg.): Proceedings of DeLFI Workshops 2016
co-located with 14th e-Learning Conference of the German Computer Society (DeLFI 2016)
Potsdam, Germany, September 11, 2016 174

APPsist Statusbericht: Realisierung einer Plattform für Assistenz- und Wissensdienste für die Industrie 4.0

Carsten Ullrich[1], Matthias Aust[2], Michael Dietrich,[1] Nico Herbig[3], Christoph Igel[1], Niklas Kreggenfeld[4], Christopher Prinz[4], Frederic Raber[3], Simon Schwantzer[5] und Frank Sulzmann [2]

Abstract: Im Projekt APPsist wird eine Architektur für Assistenz- und Wissensdienste zur Unterstützung der Beschäftigten in der Industrie 4.0 entwickelt, bestehend aus Basisdiensten und intelligent-adaptiven Diensten, die angepasst an den Kontext und den Benutzer die Unterstützung realisieren. Das Projekt befindet sich nahe dem Ende der Laufzeit, und dieser Beitrag beschreibt das finale System und gibt eine kritische Betrachtung der zu Projektbeginn getroffenen technischen Entscheidungen.

Keywords: Arbeitsplatzintegriertes Lernen, Assistenzdienste, Wissensdienste, Adaptivität

1 Assistenz- und Wissensdienste für die Industrie 4.0

Der Begriff „Industrie 4.0“ beschreibt im Wesentlichen die technische Integration von cyber-physischen Systemen (CPS) in Produktion und Logistik [KWH13]. CPS greifen mittels Sensoren auf Daten der physikalischen Welt zu und wirken auf diese mittels Aktoren ein. Damit wird die physikalische Welt mit der virtuellen Welt zu einem Internet der Dinge und Dienste verknüpft. Die dadurch entstehenden cyber-physischen Produktionssysteme (CPPS) stellen aufgrund ihrer Komplexität Herausforderungen für die Beschäftigen in der Industrie 4.0 dar.

Das Ziel des Verbundprojektes APPsist ist die Entwicklung einer neuen Generation mobiler, kontextsensitiver und intelligent-adaptiver Assistenz- und Wissensdienste für die Industrie 4.0. Als Assistenz- und Wissensdienste werden Softwarekomponenten bezeichnet, die spezifische Arten von Unterstützung leisten: Assistenzdienste unterstützen beim Beheben eines aufgetretenen Problems, während Wissensdienste die Wissensvermittlung unterstützen, d.h. das Erreichen von individuellen mittel- und langfristigen Entwicklungszielen. In APPsist wird eine Dienstearchitektur entwickelt, deren Funktionalität sich aus dem Zusammenspiel der Dienste ergibt. Jeder der Dienste realisiert dabei eine spezifische, eigenständige Funktionalität und stellt diese anderen Diensten zur Verfügung. APPsist orientiert sich an den realen Anforderungen der produzierenden Industrie, repräsentiert durch

[1] DFKI GmbH, Center for Learning Technology, Alt-Moabit 91c, 10559 Berlin
[2] Fraunhofer IAO, Virtual Environments, Nobelstr. 12, 70569 Stuttgart
[3] DFKI GmbH, Innovative Retail Laboratory, 66123 Saarbrücken
[4] Ruhr-Universität Bochum, Lehrstuhl für Produktionssysteme, 44801 Bochum
[5] IMC AG, Innovation Labs, Scheer Tower Uni-Campus Nord, 66123 Saarbrücken

drei eingebundene Unternehmen und eine Lernfabrik. Die unterstützten Arbeitsprozesse umfassen Inbetriebnahme von Anlagen, Fehlerbehebung sowie Wartung und Instandhaltung [Ul16]. Dieser Artikel aktualisiert den ersten Bericht zum Projektstand [Ul15] und dient zur Beschreibung neuer Funktionalitäten, gesammelter Erfahrungen sowie der kritischen Betrachtung getroffener Entscheidungen.

2 Vergleich mit thematisch ähnlichen Projekten

Das Potential von Diensten, Bildungsprozesse am Arbeitsplatz zu unterstützen, wird aktuell aus einer Vielzahl von Perspektiven erforscht. So wurden bisher primär Web 2.0 Technologien für das Themenfeld Wissensvermittlung verwendet, z.B. Erstellung und Austausch von Inhalten in der Instandhaltung (BMBF Projekt DiLi) und Unterstützung von Mechanikern bei Wartung und Reparatur (BMBF Projekt MOLEM). Ebenfalls gibt es Untersuchungen zur Verknüpfung von automatischer Kontexterfassung und sozialen Netzwerken für das betriebliche Wissensmanagement (BMBF Projekt AmbiWise) und für den Transfer von Erfahrungswissen (BMBF Projekt PLUTO, [BR15]). Eine Reihe von EU Projekten widmen sich der Frage, wie Daten aus fabrikweiten Sensornetzwerke verwendet werden können um die Informationsflüsse so zu steuern, dass kognitive Überlastung der Mitarbeiter vermieden wird (EU FP7 Projekt Sense& React), oder wie diese auf modernen Endgeräten dargestellt werden können, um Arbeitszufriedenheit zu erhöhen (EU H2020 Projekt SatisFactory). Der Anspruch, eine allgemein einsetzbare Architektur für Assistenz- und Wissensdienste zu entwickeln, ist nach wie vor ein Alleinstellungsmerkmal von APPsist. Zu nennen ist im Bereich intelligent-adaptiver Systeme noch DigiLern-Pro [Fr15], in dem Beschäftigte unterstützt durch KI Arbeitsprozesse aufnehmen.

3 Stand der Umsetzung

3.1 Vorgehensmodell zur Prozessaufnahme

Zur Umsetzung von Assistenzprozessen in den Unternehmensbereichen müssen diese systematisch aufgenommen und aufbereitet werden. Folgendes Vorgehensmodell hat sich in APPsist bewährt: Zuerst werden in einem Experten-Workshop alle für die Durchführung eines Prozesses notwendigen Schritte aufgenommen. Die aufgenommene Struktur entspricht in der Regel zunächst der bisherigen Vorgehensweise, ist aber nicht zwangsläufig optimal hinsichtlich Effizienz und Prozesslogik. Daher muss der Prozess in einem zweiten Schritt in der Expertenrunde optimiert werden, um somit einen Best-Practice-Ansatz zu generieren. Dabei muss vor allem die angestrebte Zielgruppe betrachtet werden, da sich die Prozessmodellierung in der Struktur und Granularität dementsprechend signifikant unterscheiden kann. Mit der Optimierung wird für den Assistenzprozess eine definierte und reproduzierbare Arbeitsmethode festgelegt. Um diese für APPsist nutzbar zu machen, müssen im Folgenden Prozessschritte annotiert werden mit relevanten Daten aus der Maschinensteuerung und aus ME- und ERP-Systemen, die den Prozess entweder auslösen,

oder die für die Interaktion des Systems mit der Maschine oder Anlage relevant sind. Weiterhin müssen die Inhalte (Texte, Fotos, Modelle, Videos und Augmented Reality Inhalte) aufgenommen werden, die der Nutzer im Rahmen der Assistenz angezeigt bekommen soll. Bevor der Prozess im realen Umfeld eingesetzt werden kann, ist es weiterhin unabdingbar, diesen in einer Evaluationsrunde mit Experten und auch Shopfloormitarbeitern zu testen, um letzte Optimierungen an den gezeigten Inhalten vorzunehmen.

3.2 Architektur/Framework

Die APPsist-Plattform vereint viele unterschiedliche Anforderungen in einem kohärenten System. Daher wurden für die Entwicklung den Prinzipien einer Microservice-Architektur verfolgt und die Funktionen in einzelnen Diensten gekapselt, welche dann in der Plattform orchestriert werden. APPsist umfasst derzeit 18 Dienste aus den Bereichen Assistenz, Weiterbildung, Adaption, Benutzer- und Maschinenschnittstellen. Als Integrationstechnologie wird das Vert.x-Framework [Ve] in Version 2.1 verwendet. Profiliert hat sich das Framework bei der Realisierung der APPsist-Plattform insbesondere bei der Orchestrierung der Dienste über das integrierte Modulsystem und der den Diensten zur Verfügung gestellten API für Server- und Clientimplementierungen. Als Laufzeitumgebung läuft Vert.x stabil und konnte problemlos in die IT-Landschaft der Pilotbereiche integriert werden. Das Framework bietet darüber hinaus eine Sock.js-Bridge zur plattformübergreifenden, bidirektionalen Kommunikation an, welche bei der Realisierung der Client-Applikation verwendet wurde. Hierbei musste jedoch zusätzlicher Aufwand für die Stabilisierung der Anbindung investiert werden, da per se keine Mechanismen zur automatischen Wiederaufnahme einer Verbindung bereitgestellt werden.

Während die Dienste von den Entwicklungspartnern eigenverantwortlich entwickelt werden, gibt es dennoch eine Reihe von Fragen, welche dienstübergreifend geklärt werden mussten: a) Welche Funktionen erfüllt welcher Dienst? b) Wer ist für die Entwicklung des Dienstes zuständig? c) Welche Kommunikationswege gibt es zwischen den Diensten? Die Erfahrung hat gezeigt, dass diese Fragen nicht nur initial adressiert werden müssen, sondern auch entwicklungsbegleitend, insbesondere im Kontext eines Forschungsprojekts: Viele Funktionen konkretisieren sich erst im Rahmen der Entwicklung und es ergeben sich darüber hinaus regelmäßig neue Anforderungen an die Plattform.

Typisch für eine Microservice-Architektur sind die besonderen Anforderungen an Analysierbarkeit des Systemverhaltens, da die Entwicklung der Dienste weitestgehend getrennt stattfindet. Die Laufzeit- und Fehleranalyse des Gesamtsystems ist für die Entwickler erschwert, da die Plattform i.d.R. nicht vollständig lokal ausgeführt und Fehlverhalten erst durch die Kombination mehrerer Dienste ersichtlich wird. Daher erfolgt das Protokollieren von Systemereignissen strukturiert in einer (dienstübergreifenden) Datenbank, der Ausführungstatus der einzelnen Dienste wird erfasst und eine Weboberfläche bereitgestellt, die Entwicklern Zugang zu den Laufzeitinformationen der Plattform ermöglicht.

Eine besondere Herausforderung stellt das Prinzip der Isolation auch in anderer Hinsicht dar: Jeder Dienst soll unabhängig von den anderen Diensten laufen, so dass Fehler einzel-

ner Dienste nicht das ganze System in einen Fehlerzustand bringen. Technisch wird diese Trennung seitens des Frameworks dahingehend unterstützt, dass die einzelnen Dienste als Module in separaten Classloadern ausgeführt werden und separat initiiert und verwaltet werden. Technische Sicherungsmechanismen schützen jedoch nicht vor logischen Abhängigkeiten zwischen den Diensten. So ist z.B. der Assistenzdienst abhängig von dem Dienst, welcher für das für die Adaption der anzuzeigenden Inhalte zuständig sind. Die Betrachtung möglicher Störungen ist daher essentieller Bestandteil der Modellierung der Kommunikation: Es muss geklärt werden wie darauf reagiert werden soll wenn ein Dienst a) nicht reagiert (Kompensation), b) über einen Fehler benachrichtigt (Fehlerbehandlung), oder c) unerwartete Daten zurückliefert (Validierung).

Eine Eigenschaft der APPsist-Architektur, welche in vielen Beispielen einer Microservice-Architektur so nicht zu finden ist, liegt in der Natur eines ereignisbasierten Systems: Es reagiert nicht nur auf Benutzereingaben und -anfragen, sondern auch auf Ereignisse des Benutzerkontexts, z.B. der bedienten Maschine. Dies sollte bei der Planung der Kommunikation besondere Beachtung finden, da die Gefahr besteht, dass ein Ereignis mehrere Aktionen anstoßen kann. Dies ist dann problematisch wenn mehrere Aktionen die gleichen Parameter des Systemzustands beeinflussen. Mögliche Lösungen sind die explizite, dienstübergreifende Spezifikation der durchzuführenden Aktionen und deren Auswirkungen, sowie die Priorisierung möglicher Zustandsänderungen.

3.3 Intelligent-adaptive Dienste

Als eindeutiges und festgelegtes Vokabular für die Kommunikation der Dienste und als Grundlage der intelligenten Entscheidungsprozesse der adaptiven Dienste dient in APPsist ein Modell der Domäne „Produktion“ (repräsentiert in OWL). Es wurde eine allgemeine Ontologie erstellt, die dann für die jeweiligen Anwendungspartner spezialisiert wurde. Die Adaptionsregeln, die die Informationen der Ontologie verwenden (gespeichert in einem Tripple-Store), um die Unterstützung der Beschäftigten zu realisieren, basieren auf der allgemeine Ontologie und sind somit auf alle Anwendungspartner anwendbar. Es lassen sich im aktuellen APPsist-System folgende vier Unterstützungsarten die durch Regeln realisiert werden unterscheiden: *Haupttätigkeit-Assistenz*: Wenn Mitarbeiter in „Haupttätigkeit“ und fordert Assistenz an, dann bestimme Maßnahmen, die relevant sind für aktuelle Station und Maschinenzustand. *Nebentätigkeit-Assistenz*: Wenn Mitarbeiter in „Nebentätigkeit Lernzeit“ und fordert Assistenz an, dann bestimme Maßnahmen, die relevant sind für langfristige Entwicklungsperspektive. *Haupttätigkeit-Inhalte*: Wenn Mitarbeiter in „Haupttätigkeit“ und Anforderung von Inhalten, dann bestimme Inhalte, die relevant sind für aktuelle Station und Maschinenzustand. *Nebentätigkeit-Inhalte*: Wenn Mitarbeiter in „Nebentätigkeit Lernzeit“ und Anforderung von Inhalten, dann bestimme Inhalte, die relevant sind für langfristige Entwicklungsperspektive. Die Regeln legen fest, welche Maßnahmen und Inhalte in der konkreten Situation *relevant* sind. Diese ziehen beispielsweise die Arbeitsplatzgruppe des Beschäftigten in Betracht, Aufgaben von Stellen und damit verbundene Maßnahmen und weiteres. Implementiert sind sie in einer Kombination aus Java-Code und SPAQRL Anfragen an den Tripple-Store.

3.4 Einbindung von Augmented Reality

Der Großteil der APPsist-Oberfläche wurde mit Webtechnologien umgesetzt. Dadurch funktioniert sie im Browser und damit auf sehr vielen Endgeräten. Eine Ausnahme dazu stellen die Augmented Reality (AR)-Funktionalitäten dar, für die spezielle Komponenten notwendig sind. Diese sind im APPsist-System Vuforia als AR-Plattform und Unity als Grafik- und App-Entwicklungs-Engine. Im Gegensatz zum ursprünglichen und in ersten Prototypen getesteten Konzept, das gesamte APPsist-Frontend in einer Unity-App laufen zu lassen, sind AR-Funktionalitäten und Inhalte, die im Browser dargestellt werden können, ab der aktuellen APPsist-Version konsequent getrennt. D.h. der Nutzer schaltet zwischen AR-App (Unity) und beispielsweise Assistenzinhalten (Browser) hin und her – allerdings nicht explizit sondern per Button in der GUI, sodass diese Trennung im Nutzungserlebnis nicht auffällt. Die Trennung verbessert die Performanz und vereinfacht das Deployment für verschiedene Plattformen und Endgeräte – je nachdem ob AR mit dem Endgerät möglich ist, oder nicht. Bis jetzt wird AR v.a. eingesetzt, um dem Nutzer die Orientierung an komplexen Maschinen und Anlagen zu erleichtern, indem bestimmte Maschinenteile im Kamerabild hervorgehoben werden; weitere AR-Szenarien sind geplant.

3.5 Maschinenanbindung

Die Anbindung an die Produktionsanlagen wurde durch eine REST API realisiert: Der Server ist als Dienst in APPsist implementiert und benötigt erst zur Laufzeit Informationen über die Daten der angeschlossenen Maschinen. Dadurch ist keine serverseitige Anpassung nötig, falls neue Maschinentypen unterstützt oder bestehende angepasst werden müssen. Um eine neue Maschine an das APPsist-System anzuschließen muss ein REST-Klient entwickelt werden, der die Daten der Maschinensteuerung ausliest und aufbereitet an den Server weitergibt. Zu Beginn sendet der Klient an den Server eine *Schemanachricht* die die verfügbaren Daten sowie eine Kurzbeschreibung der Daten enthält. Im Anschluss sendet der Klient periodisch *Datennachrichten*, welche die tatsächlichen Maschinendaten entsprechend dem vorangegangenen Schema enthalten. Durch diese Trennung kann der Server bereits im Vorfeld die Visualisierung der Maschinendaten aufbauen. Innerhalb des Projektes wurde bereits eine Anbindung an SPS-Steuerungen, OPC-Server sowie Anlagen von Mitsubishi und Siemens realisiert. Die Kommunikation kann dabei über die Formate JSON, XML, EXI oder Plaintext erfolgen. Die oben beschriebene Schnittstelle wurde um eine Funktion erweitert, die es ermöglicht Verbindungsabbrüche zu Maschinen frühzeitig zu erkennen und dies in der Oberfläche entsprechend wiederzugeben.

4 Usability Evaluation

Das APPsist-Projekt verfolgt den Ansatz eines benutzerzentrierten Entwurfsprozesses und so wurden potentielle Nutzer als wichtigste Evaluationsinstanz für die Gebrauchstauglichkeit der Benutzungsoberflächen bereits mehrere Male im Laufe den Projekts befragt bzw.

die GUI mit ihrer Hilfe getestet. Ein größer angelegter Usability-Test wird im Folgenden beschrieben. Weitere derartige Tests werden bis zum Ende des Projekts folgen.

Am aktuellen Usability-Test nahmen von jedem der drei Anwendungspartner je sechs Testpersonen teil. Als Endgeräte wurden verschiedene Tablets (Microsoft Surface 4 Pro, LG Nexus 5, Fujitsu V535 Industrial) verwendet sowie ein 22"-Touchscreen an einem Windows-7-Desktop-Rechner. Der Ablauf des Tests war jeweils wie folgt:

1. Briefing für die Testpersonen mit Datenschutzvereinbarung
2. Vorabfragebogen: Ein demografischer Fragebogen mit Zusatzfragen zu Tätigkeit, Deutschkenntnissen und Sehfähigkeit der Testperson.
3. Aufgabenstellung mit der APPsist-App (ThinkAloud-Analyse): Die Bearbeitung der Aufgaben, die nahezu den gesamten Funktionsumfang abdeckten, dauerte zwischen 20 und 45 Minuten. Dabei wurden die Testpersonen gebeten, laut zu denken und auch zu ihrer Meinung zu einigen Teilen des Systems bzw. der Benutzungsoberfläche gefragt. Dabei wurde zur Auswertung ein Bildschirm-Video inklusive Aufzeichnung des Gesprochenen gespeichert. Der Ablauf der Aufgabenstellung war standardisiert und die Fragen größtenteils vorformuliert. So können die Antworten zu geschlossenen Fragen auch quantitativ ausgewertet werden.
4. Fragebögen: System Usability Scale (SUS) [Br96], AttrakDiff [At]: Zwei schnell auszufüllende Fragebögen zur Bewertung von Gebrauchstauglichkeit (Usability) und Nutzungserlebnis (User Experience).
5. Abschlussinterview: Standardisierte Fragen zu Gesamteindruck, Vorerfahrung und Bewertung anhand der Dialogkriterien aus ISO-EN-9241-110. Vom Gespräch wurde zur Dokumentation und Auswertung jeweils eine Audioaufzeichnung gespeichert.

Dieser Ablauf hatte einen summativen Test der Gebrauchstauglichkeit zum Ziel, der sowohl qualitative als auch quantitative Test- und Auswertungsteile beinhaltet. Bis jetzt ausgewertet sind SUS und AttrakDiff. Mit einem SUS-Score von 86,9 und einer AttrakDiff-Bewertung, die sowohl was die hedonische Qualität (HQ: 1,16 Konfidenz: 0,45), als auch was die pragmatische Qualität (PQ: 1,63 Konfidenz 0,27) angeht, im Bereich „begehrt" liegt (Die Skala reicht von „überflüssig" über „neutral" bis „begehrt"), offenbarte das APPsist-System keinen akuten Handlungsbedarf. Auch eine erste Kurzauswertung der ThinkAloud-Analyse zeigte kein Gestaltungs-, Interaktions- oder Funktionskonzept, das von vielen Probanden einheitlich kritisch beurteilt worden wäre.

5 Fazit

Dieser Artikel gab einen Überblick über den aktuellen Stand des Projekt APPsist. Im Projektverlauf hat sich herausgestellt, dass die gewählten Methoden und Technologien sich zum großen Teil bewährt haben. So ist die Microservice-Architektur gut geeignet, um die einzelnen Dienste zu entwickeln und sie zu integrieren, allerdings um den Preis ei-

ner schwierigen Fehlerbehebung und nicht zufriedenstellend stabiler Client-Server Kommunikation. Die Ontologie erlaubte wie erhofft die präzise Beschreibung der jeweiligen Situationen der Anwendungspartner bei gleichzeitig allgemein formulierten Adaptionsregeln, auch wenn eine komplett deklarative Formalisierung nicht möglich war, sondern eine Kombination von Java und SPARQL notwendig gewesen ist. In den bisherigen Evaluationen wurde die Usability sehr positiv bewertet. Kommende Evaluationen werden noch die Güte der Adaptionsregeln untersuchen müssen.

Danksagung

Dieser Beitrag entstand im Rahmen des Projekts „APPsist Intelligente Wissensdienste für die Smart Production“, das vom Bundesministerium für Wirtschaft und Energie unter dem Kennzeichen 01MA13004C gefördert und vom DLR-Projektträger betreut wird.

Literaturverzeichnis

[At] AttrakDiff Publikationen. `http://www.attrakdiff.de/sience.html`. Last accessed: 2016-06-02.

[Br96] Brooke, J.: SUS: A quick and dirty usability scale. In (Jordan, P. W.; Weerdmeester, B.; Thomas, A.; Mclelland, I. L., Hrsg.): Usability evaluation in industry. Taylor and Francis, London, 1996.

[BR15] Blümling, Sabrina; Reithinger, Norbert: Aufbereitung und Speicherung von Erfahrungswissen im PLuTO-Projekt. In: 8. AAL-Kongress. VDE Verlag, 2015.

[Fr15] Freith, Sebastian; Schütze, Glenn; Ullrich, Carsten; Welling, Stefan; Kreimeier, Dieter; Kuhlenkötter, Bernd: Digitale Lernszenarien zur ganzheitlichen Unterstützung von Mitarbeitern im Arbeitsalltag. In: Proceedings of DeLFI Workshops 2015. Jgg. 1443 in CEUR Workshop Proceedings. CEUR-WS.org, S. 79–87, 2015.

[KWH13] Kagermann, Henning; Wahlster, Wolfgang; Helbig, Johannes, Hrsg. Deutschlands Zukunft als Produktionsstandort sichern: Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0, Abschlussbericht des Arbeitskreises Industrie 4.0. acatech Deutsche Akademie der Technikwissenschaften, Berlin, 2013.

[Ul15] Ullrich, Carsten; Aust, Matthias; Blach, Roland; Dietrich, Michael; Igel, Christoph; Kreggenfeld, Niklas; Kahl, Denise; Prinz, Christopher; Schwantzer, Simon: Assistenz- und Wissensdienste für den Shopfloor. In: Proceedings of DeLFI Workshops 2015. Jgg. 1443 in CEUR Workshop Proceedings. CEUR-WS.org, S. 47–55, 2015.

[Ul16] Ullrich, Carsten; Hauser-Ditz, Axel; Kreggenfeld, Niklas; Prinz, Christopher; Igel, Christoph: Assistenz und Wissensvermittlung am Beispiel von Montage- und Instandhaltungstätigkeiten. In (Wischmann, Stefan, Hrsg.): Zukunft der Arbeit – eine praxisnahe Betrachtung. Springer Verlag, 2016. In press.

[Ve] Vert.x Project: . `http://vertx.io/`. Last accessed 8.6.2015.