SensIDL[i]: Ein Open-Source-Werkzeug für die Entwicklung von Kommunikationsschnittstellen smarter Sensorsysteme

Dr. Christoph Rathfelder[1], Dr. Henning Groenda[2] und Emre Taspolatoglu[2]

[1] Hahn-Schickard,
Wilhelm-Schickard-Str. 10, 78052 Villingen-Schwenningen,
christoph.rathfelder@hahn-schickard.de

[2] FZI Forschungszentrum Informatik am Karlsruher Institut für Technologie (FZI),
Haid-und-Neu-Str. 10-14, 76131 Karlsruhe
{groenda | taspolat}@fzi.de

Kurzfassung

Der zunehmende Grad der Vernetzung zwischen Systemen bezieht zunehmen ressourcenarme Sensorsysteme mit ein und spiegelt sich in Initiativen zum Internet der Dinge, Industrie 4.0 oder Smart-Home wieder. Mit SensIDL wollen wir ein Entwicklungswerkzeug realisieren, welches Sensorentwickler bei der Umsetzung von Kommunikationsschnittstellen für Sensorsysteme unterstützt. Das Werkzeug adressiert explizit den Zugriff auf diese Schnittstellen aus anderen Systemen heraus. Die Basis bilden eine Schnittstellenbeschreibungssprache mit zugehörigen Editoren sowie modellgetriebene Techniken. Diese Umsetzung ermöglicht den Entwicklern die einfache und effiziente Nutzung der Sensorschnittstellen. In diesem Paper und dem zugehörigen Poster geben wir einen Überblick über die Einsatzszenarien des SensIDL Werkzeuges und seinen Aufbau.

1 Motivation

Die allgegenwärtige mobile Nutzung des Internets betrifft sowohl den Heimbereich als auch das industrielle Umfeld. Alltagsgegenstände besitzen zunehmend Kommunikationsfähigkeiten. Der Mehrwert liegt in der einfachen Kombination und Integration von Systemen. Im Heimbereich werden Fernseher, Smartphones, aber auch Licht-, Fenster- und Heizungssteuerungen, Kühlschränke und ganze Hausautomatisierungssysteme intelligent vernetzt. Im Industrieumfeld wird die Vernetzung als Teil der vierten industriellen Revolution gesehen, welche im Rahmen der Industrie 4.0 Initiative [1] gefördert wird. Die Bandbreite der eingesetzten Systeme reicht von Cloud-Diensten über leistungsfähige PC-Systeme und mobile Endgeräte, wie Smartphones und Tablets, bis hin zu eingebetteten Sensorsystemen mit eingeschränkter Energieversorgung.

Durch die sehr heterogene Mischung von Systemen sowohl in Bezug auf deren Rechenkapazität und Kommunikationsanbindung als auch in den eingesetzten Prozessorarchitekturen, Betriebssystemen und Programmiersprachen, sind verschiedenste - zum größten Teil inkompatible - Kommunikationstechnologien und Schnittstellen im Bereich der eingebetteten smarten Sensorsysteme entstanden. Dies führt unter anderem dazu, dass die Vernetzung und Interoperabilität als eine zentrale Herausforderung angesehen wird [2].

Die Integration von intelligenten Sensorsystemen in die IT-Landschaft und die Kommunikation mit mobilen Endgeräten stellt für Entwickler immer eine aufwändige und fehlerträchtige Aufgabe dar. Es fehlen hier sowohl einheitliche Standards als auch effiziente Entwicklungswerkzeuge die den Entwickler bei der Programmierung unterstützen und entlasten. Der Erfolg von XML als Datenaustauschformat in leistungsfähigen Softwaresystemen zeigt die Notwendigkeit und Vorteile eines gemeinsamen Datenstandards für die Integration.

Entwickler setzen die Kommunikation und Datenverarbeitung aus diesen Gründen häufig manuell als individuelle Lösungen um. SensIDL vereinfacht oder löst die sich aus der manuellen Implementierung ergebenden Probleme:

- Hoher immer wiederkehrender Implementierungsaufwand für die Kommunikation
- Aufwändige und fehleranfällige manuelle Pflege für verschiedene Plattformen
- Einheitliche und konsistente Änderung von Schnittstellen
- Pflege der Dokumentation bei Änderungen der Schnittstellen

2 Anwendungsszenarien

SensIDL unterstützt sowohl die Entwickler von Sensorsystemen, die Daten liefern, als auch die Entwickler von IT-Systemen, die auf diese Daten zugreifen, bei der Umsetzung der Kommunikationsschnittstellen.

Die Einsatzpunkte von SensIDL werden am Beispiel der Übertragung von Temperaturmessdaten veranschaulicht. Abbildung 1 skizziert die notwendigen Verarbeitungsschritte um Sensorinformationen vom Sensorsystem zu einem mobilen Endgerät zu übertragen.

Nachdem die Messwerte des Sensormoduls für die Temperatur erfasst wurden, muss der Wert zur Übertragung in eine Bitfolge umgewandelt werden. Meist

besitzt ein Sensorsystem mehrere Sensormodule für unterschiedliche Messwerte, die dann gemeinsam in eine Bitfolge umgewandelt werden müssen. Aktuell müssen Sensorentwickler die Umwandlung und Zusammenstellung der Messwerte selbst implementieren. Die Lösungsalternativen reichen von einer sensorspezifischen Binär-Repräsentation über ASCII Darstellungen bis hin zu strukturierten Datenformaten wie XML oder JSON. Diese Bitfolgen werden dann einer Übertragungstechnologie übergeben, die sicherstellt, dass die Bitfolge zum Empfänger übertragen wird. Nachdem die Bitfolge beim empfangenden IT-System entgegengenommen wurde, müssen die Daten wieder zurück umgewandelt werden. Hierzu muss der Software-Entwickler wissen, wie der Sensorentwickler die Daten codiert hat. Ohne dieses Wissen, ist der Software-Entwickler nicht in der Lage die verarbeitbaren Nutzdaten/Informationen aus der Bitfolge zu extrahieren. Das Wissen wird häufig nicht dokumentiert. Der Zugriff erfordert somit reverse-engineering-Techniken und hohen Aufwand.

Mit SensIDL wollen wir sowohl den Sensorentwickler wie auch den Software-Entwickler für IT-Systeme unterstützen und wiederkehrende Aufgaben automatisieren.

Der **Sensorentwickler** wird durch die entwickelte SensIDL Sprache und die dazugehörigen Editoren dabei unterstützt, die Sensordaten und –schnittstellen eindeutig auf einer semantischen Ebene zu beschreiben. Die notwendige Implementierung der Schnittstelle kann durch das Werkzeug aus der Beschreibung automatisiert abgeleitet werden. Ein Sensorentwickler muss sich nicht mehr mit der Implementierung der Datencodierung beschäftigen und kann sich auf die Verarbeitungslogik innerhalb des Sensors fokussieren.

Der **Software-Entwickler** für das empfangende IT-System bekommt mit der vom Sensorentwickler erstellten Schnittstellenbeschreibung eine formale und eindeutige Beschreibung der Schnittstelle basierend auf der SensIDL Sprache. Der notwendige Code auf Empfängerseite kann ebenfalls abgeleitet werden. Hierdurch wird die Implementierung des Zugriffs auf Sensordaten und somit die Integration signifikant vereinfacht.

3 Das Projekt SensIDL

Ziel des Projektes SensIDL (http://www.sensidl.de) ist die Entwicklung eines Open-Source-Entwicklungswerkzeugs für die Spezifikation und effiziente Übermittlung von Daten smarter Sensoren unabhängig von gewählten Kommunikationstechnologien.

Eine einheitliche Spezifikationssprache ermöglicht die Definition der von Sensoren angebotenen Daten. Diese Beschreibung dient als Basis für eine automatisierte Code-Generierung sowohl für die Sensor- als auch die Empfängerseite. Wie in Abbildung 2 veranschaulicht, nutzt der im Framework integrierte Generator die Schnittstellendefinition als Eingabe um eine semantisch angereicherte schnittstellenspezifische API zu erzeugen. Diese API erlaubt dem Entwickler einen einfachen Zugriff auf die spezifizierten Daten und kapselt mittels des SensIDL Kommunikationsframework konkrete Kommunikationstechnologien. Dies erlaubt die effiziente (De-)Codierung und Übertragung der Daten ohne zusätzliche Entwicklungsarbeit

Das Vorgehen vereinfacht die Implementierung sowohl der smarten Sensoren als auch der empfangenden Gegenstelle signifikant. Hierdurch wird sowohl die Fehleranfälligkeit als auch die Entwicklungszeit reduziert, was einen entscheidenden Wettbewerbsvorteil bedeuten kann. Zusätzlich dazu wird der Entwickler von den wiederkehrenden Aufgaben der Schnittstellen- und Kommunikationsentwicklung entlastet und kann sich auf seine Kernaufgaben fokussieren.

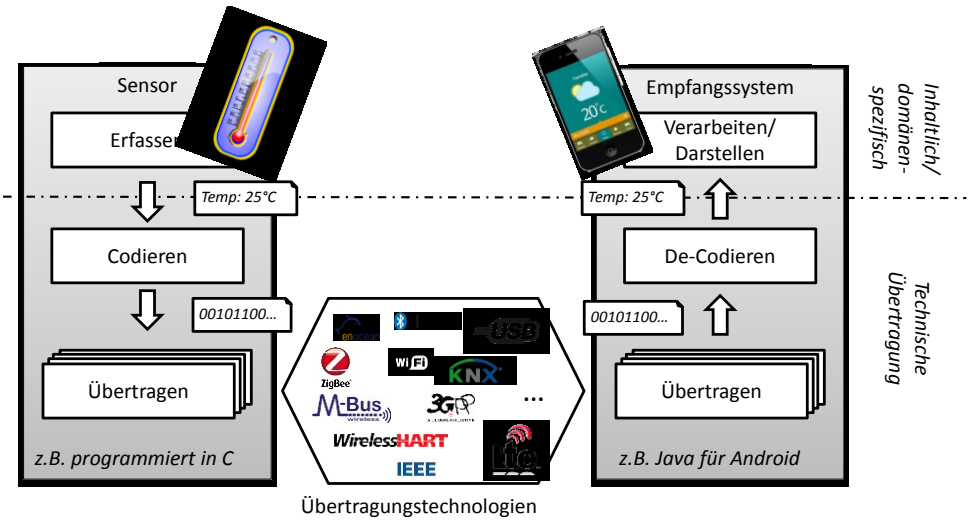

Abbildung 1: Anwendungsszenario

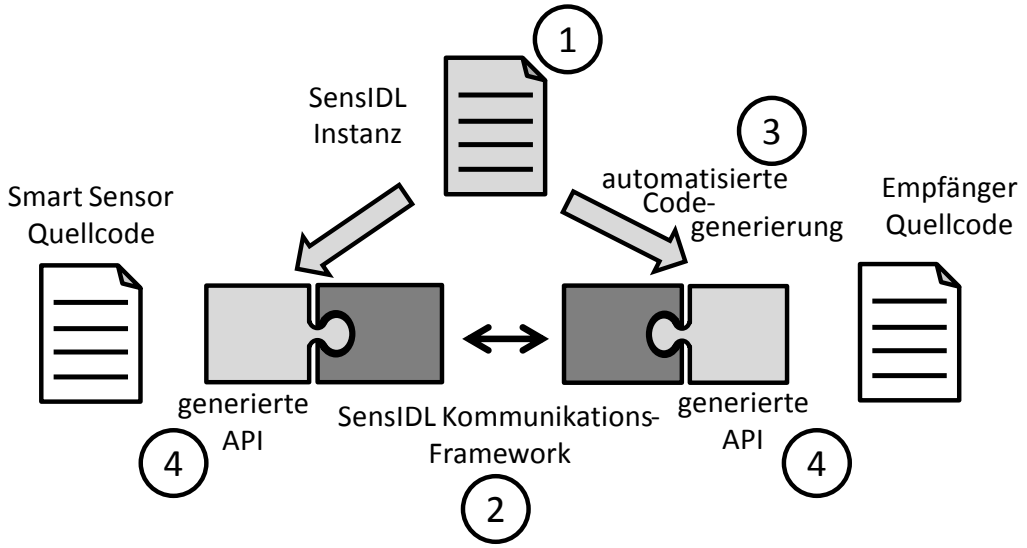

Abbildung 2: SensIDL Überblick

Im Folgenden geben wir einen kurzen Überblick über die verschiedenen Bestandteile des SensIDL Frameworks, welche auch in Abbildung 2 dargestellt sind:

1. Die **SensIDL Sprache** und der **SensIDL Editor** als Teil des Werkzeugs ermöglichen es, die von smarten Sensoren angebotene Schnittstellen und Daten einfach, intuitiv und semantisch angereichert zu spezifizieren und zu dokumentieren.
2. Das **SensIDL Kommunikations-Framework s**tellt verschiedene Methoden für eine effiziente Kodierung und Übertagung der spezifizierten Daten zur Verfügung.
3. Die **SensIDL Code-Generatoren** sind das Bindeglied zwischen der SensIDL Sprache und dem Kommunikations-Framework und leiten automatisiert eine für den Entwickler nutzbare sensor-spezifische API ab.
4. Die **sensorspezifische SensIDL API** bietet eine einfach zu verwendende sensorspezifische Programmierschnittstelle, welche den Zugriff auf das zugrundeliegende Kommunikations-Framework kapselt und den Programmierer von allen kommunikationsbezogenen Implementierungsaufgaben entlastet.

4 Modellgetriebener Ansatz

Die modellgetriebene Software-Entwicklung kombiniert die beiden technischen Konzepte der domänenspezifischen Sprachen (domain-specific languages DSLs) und Code-Generierung mittels Modelltransformationen [3]. Anstelle einer Vorstellung und Diskussion der generellen Konzepte und Vorteile der modellgetriebenen Software-Entwicklung, wie z.B. Trennung von Belangen, Abstraktionsebenen, schnellere Entwicklungszyklen oder geringere Fehlerwahrscheinlichkeit [4] fokussieren wir im Folgenden die Anwendung der Konzepte in SensIDL.

4.1 Domänen-spezifische Sprache

Domänen-spezifische Sprachen bieten formale Konzept, um Strukturen, Abhängigkeiten, Verhaltensmodelle oder Anforderungen von Software-Anwendungen innerhalb definierter Domänengrenzen zu beschreiben [3]. Dies schränkt die Anwendbarkeit im Vergleich zu Turingvollständigen Programmiersprachen wie Java oder C ein. Im Gegensatz dazu ermöglichen DSLs den Entwicklern sich auf die wichtigen Aspekte der Modellierung konzentrieren können. Aus diesem Grund werden DSLs häufig dann angewendet, wenn die Komplexität reduziert bzw. gekapselt werden soll oder Technologien und Implementierungsdetails ausgeblendet werden sollen [5]. DSLs besitzen immer ein formales Modell im Hintergrund. Dieses wird als Metamodell oder abstrakte Syntax bezeichnet, da es die Struktur und den Aufbau von Informationen vorgibt. Die DSL ist eine konkrete Syntax und definiert wie die Elemente des Metamodells dem Benutzer präsentiert von diesem mit Inhalt gefüllt werden.

In SensIDL entwickeln wir eine domänen-spezifische Sprache für die Beschreibung der Schnittstellen von Sensorsystemen. Der Entwickler soll sich dabei auf die semantische Beschreibung der Daten konzentrieren, die von einem Sensorsystem geliefert werden, ohne sich bereits Gedanken über deren Codierung oder Übertragung machen zu müssen. Die SensIDL Sprache und die dazugehörigen Editoren sollen die Arbeit der Sensorentwickler bei der Definition der Kommunikationsschnittstellen, der übertragenen Daten und deren Struktur vereinfachen. Durch die spezifische Beschreibungssprache und die darauf abgestimmten graphischen oder textuellen Editoren fällt den Sensorentwicklern die Beschreibung und Dokumentation der Schnittstellen leicht. Sie können sich auf ihre eigentlichen Entwicklungsaufgaben konzentrieren und können so die Produktivität und Qualität steigern [5]. Durch die Verwendung einer speziellen Sensorbeschreibungssprache, welche explizit für die Beschreibung der Schnittstellen entwickelt wurde, versprechen wir uns eine bessere Trennung der verschiedenen Sichten und Aufga-

ben bei der Entwicklung. Die Ergänzung mit Code-Generatoren, wie im Folgenden beschrieben, führt zu einer signifikanten Steigerung der Entwicklungs- und Implementierungs-Effizienz.

4.2 Code-Generatoren

Code-Generatoren und Transformationsmechanismen sind das zentrale Element für die praktische Umsetzung der modellgetriebenen Software-Entwicklung. Sie erzeugen aus dem per DSL bearbeiteten Modell automatisiert neue Elemente wie zum Beispiel Quellcode. Mehrstufige Verfeinerungen zur Anreicherung um plattform- oder technologiespezifischen Informationensind möglich. Das Generieren von Code auf Basis einer DSL ist mit dem Kompilieren einer Programmiersprache in ausführbaren Code vergleichbar [5].

In SensIDL werden Code-Generatoren zur Erzeugung einer semantisch angereicherten API (application programming interface) verwendet. Diese API ermöglicht dem Entwickler auch auf beschränkten Sensoren einen intuitiven Zugriff zum Festlegen der Werte. Weitere Code-Generatoren verbinden die semantisch angereicherte und sensor-spezifische API mit dem Kommunikationsframework. Dieses Framework stellt einzelne Technologieadapter zur Verfügung, so dass die Daten über verschiedene Kommunikationstechnologien übertragen werden können. Die automatisierte Generierung des Codes sowohl für einen Sensor als auch Empfänger aus derselben Schnittstellenbeschreibung heraus stellt die Kommunikation und Datenkonsistenz sicher. Die Automatisierung wiederkehrender Implementierungsaufgaben wie Codierung und Decodierung in Kombination mit der Generierung einer typsicheren API auf Sensor und Empfängerseite lässt eine Effizienzsteigerung und Reduktion der potentiellen Fehlerquellen bzw. Qualitätssteigerung des Codes vermuten, wie sie auch in [4] und [6] versprochen werden.

5 Zusammenfassung

In diesem Paper und dem zugehörigen Poster haben wir einen Überblick über Anwendungsszenarien und Aufbau des SensIDL Werkzeuges gegeben. Die SensIDL Entwicklungsumgebung unterstützt mit einer dedizierten Schnittstellenbeschreibungssprache und zugehörigen Editoren die effiziente Definition und Dokumentation der Schnittstellen durch den Sensorentwickler. Code-Generatoren, welche diese Beschreibung als Eingabe verwenden, erzeugen automatisiert den auf Sensor- und Empfängerseite notwenigen Code, um die Daten zu instantiieren und zu kommunizieren. Hierdurch werden die Entwickler entlastet und die Effizient der Entwicklung und Implementierung kann gesteigert werden.

6 Literaturverzeichnis

[1] Forschungsunion, „Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0,“ http://www.plattform-i40.de/sites/default/files/Abschlussbericht_Industrie4%200_barrierefrei.pdf, 2013.

[2] acatec, Cyber-Physical Systems, Springer-Verlag Berlin Heidelberg, 2011.

[3] D. C. Schmidt, "Model-Driven Engineering," *IEEE Computer,* vol. 39, no. 2, pp. 25-31, 2006.

[4] T. Stahl und M. Völter, Modellgetriebene Softwareentwicklung, Heiderlberg: dpunkt.verlag, 2005.

[5] M. Fowler, Domain-Specific Languages, Addison Wesley, 2010.

[6] C. Bunse, H.-G. Gross and C. Peper, "Applying a Model-based Approach for Embedded System Development," in *33rd EUROMICRO Conference on Software Engineering and Advanced Applications*, Lubeck, 2007.

[7] Acatech, „agendaCPS - Integrierte Forschungsagenda Cyber-Physical Systems,“ März 2012. [Online]. Available: http://www.acatech.de/de/publikationen/empfehlungen/acatech/detail/artikel/acatech-studie-agendacps-integrierte-forschungsagenda-cyber-physical-systems.html. [Zugriff am Februar 2015].

[8] Eclipse Foundation, „Eclipse Modeling Project,“ [Online]. Available: http://www.eclipse.org/modeling/. [Zugriff am October 2014].

[9] R. Reussner und W. Hasselbring, Handbuch der Software Architektur, Heidelberg: dpunkt.verlag, 2006.

[10] N. Hili, C. Fabre, S. Dupuy-Chessa and D. Rieu, "A model-driven approach for embedded system prototyping and design," in *25th IEEE International Symposium on Rapid System Prototyping (RSP)*, New Delhi, 2014.

[11] EPoSS - The European Technology Platform on Smart Systems Integration, „Strategic research agenda of the european technology platform on smart system integrartion,“ 2013. [Online]. Available: http://www.smart-systems-integration.org/public/documents/publications/EPoSS%20SRA%20Pre-Print%20September%202013. [Zugriff am Jan 2015].

[i] *Das IGF-Vorhaben (18363N) der Forschungsvereinigung (Hahn-Schickard Gesellschaft für angewandte Forschung e.V.) wurde über die AiF im Rahmen des Programms zur Förderung der Industriellen Gemeinschaftsforschung (IGF) vom Bundesministerium für Wirtschaft und Energie aufgrund eines Beschlusses des Deutschen Bundestages gefördert.*