IoT – Your Way
Ein pragmatischer DSL-Ansatz

Christoph Woskowski und Alexander Leupold, Zühlke Engineering GmbH

Die Menge an Produkten und Lösungen im Bereich vernetzter Geräte beziehungsweise „Internet of Things“ ist überwältigend. Verschiedenste neue – und auch altbekannte – Übertragungstechnologien, Protokolle, Standards und die dahinterstehenden Konsortien und Interessensgruppen stehen bereit. Innerhalb dieser Bandbreite an Möglichkeiten sollte – so die Annahme – für jede Problemstellung und jeden Use Case eine passende Lösung zu finden sein. Dem gegenüber stehen vielfach sehr spezifische Anforderungen und Gegebenheiten von Unternehmen, die sich unter Umständen erstmals ernsthafter mit dem Thema IoT befassen – oder befassen müssen, weil der Wettbewerb diesen Schritt womöglich bereits gegangen ist. Vor allem dann, wenn Business Case und Use Cases noch unklar sind, sollte eine schwergewichtige Lösung beziehungsweise die frühzeitige Bindung an Technologien und Anbieter vermieden werden. Der hier vorgestellte Ansatz zur Entwicklung eines unternehmensweiten IoT-Datenmodells erfüllt diese Bedingung, indem er auf Leichtgewichtigkeit und maximale Flexibilität ausgelegt ist.

Das im Folgenden beschriebene Beispiel hat einen realen Hintergrund, betrifft jedoch auch eine strategische Initiative des dahinterstehenden Unternehmens. Daher wurde der Kontext etwas verfremdet. So wurde einerseits der Anwendungsbereich in die nicht verwandte Domäne „Haushaltsgeräte“ verschoben. Sowohl die Vorgehensweise als auch die gewonnenen Erfahrungen und daraus resultierende Schlussfolgerungen sind jedoch andererseits davon unabhängig und nicht auf einen Industriebereich beschränkt. Vielmehr geht es auch darum, die Flexibilität des Ansatzes aufzuzeigen. Insofern ist die „Portierbarkeit“ in eine andere Domäne bereits ein erstes wichtiges Indiz. Hervorzuheben ist zudem, dass es sich nicht um eine generalisierende Lösung handelt – also um einen abstrakten Ansatz der sich aufgrund seiner Distanz zum Problemraum eins zu eins auf viele bzw. so gut wie alle Bereiche anwenden lässt– sondern um echte Flexibilität im Sinne von Anpassbarkeit an das Zielproblem. Eine Anpassung muss also in jedem Fall vorgenommen werden.

Problemstellung

Ein Großunternehmen, welches technologisch und fachlich sehr breit aufgestellt ist, hat trotz Plattform- und Produktlinienansätzen mit einer großen Variantenvielfalt seiner am Markt und in der Entwicklung befindlichen Produkte zu kämpfen. Eine solche Situation kann zum Beispiel auch für einen Haushaltsgerätehersteller angenommen werden. Das Drängen auf Vereinheitlichung aus Kostengründen ist in diesem Fall nicht nur in der Entwicklung stark ausgeprägt, sondern ist auch bei der Herstellung, im Bereich Logistik, aber vor allem im Service- und Repair-Bereich spürbar. Der Standardisierungsdruck wird durch den Trend zu Mehrwertdiensten und geräteübergreifenden Funktionen (die Kaffeemaschine „weiß“, wann der Toaster fertig ist und passt die Getränkeausgabe entsprechend an) im Sinne von IoT noch einmal stärker. Lag die Datenhoheit – welche Daten werden wie erfasst, abgelegt und bereitgestellt – bisher bei den jeweiligen Entwicklungsprojekten in Abstimmung mit mehr oder weniger an Standards orientierten Stakeholdern, wird die Vielfalt an historisch gewachsenen Datenformaten und Datenmodellen nun zu einem Problem. Im Extremfall besitzt jedes Produkt in jeder neuen

Version ein eigenes Datenmodell, welches demjenigen, der die Daten nutzen möchte bzw. auf diese angewiesen ist, bekannt sein muss.

Auch wenn diese Vielzahl an Datenmodellen im vorliegenden Beispiel einem Metamodel mit einheitlicher Basis-Syntax folgen (z.B. gültige Datentypen sind (u)int8_t, (u)int16_t, (u)int32_t usw.), fehlt bislang die maschinenlesbare Semantik (Wert X ist eine Temperatur in °C mit einer Nachkommastelle) zur Interpretation der Werte. Außerdem gibt es keine einheitlichen Namenskonventionen, weshalb es allein für so gängige Betriebsdaten wie Laufzeit- und Nutzungszähler über alle Produkte hinweg bis zu mehrere Hundert Varianten gibt. Darunter sind einerseits viele Kopien (unterschiedliche Namen + gleiche Bedeutung), aber auch Mehrdeutigkeiten (gleiche Namen + unterschiedliche Bedeutung). Trotz einheitlichem und sehr flexiblem Protokoll zum Auslesen der Binärdaten aus den verschiedensten Produkten – sei es kabelgebunden oder per Funk – wird immer das spezifische Datenmodell benötigt, um den ausgelesenen Binär-„Blob“ zu interpretieren. Der Use Case ist hier natürlich eindeutig der Repair-Fall, wo alle verfügbaren Daten zur Analyse vom Gerät heruntergeladen und mit Hilfe einer großen Datenbank aus gerätespezifischen Datenbeschreibungen analysiert werden können.

Diese Variante funktioniert bereits nicht mehr, falls der Nutzer per Smartphone für ihn interessante Daten aus einem Haushaltsgerät auslesen möchte. Die zugehörige App müsste quasi alle Geräte kennen und über ihre Datenbeschreibungen in aktueller Version verfügen (auch im Offline-Fall). Spätestens wenn es um eine Kommunikation von Gerät zu Gerät geht (M2M-Fall), ist das Varianten-Chaos vorprogrammiert.

Die Aufgabe zur Lösung dieser Probleme besteht also einerseits in der Zusammenführung und Vereinheitlichung der existierenden Datenmodelle und andererseits im „Fit-Machen“ für neue Anwendungsfälle und Mehrwertdienste im Sinne von IoT. Ein wichtiger Aspekt dabei ist natürlich die Verwendbarkeit des neuen, einheitlichen Datenmodells in Low-Power- und bandbreitenbeschränkten Embedded-Systemen. Als Beispiel seien hier knopfzellenbetriebene Bluetooth LE Tags bzw. Beacons genannt. Auch Echtzeit-Kommunikation zwischen Geräten bei einer maximalen Verzögerung im Bereich weniger Millisekunden ist ein Use Case, den das neue Datenmodell ermöglichen soll. Es wird schnell klar, dass neben dem Datenmodell auch Datenformate und Übertragungsprotokolle zu entwickeln oder zumindest anzupassen sind.

Vorgehen

Initial ist es für einen außenstehenden Berater wichtig, den Ist-Stand zu verstehen, die beteiligten Stakeholder und ihre Anforderungen abzuholen und so die Rahmenbedingungen für mögliche Lösungen zu setzen. Ein erster Requirements-Workshop, welcher unter anderem die Entwicklung einer gemeinsamen Vision, das Festlegen des Kontextes (In- und Off-Topics), die Identifikation weiterer Stakeholder, das Festhalten von Szenarien und Use Cases und die Aufstellung einer ersten groben Timeline beinhaltet, ist eine mögliche Methode.

Der zweite Schritt zur Lösung eines jeden Problems sollte immer sein, zu prüfen, ob es nicht bereits existierende, adäquate Ansätze gibt, die sich mit geringem Aufwand auf die aktuelle Aufgabenstellung anwenden lassen. Externes Knowhow aus fachfremden Industriebereichen / Domänen ist hier durchaus hilfreich.

Die Recherche ergibt jedenfalls keine direkt anwendbaren Standards in der Zieldomäne, jedoch eine Vielzahl von generischen und domänenübergreifenden Ansätzen. Als Beispiele seien hier

nur OMA LWM2M und Eclipse Vorto genannt. Die tiefergehende Analyse anhand des ermittelten ersten Satzes an Use Cases und der zur Verfügung stehenden Technologien bestärkten vor allem Bedenken in Bezug auf die Skalierbarkeit existierender generischer Lösungen. Keiner der untersuchten Standards bzw. Datenmodellierungsansätze versprach die nötige horizontale (Use Case-bezogene) sowie vertikale (technologiebezogene) Erweiterbarkeit. Viele der existierenden Lösungen sind entweder ausgesprochen technologiespezifisch (z.B. LWM2M) oder unterstützen nur einen Teil der benötigten Use Cases – von Logdaten/Firmware Down-/Upload, über Personalisierung von Geräten bis hin zu Echtzeit-M2M-Kommunikation.

Andererseits liefert die Recherche jedoch eine interessante Auswahl wiederkehrender Lösungsmuster und Best Practices wie Unique Identifier, objektorientierte Ansätze, hierarchische Strukturen sowie Domain Specific Languages (DSLs). All diese Pattern eignen sich hervorragend als Building Blocks einer spezifischen Lösung.

Ausblick auf den Lösungsansatz

Wir haben einen DSL-Ansatz gewählt, um den einzelnen Entwicklungsprojekten des Kunden die Möglichkeit zu geben, die einem jeweiligen Gerät zugeordneten Daten in einer der Zieldomäne angepassten Sprache zu beschreiben. Statt sich über Integer, Boolean und Float bzw. Big- oder Little-Endian Gedanken zu machen, definiert der Entwickler z.B. Motorlaufzeiten, Betriebstemperaturen oder auch Eventlog-Einträge. Durch eine Gruppierung zusammengehöriger Daten basierend auf physischen Komponenten und übergreifenden Funktionen entsteht eine Bibliothek aus Bausteinen, die für andere ähnliche Geräte wiederverwendet werden können. Die Wiederverwendbarkeit und Erweiterbarkeit werden durch Sprachmittel, wie optionale Datenfelder innerhalb der Bausteine, Composites und Vererbung, sichergestellt. Einfache Generatoren und anpassbare Transformatoren setzen das jeweilige Modell in die beteiligten Technologien und Umgebungen um – von Low-Power-Embedded über Cloud-Backend bis zu Mobile-Clients – und ermöglichen gleichzeitig eine Anbindung bzw. Unterstützung existierender und zukünftiger Standards und Protokolle.

Die technologische Basis der schlussendlich implementierten Lösung wird durch einen extrem leichtgewichtigen DSL-Ansatz inkl. Parser-Generator und Templating-Mechanismus gebildet. Konkret wird ANTLR eingesetzt, um aus einer EBNF-Grammatik einen Parser in der Zielsprache Python zu generieren. Der darauf aufsetzende Python-Code ist sehr übersichtlich – unter 1000 Lines of Code – und dient hauptsächlich dazu, die Informationen für die anschließende Generierung bereitzustellen. Die Implementierung benötigter Generatoren bzw. Transformatoren erfolgt entweder ebenfalls in Python-Code oder unter Verwendung von Templates im Zielformat im Zusammenhang mit einer Templating-Engine.

Autoren

Christoph Woskowski ist Berater und Architekt für Embedded Systeme bei Zühlke und seit mehr als 8 Jahren im Unternehmen. Er verfügt über eine breite Erfahrung in der Konzeption und Umsetzung von Gerätesoftware. Er hat über viele Jahre in unterschiedlichen Rollen Projekte als Software- und Systemarchitekt begleitet und dort die gesamte Kette von der Anforderungserstellung über die Ableitung von Testfällen, den Nachweis der Anforderungen im Code bis hin zu Integrations- und Abnahmetests erfolgreich durchgeführt. Herr Woskowski begleitete auch mehrfach Embedded Architektur Reviews und Technologieauswahlen für kritische Systeme.

Kontakt
Internet: blog.zuehlke.com/en/author/christoph-woskowski
Email: christoph.woskowski@zuehlke.com

Alexander Leupold ist seit 2015 Embedded-Software-Entwickler bei Zühlke. Er entwirft und implementiert Software-Systeme im IoT- und Medical-Umfeld. Neben Funktechnologien wie ZigBee und Bluetooth LE hat er tiefgehende Erfahrung mit Continuous Integration und Testautomatisierung. Des Weiteren beschäftigt er sich mit IT-Security und führt Bedrohungsmodellierungen bei IoT- und Embedded-Systemen durch.

Kontakt
Email: alexander.leupold@zuehlke.com