Web Services - zu groß für eingebettete Systeme ?

Elmar Zeeb[*], Andreas Bobek[*], Frank Golatowski[+] und Dirk Timmermann[*]

[*] Universität Rostock, 18051 Rostock, {elmar.zeeb, andreas.bobek, dirk.timmermann}@uni-rostock.de
[+] CELISCA - Center for Life Science and Automation, 18119 Warnemünde, frank.golatowski@celisca.de

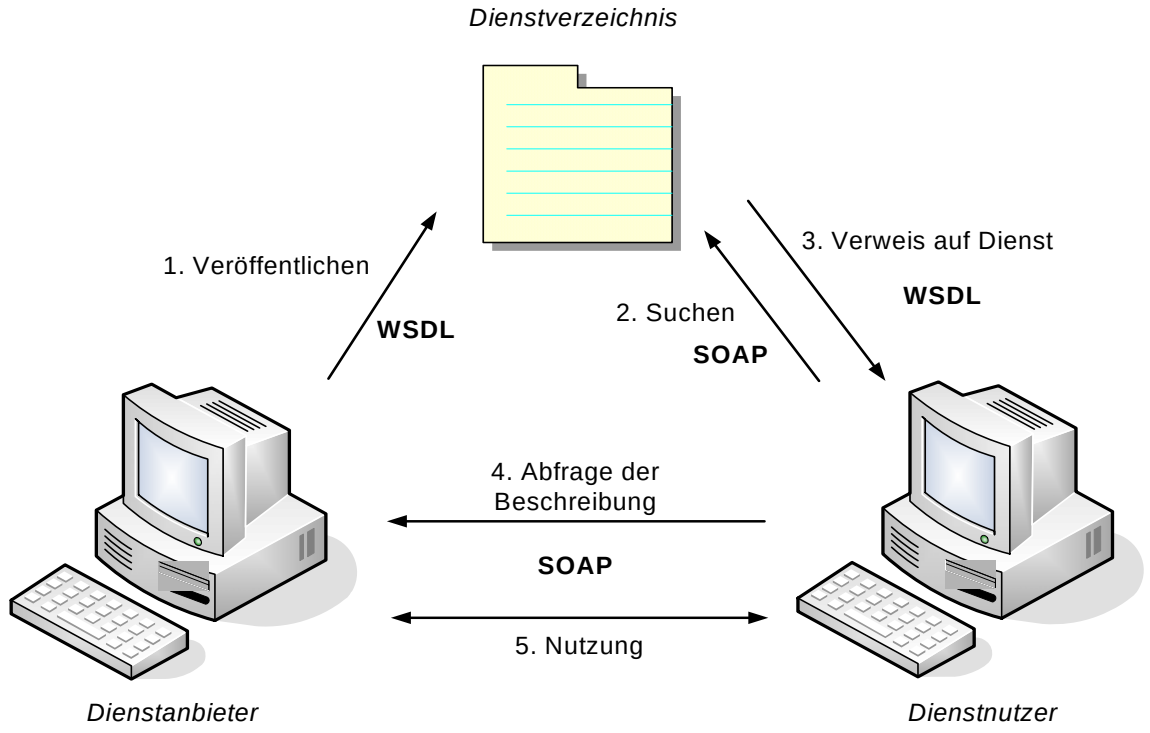
1. Einleitung

In den letzten Jahren ging die Entwicklung von eingebetteten Systemen zu autonomen und selbst konfigurierenden verteilten eingebetteten Systemen. Mit dem Einzug von Internettechnologien in die Bereiche Industrieautomatisierung, Telekommunikation, Heimkommunikation und Automobiltechnologie werden zudem Technologien mit größerem Funktionsumfang benötigt. Die Web Services Technologie ist ein viel versprechender Ansatz, dem allerdings noch ein Konzept zur Integration von Geräten mit beschränkten Ressourcen also eingebetteten Systemen fehlt.
In dieser Arbeit soll das Devices Pofile for Web Services (DPWS) (Schlimmer J. 2006) vorgestellt werden, das den Einsatz von Web Services auf Ressourcenbeschränkten Geräten ermöglicht. Es wird beschrieben, welche Einschränkungen und Erweiterungen DPWS für den Einsatz von Web Services zur Vernetzung von eingebetteten Systemen definiert. Neben der Spezifikation werden Werkzeuge zur Erstellung von Web Services für Geräte vorgestellt, die innerhalb des SIRENA ITEA Projektes entwickelt wurden und deren Ergebnisse in der WS4D Initiative fortgeführt werden.

2. Web Services und Service-orientierte Architekturen

Der Begriff Web Service wird heute mit Diensten verbunden, die über das Internet nutzbar sind und mit Hilfe von Internettechnologien realisiert werden. Es handelt sich dabei um Technologien, die dem Bereich der verteilten Systeme zugeordnet werden können. Die Technologien, die das W3C zur Realisierung von Web Services vorsieht, werden in der Web Services Architecture (Booth D. 2004) beschrieben. Hier werden vor allem XML, SOAP, WSDL und UDDI erwähnt.
Die Nachrichten, die zwischen Endpunkten ausgetauscht werden, sind in XML kodiert und entsprechend SOAP aufgebaut. Mit Hilfe von SOAP können die Nachrichten sowohl Plattform-übergreifend als auch Programmiersprachenu-

nabhängig verarbeitet werden. SOAP selber liegt über den Protokollen zur Übertragung von Nachrichten, ist also weitestgehend unabhängig von den darunter liegenden Protokollen, wird aber in den meisten Fällen mit HTTP verwendet. Die Schnittstellen der Dienste werden mit Dokumenten in der Web Services Description Language (WSDL) beschrieben. In diesem Dokument werden die Datentypen, die Operationen und die Bindungen an darunter liegende Protokolle, wie zum Beispiel HTTP, und die Adresse eines Dienstes definiert. Viele Web Services Frameworks unterstützen die Generierung von Stub und Skeleton mit Hilfe der Dienstbeschreibung in WSDL, was die Entwicklung von Web Services erheblich vereinfacht. Die Schnittstellendefinitionen werden bei dem zentralen Universal Description, Discovery and Integration Dienst (UDDI) hinterlegt. Ein Dienstnutzer kann bei diesem Service nach Services mit bestimmten Kriterien suchen und erhält als Antwort die Schnittstellenbeschreibungen der Dienste die diese Kriterien erfüllen.
Aufbauend auf den zuvor beschriebenen Standards gibt es beim W3C noch weitere Standards, die Protokolle für Sicherheit, Transaktionen, Richtlinien, Prozesse und weitere definieren. Diese Protokolle wiederum können zu Profilen zusammengefasst werden, um damit einen bestimmten Anwendungsfall abzudecken. Profile definieren, welche Protokolle verwendet und wie diese für einen bestimmten Zweck genutzt werden. Daher ermöglichen Profile die größtmögliche Interoperabilität, da die Protokolle noch zu viele Fragen bei der Implementierung offen lassen.



A

Abb. 1: Rollen und Abläufe in einer SOA und die Umsetzung mit Web Services.

Im Zuge der Web Services hat sich auch ein entsprechender Architektur-Stil für verteilte Systeme etabliert, der Service-oriented Architectures (SOA) genannt wird. „Unter einer SOA versteht man eine Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden oder Applikationen als wiederverwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachenunabhängige Nutzung und Wiederverwendung ermöglicht“(Dostal W. 2005). Die typischen Rollen und Abläufe in einer SOA, und wie diese mit Web Services umzusetzen sind, wird in Abbildung 1 gezeigt.
Die Vorteile von SOA und Web Services liegen vor allem in der Plattform- und Sprachunabhängigkeit und in der Wiederverwendbarkeit der Dienste. Zudem lassen sich Prozessmodelle sehr leicht in SOAs umsetzen und es gibt viele High-Level Konzepte wie zum Beispiel Workflow, die auf Web Services aufsetzen.

3. Web Services als Middleware für eingebettete Systeme

In (Asif M. 2007) wird ein Konzept vorgestellt wie Web Services im Bereich eingebetteter Systeme eingesetzt werden können. Es gibt noch weitere Konzepte zu diesem Thema, allerdings haben alle das Problem, dass der typische Architekturstil von verteilten Systemen basierend auf Web Services für bestimmte Szenarien im Bereich eingebetteter Systeme unzulänglich ist, da der zugrunde liegende Architekturstil SOA nicht immer alle Anforderungen eines Anwendungsszenarios erfüllen kann. SOA im Bereich der Web Services und damit auch WSA fehlen das Ad-Hoc Szenario und die Selbstbeschreibung der Geräte, da es einen zentralen Verzeichnisdienst vorsieht. Außerdem ist der Umfang mancher Spezifikationen wie zum Beispiel SOAP zu groß, um sie vollständig auf eingebetteten Systemen zu implementieren, was zu Inkompatibilitäten zwischen den Konzepten führt. Auch der hohe Kommunikations-Overhead und der hohe Aufwand bei der Verarbeitung von XML-Daten darf nicht unterschätzt werden.
Andere Middleware-Konzepte wie Universal Plug and Play (UPnP), Java Intelligent Network Infrastructure (JINI) und Home Audio/Video Ineroperability (HAVi) gehen viel stärker auf die Aspekte eingebetteter Systeme ein, haben allerdings das Problem, dass sie sich außerhalb ihrer Anwendungsgebiete nicht durchgesetzt haben. HAVi weist Mechanismen zur spontanen Vernetzung sowie für Dienstgüten auf, ist aber auf den Heimbereich beschränkt (Teirikangas J. 2001). JINI wurde von Sun Microsystems entwickelt und ermöglicht die spontane Vernetzung von Diensten und Ressourcen basierend auf der Java-Plattform. UPnP ermöglicht die spontane Vernetzung von Geräten und Diensten basierend auf einer eingeschränkten Version von SOAP hat aber Probleme bei der Skalierung der Mechanismen für die spontane Vernetzung und bietet kein Sicherheitskonzept.

DPWS ist eine neue Technologie basieren auf Web Services. Es wurde entwickelt, um sichere Web Services auf Geräte mit eingeschränkten Ressourcen zu bringen. Es beinhaltet sicheren Nachrichtenaustausch mit Web Services, spontane Vernetzung von Geräten und Diensten, Mechanismen zur Selbstbeschreibung von Geräten und der darauf angebotenen Dienste und einen Ereignismechanismus mit Abonnements. DPWS ist für die Steuerung von Geräten geeignet, wobei ein Control Point (Steuereinheit) die Client-Seite und das Gerät die Server-Seite implementieren muss. Die Implementierung der Server-Seite verlangt die geringsten Ressourcen, der Ressourcenbedarf der Client-Seite hängt vom Anwendungsszenario ab, da bestimmte Teile der Spezifikation nicht umgesetzt werden müssen, wenn sie nicht benötigt werden. DPWS ist auch für die Kommunikation zwischen Geräten geeignet. Dafür müssen allerdings beide Seiten implementiert werden, was den größten Ressourcenbedarf verlangt.

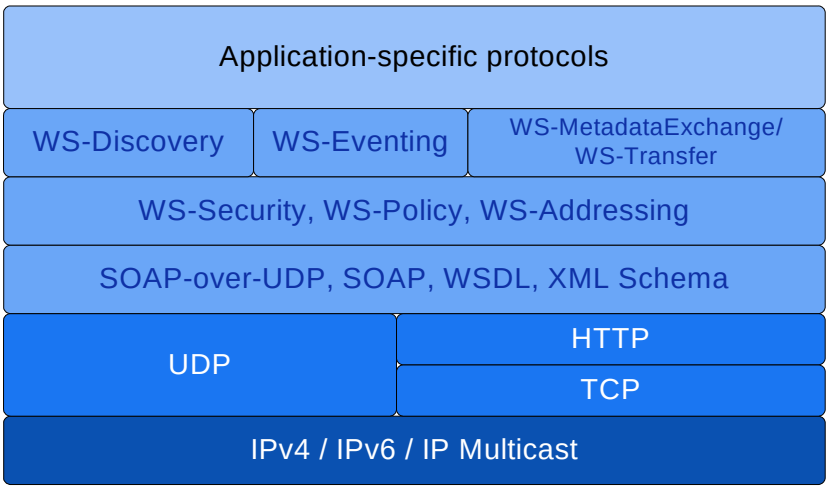


Abb. 2: Die in DPWS verwendeten Protokolle

Wie in Abbildung 2 gezeigt wird, basiert DPWS auf im Internet weit verbreiteten Protokollen und mehreren Web Service Spezifikationen. Der Nachrichtenaustausch findet in einer ähnlichen Weise wie in der Web Services Architektur statt. DPWS schränkt Nachrichten in der Komplexität und der Größe ein. Aufbauend auf Protokolle wie IP, IP-Multicast, TCP, UDP, und HTTP verwendet DPWS SOAP-over-UDP, SOAP, und XML Schema für den Nachrichtenaustausch. WS-Policy, WS-Addressing und WS-Security befinden sich über den Protokollen zum Nachrichtenaustausch. WS-Policy wird verwendet um Erfordernisse und Parameter auszuhandeln, die für die Service-Nutzung nötig sind. WS-Addressing löst das SOAP-Protokoll von seiner starken Abhängigkeit von HTTP. Es führt die Konzepte ein, um Dienstnutzer, Dienstanbieter und Nachrichten adressieren zu können. WS-Security dient zur Steuerung von Sicherheitsmechanismen um Standards wie XML-Encryption, XML-Signature und Secure Sockets Layer (SSL) wirksam einsetzen zu können.
Des Weiteren spezifiziert DPWS Mechanismen für die spontane Vernetzung von Geräten, Beschreibung der angebotenen Dienste auf einem Gerät sowie einen Ereignismechanismus. Die spontane Vernetzung basiert auf WS-Discovery,

SOAP-over-UDP und IP-Multicast. Geräte können ihre Verfügbarkeit im Netz bekannt geben und Clients können ein Netzwerk nach bestimmten Geräten durchsuchen. In ihrer Beschreibung, die der Client zur Laufzeit abfragen kann, sind sowohl Geräte- als auch Dienste-spezifische Metadaten hinterlegt. Die Struktur der Gerätespezifischen Daten wird in DPWS definiert. Diese Beschreibung enthält zum einen Daten, die für den Endanwender relevant sind und Daten die beschreiben welche Dienste sich auf dem Gerät befinden und unter welchen Adressen deren Beschreibung zu finden ist. Für die Beschreibung der Dienste wird WSDL verwendet. Diese kann der Client dazu verwenden, die gewünschte Schnittstelle zu bestimmen und dann den Dienst zu nutzen. Zusätzlich definiert DPWS einen Ereignismechanismus basierend auf WS-Eventing, der es einem Client mit Hilfe eines Abonnementsystems ermöglicht, sich für die Zustellung bestimmter Ereignisse auf dem Gerät einzutragen. Eine ausführlichere Beschreibung von DPWS ist in (Zeeb E. 2007) zu finden.
Somit sieht die SOA mit DPWS etwas anders als die zuvor vorgestellte SOA aus, wie in Abbildung 3 zu sehen ist. Das zentrale Diensteverzeichnis fällt weg und wird durch einen Mechanismus zur spontanen Vernetzung gekoppelt mit der Selbstbeschreibung der Geräte ersetzt. Alternativ kann bei Netzwerken, die sich über mehrere Netzwerksegmente erstrecken oder in denen sich viele Geräte befinden, ein Discovery-Proxy eingesetzt werden, der vergleichbar mit einem zentralen Diensteverzeichnis ist.
Im Rahmen der WS4D-Initiative stehen bald Werkzeuge zur Entwicklung von Diensten basierend auf DPWS für eingebettete Systeme für mehrere Sprachen und Plattformen zur Verfügung. Die Initiative, bestehend aus ehemaligen Partnern des SIRENA-Projektes, wird das Know-how des abgeschlossenen Projektes fortführen. Zusätzlich wird eine Plattform geschaffen um die noch junge DPWS Spezifikation bekannt zu machen. Bei der Implementierung dieser Werkzeuge hat sich gezeigt, dass DPWS technologisch nicht die effizienteste Lösung

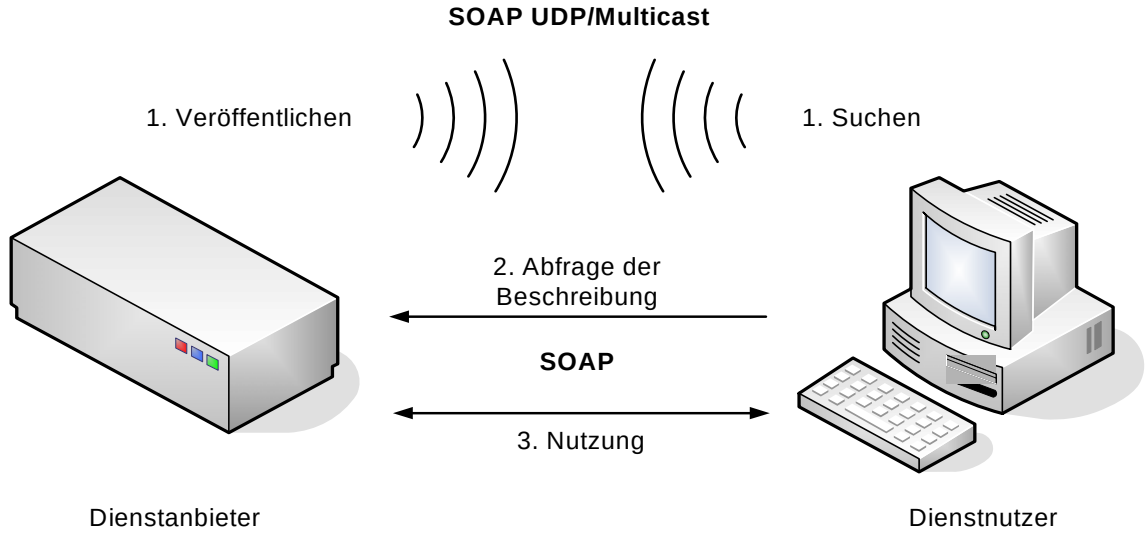


Abb. 3: Rollen und Abläufe in einer SOA speziell für eingebettete Systeme und DPWS.

für die Vernetzung eingebetteter System ist, allerdings einen sehr guten Kompromiss hinsichtlich der Interoperabilität über mehrere Anwendungsbereiche und mit anderen Technologien bildet.

4. Zusammenfassung

Wie in dieser Arbeit gezeigt wurde, können Web Services und SOAs mit DPWS auf eingebettete Systeme gebracht werden, ohne dass auf Funktionen wie spontane Vernetzung verzichtet werden muss. DPWS bietet einen Kompromiss zwischen Ressourcenverbrauch und Interoperabilität der den Einsatz zur Vernetzung von eingebetteten Systemen attraktiv macht und weiter Möglichkeiten eröffnet Highlevel-Technologien aus dem Bereich der Web Services auf eingebettete Systeme zu bringen. Zukünftig werden im Rahmen des LOMS-Projektes die Werkzeuge der WS4D-Initiative so erweitert, dass Web Services mit Kontextinformationen, wie zum Beispiel Ortsinformationen kombiniert werden können.

5. Danksagung

Diese Arbeit wurde über das LOMS Projekt vom Bundesministerium für Bildung und Forschung (BMBF) unter der Referenznummer 01|SF11H gefördert.

6. Literatur

Schlimmer J., Thelin J., Chan S., Conti D. et al (2006) Devices Profile for Web Services http://specs.xmlsoap.org/ws/2006/02/devprof/

Booth D., Haas, H., McCabe F et al (2004) Web Services Architecture http://www.w3.org/TR/ws-arch/

Dostal W., Jeckle M., Melzer I. und Zengler B. (2005) Service-orientierte Architekturen mit Web Services.

Asif M., Majumdar S. und Dragenea R. (2007) Hosting Web Services on Resource Constrained Devices

Teirikangas J. (2001) Home Audio Video Interoperability, Technical Report

Zeeb E., Bobek A., Bohn H. und Golatowski F. (2007) Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services