



□ Dr.-Ing. Henning Groenda

(groenda@fzi.de)

ist Abteilungsleiter im Forschungsbereich Software Engineering des FZI Forschungszentrum Informatik in Karlsruhe. Seine Arbeitsschwerpunkte sind der Architekturdentwurf und die frühzeitige Sicherstellung von Qualitätseigenschaften beim Entwurf und der Entwicklung von Software. Dies umfasst sowohl die Neuentwicklung als auch die Pflege bestehender Systeme und schließt die Berücksichtigung generierter Bestandteile eines Systems mit ein. Er ist für die Forschungsfelder Software- und Architektur-analyse sowie Mobile- und Multiplattformentwicklung verantwortlich.



□ Dr.-Ing. Christoph Rathfelder

(Christoph.Rathfelder@Hahn-Schickard.de)

studierte Informatik an der Universität Karlsruhe und promovierte 2012 am Karlsruher Institut für Technologie im Bereich der Qualitätsanalysen für verteilte Systeme. Seit 2013 ist Dr. Rathfelder im Bereich Sensoren und Systeme des Hahn-Schickard-Instituts für Mikro- und Informationstechnik in Villingen-Schwenningen für die Software- und App-Entwicklung verantwortlich. Er ist Mitglied des Lenkungskreises der Smart Home & Living Initiative Baden-Württemberg sowie verschiedener nationaler und internationaler Arbeitsgruppen zu den Themen Industrie 4.0, Smart Systems und dem Internet der Dinge bei MST BW, der Bitkom sowie der European Plattform on Smart System Integration (EPoSS).



□ Emre Taspolatoglu

(taspolat@fzi.de)

studierte Informatik am Karlsruher Institut für Technologie (KIT). Seine Diplomarbeit führte er im Forschungsbereich SE des FZI durch und beschäftigte sich mit den Code-relevanten Entwurfsentscheidungen während der Software-Evolution. Seit September 2014 ist er für das FZI Forschungszentrum Informatik als wissenschaftlicher Mitarbeiter im Forschungsbereich Software Engineering (SE) tätig und beschäftigt sich als Doktorand mit sicherheitsrelevanten Fragestellungen und Entwurfsentscheidungen auf der Ebene von Softwarearchitekturen sowie deren mus-terbasierten Analysen.

SensIDL: Ein Werkzeug zur Vereinfachung der Schnittstellenimplementierung intelligenter Sensoren

Die allgegenwärtige mobile Nutzung des Internets sowie die zunehmende Integration von Kommunikationsfähigkeiten in Alltagsgegenstände sowohl im Heimbereich als auch im industriellen Umfeld, besser bekannt als das Internet der Dinge, führen zu einer zunehmenden Vernetzung verschiedenster Systeme. Im Heimbereich werden Fernseher, Smartphones, aber auch Licht-, Fenster- und Heizungssteuerungen, Kühlschränke und ganze Hausautomatisierungssysteme vernetzt. Im Industrieumfeld wird die Vernetzung als Teil der vierten industriellen Revolution stark intensiviert. Die Bandbreite der eingesetzten Systeme reicht von hochleistungsfähigen Server- und PC-Systemen über Cloud-Dienste und mobile Endgeräte, wie Smartphones und Tablets, bis zu intelligenten eingebetteten mobilen oder stationären heterogenen Sensorsystemen mit eingeschränkter Energieversorgung und begrenzten Rechenkapazitäten.

Aktuelle Situation

Die Integration von intelligenten Sensorsystemen in die IT-Landschaft und das Internet der Dinge sowie die Kommunikation mit mobilen Endgeräten stellen für Entwickler immer eine aufwendige und fehlerträchtige Aufgabe dar [For13]. Es fehlen hier sowohl einheitliche Standards als auch effiziente Entwicklungswerkzeuge, die den Entwickler bei der Programmierung unterstützen und entlasten.

Durch die sehr heterogene Mischung von Systemen in Bezug auf deren Rechenkapazität und Kommunikationsanbindung sowie die eingesetzten Prozessorarchitekturen, Betriebssysteme und Programmiersprachen sind sogar verschiedenste zum größten Teil inkompatible Kommunikationstechniken und Schnittstellen im Bereich der eingebetteten Systeme entstanden [Aca11]. Der Erfolg von XML als Datenaustauschformat in größeren Softwaresys-

temen zeigt die Notwendigkeit eines gemeinsamen Datenstandards für die Integration verschiedener Systeme.

Aufgrund des Mangels eines einheitlichen Standards sowie unterstützender Entwicklungswerkzeuge sind Entwickler meist gezwungen, die Kommunikation manuell als individuelle Lösungen selbst zu implementieren, um innerhalb ihrer Anwendungen und Komponenten auf die Daten zuzugreifen und diese zu verarbeiten.

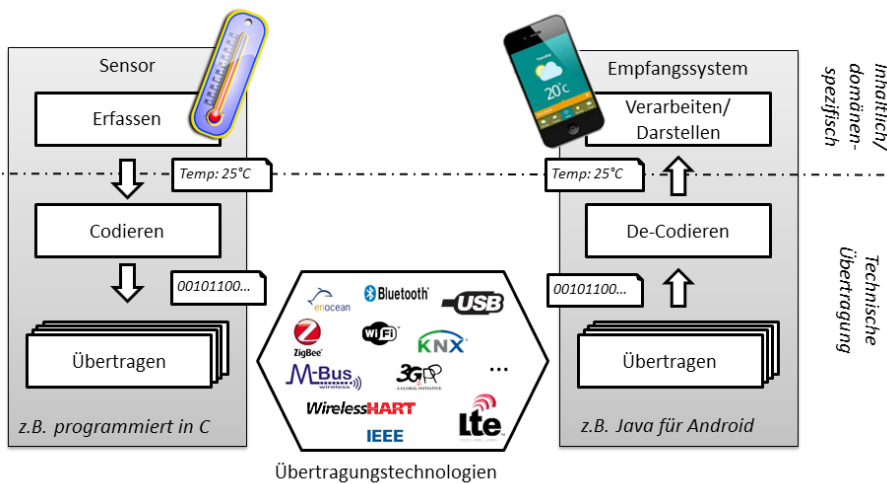


Abb. 1: SensIDL-Einsatzszenario

Durch die manuelle Implementierung ergeben sich die folgenden Probleme, welche durch die Werkzeuge von SensIDL gelöst oder vereinfacht werden sollen:

- Hoher immer wiederkehrender Implementierungsaufwand
- Manuelle Pflege für verschiedene Plattformen ist aufwendig und fehleranfällig
- Nachträgliche Schnittstellenänderungen erzeugen ein hohes Fehlerpotenzial, da Anpassungen manuell durchgeführt werden müssen
- Verbesserung der Dokumentation durch Kopplung von Schnittstellenbeschreibung und -implementierung

Anwendungsszenarien

SensIDL unterstützt sowohl die Entwickler von Sensorsystemen, welche 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 Bi-

när-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 Softwareentwickler wissen, wie der Sensorentwickler die Daten codiert hat. Ohne dieses Wissen ist der Softwareentwickler nicht in der Lage, die bearbeitbaren Nutzdaten/Informationen aus der Bitfolge zu extrahieren. Das Wissen wird häufig nicht ausreichend dokumentiert.

SensIDL will sowohl den Sensorentwickler wie auch den Softwareentwickler 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 so auf die Verarbeitungslogik innerhalb des Sensors fokussieren.

Der **Softwareentwickler** 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 vereinfacht.

Das Werkzeug SensIDL

Das Open-Source-Projekt SensIDL (vgl. [SensIDL]) hat die Entwicklung eines Open-Source-Entwicklungswerkzeugs für die Spezifikation und effiziente Übermittlung von Daten smarter Sensoren unabhängig von gewählten Kommunikationstechnologien als Ziel. Eine einheitliche Spezifikationsprache 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-Kommunikations-Frameworks 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.

Das SensIDL-Framework lässt sich in die in **Abbildung 2** dargestellten und im Folgenden genauer erläuterten Bausteine unterteilen:

1. Die SensIDL-Sprache und der SensIDL-Editor als Teil des Werkzeugs ermöglichen es, die von smarten Sensoren angebotenen Schnittstellen und Daten einfach, intuitiv und semantisch angereichert zu spezifizieren und zu dokumentieren.
2. Das SensIDL-Kommunikations-Framework stellt verschiedene Methoden für eine effiziente Codierung und Übertragung 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 sensorspezifische API ab.
4. Die sensorspezifische SensIDL-API bietet eine einfach zu verwendende sensorspezifische Programmierschnittstelle, welche den Zugriff auf das zugrunde liegende Kommunikations-Framework kapselt und den Programmierer von allen kommunikationsbezogenen Implementierungsaufgaben entlastet.

Modellgetriebener Ansatz

Die modellgetriebene Softwareentwicklung kombiniert die beiden technischen Konzepte der domänenspezifischen Sprachen (domain-specific languages DSLs) und der Code-Generierung mittels Modelltransformationen [Sch06]. Anstelle einer Vorstellung und Diskussion der generellen Konzepte und Vorteile der modellgetriebenen Softwareentwicklung, wie z. B. Trennung von Belangen, Abstraktionsebenen, schnellere Entwicklungszyklen oder geringere Fehlerwahrscheinlichkeit [Sta05] fokussieren wir uns im Folgenden auf die Anwendung der Konzepte in SensIDL.

Domänenspezifische Sprache

Domänenspezifische Sprachen bieten formale Konzepte, um Strukturen, Abhängigkeiten, Verhaltensmodelle oder Anforderungen von Softwareanwendungen innerhalb definierter Domänengrenzen zu beschreiben [Sch06]. 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 zu können. Aus diesem Grund werden DSLs häufig dann angewendet, wenn die Komplexität reduziert bzw. gekapselt werden soll oder Technologien und Implementierungsdetails ausgeblendet werden sollen [Fow10]. 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 und von diesem mit Inhalt gefüllt werden können.

In SensIDL entwickeln wir eine domänenspezifische Sprache für die Beschrei-

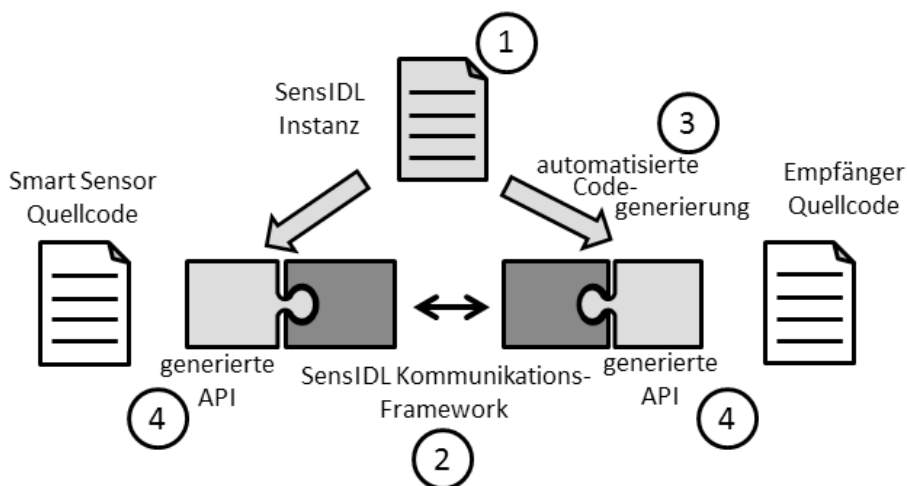


Abb. 2: Bausteine des SensIDL-Frameworks

bung 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 grafischen 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 [Fow10].

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

Code-Generatoren

Code-Generatoren und Transformationsmechanismen sind das zentrale Element für die praktische Umsetzung der modellgetriebenen Softwareentwicklung. Sie erzeugen aus dem per DSL bearbeiteten Modell automatisiert neue Elemente wie Quellcode. Mehrstufige Verfeinerungen zur Anreicherung um plattform- oder

technologiespezifische Informationen sind möglich. Das Generieren von Code auf Basis einer DSL ist mit dem Kompilieren einer Programmiersprache in ausführbaren Code vergleichbar [Fow10].

In SensIDL werden Code-Generatoren zur Erzeugung einer semantisch angereicherten API 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 sensorspezifische API mit dem Kommunikations-Framework. Dieses Framework stellt einzelne Technologieadapter zur Verfügung, sodass 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 typischeren API auf Sensor und Empfängerseite lässt eine Effizienzsteigerung und Reduktion der potenziellen Fehlerquellen bzw. Qualitätssteigerung des Codes vermuten, wie sie auch in [Bun07] und [Sta05] versprochen werden.

Zusammenfassung

SensIDL [SensIDL] ist ein aktuell in der Entwicklung befindliches Open-Source-Werkzeug für die vereinfachte Implementierung von Kommunikationsschnittstellen intelligenter und ressourcen-eingeschränkter Sensorsysteme als kleinste Bausteine des Internets der Dinge. Die SensIDL-Entwicklungsumgebung unter-

stützt 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 automatisch den auf Sensor- und Empfängerseite notwendigen Code in Form einer semantisch angereicherten API, um die Daten zu instanziiieren und zu kommunizieren. Hierdurch werden die Entwickler entlastet und die Effizienz der Entwicklung und Implementierung sowie deren Qualität kann gesteigert werden. ■

Referenzen

[SensIDL] <http://www.sensidl.de>

[Sch06] D. C. Schmidt, Model-Driven Engineering, IEEE Computer, vol. 39, no. 2, pp. 25-31, 2006.

[Sta05] T. Stahl und M. Völter, Modellgetriebene Softwareentwicklung, Heidelberg: dpunkt.verlag, 2005.

[Fow10] M. Fowler, Domain-Specific Languages, Addison Wesley, 2010.

[Bun07] 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.

[For13] Forschungsunion, „Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0,“ http://www.plattform-i40.de/sites/default/files/Abschlussbericht_Industrie4%20_barrierefrei.pdf, 2013.

[Aca11] acatec, Cyber-Physical Systems, Springer-Verlag Berlin Heidelberg, 2011.

Das IGF-Vorhaben 18363 N der Forschungsvereinigung Hahn-Schickard-Gesellschaft 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.