Erstellung vollständiger Systemspezifikationen im Embedded Computing

Roland Kapeller

MID Enterprise Software Solutions GmbH
Eibacher Hauptstr. 141
90451 Nürnberg
r.kapeller@mid.de

1 Zusammenfassung

Im Bereich der Entwicklung eingebetteter Systeme sind formale und semiformale Ansätze zur Spezifikation seit langem gang und gäbe. Mit Blick auf die Produktion elektronischer Steuergeräte im Automobilbereich wird ein praktikables Verfahren vorgestellt, welches die Prozesse des Lieferanten (hier konkret der Automobilzulieferer) solcher Systeme möglichst gut unterstützt. Besonderen Wert wurde dabei auf die methodische Unterstützung der konkreten Tätigkeit des Spezifizierens gelegt. Ergänzende Hinweise betreffen die Realisierung eines entsprechenden Workflows mithilfe gängiger Tools.

2 Problemstellung

Der hier beschriebene Ansatz beantwortet die Frage, wie ein zu realisierendes eingebettetes System so zu spezifizieren sei, daß nicht nur die Grundforderung nach Übernahme *aller* relevanten Kundenanforderungen des Lastenhefts erfüllt ist, sondern außerdem folgende Sachverhalte ausreichend berücksichtigt sind:

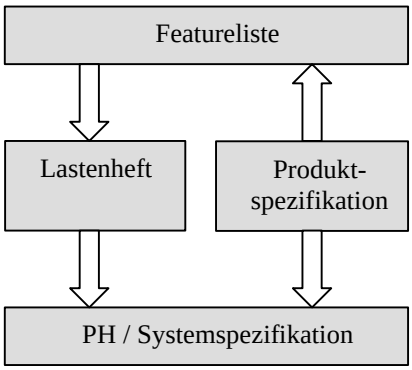
- Der industrielle Produktentwicklungsprozeß stellt hohe Qualitätsansprüche an alle Arbeitsprodukte (Artefakte) – auch hinsichtlich der Effizienz ihrer Verwendung bei hoher Änderungsfrequenz der Anforderungen bei gleichzeitiger Wahrung der Traceability.
- Der Komplexitätsgrad des aus Hardware, Software und ggf. auch aus mechanischen Elementen bestehenden Systems kann recht hoch sein; im Automotive-Bereich steigt er gerade *im Detailbereich* stetig an.
- Das Vorgehen in der Entwicklung von HW, SW und Mechanik ist jeweils durchaus unterschiedlich, und es werden verschiedene Arten von Entwurfswerkzeugen verwendet, die alle möglichst an das Requirements Management-Werkzeug angebunden sein sollten.
- Mitarbeiter, die keine ausgebildeten System- oder Anforderungsanalytiker sind, werden oft temporär mit der Erstellung von Systemspezifikationen betraut.

Die beschriebene Methode findet in allen ihren Bestandteilen bereits heute in mehreren Projekten Anwendung. Sie ist weniger für Einzelentwicklungen konzipiert, als für das Produktgeschäft mit hohen Stückzahlen.

3 Systemspezifikation und Prozeß

Die *Systemspezifikation* ist derjenige Teil des Pflichtenhefts, welcher alle direkten Anforderungen an das herzustellende System (*Zielsystem*) beinhaltet, während sich weitere Anforderungen an den *Prozeß* der Entwicklung und Fertigung richten und somit einen *Rahmen* vorgeben. (Nebenbei: Diese Unterscheidung entspricht *nicht* der von funktionalen und nichtfunktionalen Anforderungen.)

Um das Spezifikationsartefakt nicht bei jedem Projekt neu erstellen zu müssen, greift man vielerorts einfach auf die des unmittelbaren Vorgängerprojekts mit gleichem Produktfokus zurück. Effizienter wird die Wiederverwendung jedoch, wenn man eine dafür optimierte *Produkt*spezifikation verwendet, an der *projekt*spezifische Änderungen *als solche* vorgenommen werden. Extrakt jener Produktdefinition ist eine allgemeine *Featureliste*, die in der jeweiligen Angebotsphase projektspezifisch angepaßt wird und der das Lastenheft entsprechen muß.



4 Aufbau der Systemspezifikation

Das Ziel ist die also die Erstellung eines Systems, das (1.) eine bestimmte Funktionalität hat, (2.) bestimmte konkrete Eigenschaften und eine bestimmte Beschaffenheit aufweist und (3.) spezifizierte Schnittstellengrößen besitzt. Diese einfache Definition kann, wie sich zeigt, leicht durch Trennung der drei Aspekte voneinander in die Struktur der Spezifikation übertragen werden.

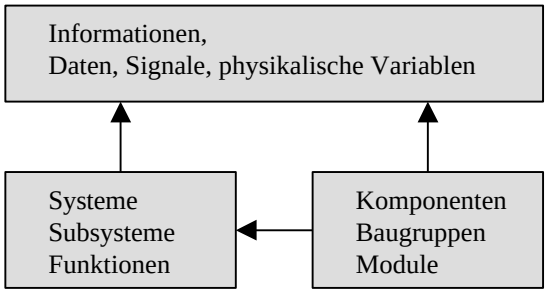
Eine solche Aufteilung in *mehrere Sichten* wurde bereits in vielen Publikationen dargelegt und empfohlen [1; 2; 3]. Der folgende Ansatz, der im Prinzip der klassischen

Unterscheidung von Analyse und Design bei der Software-Entwicklung entspricht, ist dabei ein Minimum:

1. **Verhaltensbeschreibung** (Problembereich)
2. **Architekturbeschreibung** (Lösungsbereich)
3. **Größenkatalog** (Problem- und Lösungsbereich)

Die *Verhaltensbeschreibung* enthält Anforderungen an die Funktionalität des Zielsystems und ist hierarchisch in Systeme, Subsysteme und Funktionen gegliedert, die ihrerseits Schnittstellen und Zustände aufweisen.

Die Elemente der *Architekturbeschreibung* hingegen (Komponenten, Module, Baugruppen u. dgl. samt ihren Schnittstellen und Eigenschaften) beziehen sich einerseits auf *ihresgleichen* und geben so die grobe Struktur des realen, physikalischen Zielsystems vor, und referenzieren andererseits diejenigen Elemente der Verhaltensbeschreibung, welche sie realisieren. Anforderungen, die sich auf den *technischen Entwurf* des Zielsystems richten, finden hier ihren Ort.



Die in diesen beiden Beschreibungen verwendeten *Größen* (Informationen, Daten, Signale, physikalische Variablen) werden in einem *Katalog* in eindeutiger Weise getrennt definiert; von den Schnittstellenbeschreibungen aus wird (unter Verwendung von Vereinfachungsregeln) auf die jeweils verwendeten Definitionen verwiesen.

5 Vollständigkeit

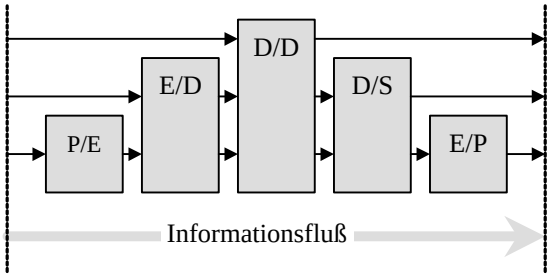
Selbstverständlich findet auch in bezug auf die Systemspezifikation die allgemeine Forderung nach *Vollständigkeit der Anforderungen* Anwendung. Prinzipiell ist sie dann erfüllt, wenn sowohl die Kundenanforderungen des Lastenhefts, als auch die Basisanforderungen, die ein solches *Modell* zu einem „runden Ganzen“ machen, modelliert sind. Das Ziel kann jedoch keine *absolut* vollständige Systemspezifikation sein, denn das würde bedeuten, die Arbeit des funktionalen und technischen Entwurfs vorwegzunehmen.

Das Modell ist lediglich *temporär abgeschlossen*, wenn sich, was Basisanforderungen betrifft, auf ein Minimum beschränkt wurde. Das ist dann der Fall, wenn es bestimmten Kriterien genügt, welche formal festlegen, ob und wann es *relativ* vollständig ist. Zu diesen Regeln gehört beispielsweise, daß alle irgend erwähnte Elemente vor ihrer Verwendung *definiert* werden müssen und was jeweils zu einer Definition gehört.

Dies setzt ein gewisses Maß an Genauigkeit und Einheitlichkeit der Modellbildung voraus. Um es zu gewährleisten, bieten sich zwei wichtige Hilfsmittel an, die gleichzeitig oder unabhängig voneinander eingesetzt werden können und die beide mit der festgelegten Kapitelstruktur der Systemspezifikation korrelieren. Dabei handelt es sich (a) um eine Vorlage für die *einheitliche Identifikation* von Subsystemen und Funktionen (*Systemschablone*), und (b) um *Textschablonen* für die einheitliche und exakte Formulierung einzelner Anforderungen.

6 Systemschablone

Eingebettete Systeme weisen eine bestimmte Grundstruktur auf. Ihren Kern bildet natürlich die *Datenverarbeitung* (in der Graphik D/D als Verarbeitung von Eingabe-Daten zu Ausgabe-Daten). Sie wird eingabeseitig ergänzt durch die Umwandlung von elektrischen Signalen (von Konvertern etc.) zu Daten (E/D) und ggf. auch, dem vorgeschaltet, von physikalischen Größen (bspw. mittels Sensoren) zu elektrischen Signalen (P/E). Ausgabeseitig sind ergänzende Funktionen spiegelbildlich anzusetzen (bspw. Aktoren). Dabei bildet die *Systemgrenze* die Schnittstelle zu allen angrenzenden Systemen.



Ein zu spezifizierendes System setzt sich demzufolge aus einigen Instanzen von vordefinierten Elementklassen zusammen. Diese bilden zusammen als mehrschichtiges Grundmodell eine „Systemschablone“ und sind darin einer der folgenden Gruppen zugehörig:

- **Systemschnittstellen** (z.B. natürliche Umgebung, Benutzer, mechanisches Stellglied)
- **Subsysteme und Funktionsgruppen** (z.B. Sensorik, Initialisierung, Ausgabekontrolle, Filter, Remote Control, Fehlererkennung)
- **Informationsflüsse** (z.B. Benutzerkommando, Sensorinformation, erkannter Fehler)

Eine detaillierte Darstellung samt Hinweisen zur praktischen Verwendung findet sich in [4]. Die dort gegebene Systematik läßt sich leicht auf die Analyse eingebetteter Systeme übertragen.

7 Verwendung von Textschablonen

Die natürlichsprachige Notation der Anforderungen mit vorformulierten Mustern erbringt einen hohen Grad der

Formalisierung, die nebst weiteren Vorteilen die Möglichkeit der *automatisierten Überprüfung* der Systemspezifikation mit sich bringt. Im Bereich der *Sicherheitsfachsprachen* finden sie bereits seit längerem Anwendung.

Das folgende Beispiel für ein Anforderungsmuster enthält sowohl *Alternativen* („Wenn“ oder aber „Sobald“) als auch *optionale Bestandteile* („und ... erfüllt ist“; „Zustand“).

```
(Wenn | Sobald) im [Zustand] „<Zu-
stand>“ des Systems „<System>“ das
Ereignis „<Ereignis>“ eintritt [und
die Bedingung „<Bedingung>“ erfüllt
ist], geht es in den [Zustand] „<Zu-
stand>“ über.
```

Die konkrete Anforderung wird formuliert, indem man die gewünschte Formulierung auswählt und dann die mit spitzen Klammern markierten Terme („<System>“, „<Ereignis>“, evtl. auch „<Bedingung>“ und „<Zustand>“) durch die konkreten Bezeichnungen und Ausdrücke ersetzt. Ein Beispiel:

```
Wenn im Betriebszustand des Systems
“Kurvenlicht“ das Ereignis „Fahrzeug-
geschwindigkeit fällt unter  Mindest-
geschwindigkeit Kurvenlicht“ ein-
tritt, geht es in den Zustand „Rück-
lauf“ über.
```

Folgende Elemente können also mittels der Schablone im Rahmen einer *automatisierten Verifikation* identifiziert und auf Einhaltung der zugrundegelegten Konsistenzregeln untersucht werden:

System „Kurvenlicht“ (hier nur Verwendung)
Ereignis „Fahrzeuggeschwindigkeit fällt unter Mindestgeschwindigkeit Kurvenlicht“
Systemzustände „Kurvenlicht“: „Betriebszustand“, „Rücklauf“

Praktische Erfahrungen in der Anwendung der Methode werden in [5] geschildert.

8 Realisierung mit *DOORS*

In der Automobilindustrie ist das RM-Werkzeug *DOORS* weit verbreitet. Es ist ursprünglich für die natürlichsprachige Anforderungsdokumentation entwickelt worden und daher für Ansätze wie den hier vorgestellten geeignet [6]. Dieser läßt sich im Kern wie folgt realisieren:

Verhaltens-, Architekturbeschreibung und Größenkatalog werden jeweils als ein formales Modul angelegt. Ein einziges Linkmodul (*Internes Mapping*) verwaltet die gegenseitigen Verknüpfungen der Anforderungsobjekte innerhalb der drei Module.

Die Lastenheftmodule werden über ein weiteres Linkmodul (*Mapping zur Systemspezifikation*) angebunden. Um auch eine Verknüpfung mit den Elementen in Entwurfstools zu ermöglichen, werden *Stellvertretermodule* benötigt, in die hinein die Entwurfselemente der Entwurfstools (etwa SW-Module, mechanische Baugruppen, HW-Funktionsblöcke) gespiegelt (*Mapping zum Entwurf*) und dann automatisiert gegenüber Änderungen konsistent gehalten werden.

9 Toolkette

Damit die angestrebte Durchgängigkeit der Anforderungen erreicht werden kann, müssen die eingesetzten Werkzeuge eine *Kette* bilden, d.h. über ihre Schnittstellen verlustfrei und möglichst ohne manuelles Eingreifen miteinander verbunden sein. Beispielsweise wird das Lastenheft in *Word* erstellt, das Pflichtenheft mit Text und Graphiken in *DOORS*, und der funktionale und technische Softwareentwurf mithilfe von Strukturierter Analyse / Strukturiertem Design (SA/SD) in *INNOVATOR Function*. Leider sind hingegen die Schnittstellen von *Konstruktionswerkzeugen* und auch der *Hardwareentwicklungs-Tools* zur Anbindung von RM-Werkzeugen noch vergleichsweise unterentwickelt. Gleichwohl erfüllt bereits heute der zunächst nur in bezug auf die *Software*-Anforderungen *durchgängige* Workflow seine Zielvorgabe mit Bravour.

10 Literatur

[1] Liggesmeyer; Rombach: Software Engineering eingebetteter Systeme. Grundlagen - Methodik - Anwendungen. München 2005. S. 107ff, S. 148ff.

[2] Greeenfield; Short: Software Factories. Assembling Applications with Patterns, Models, Frameworks, and Tools. Indianapolis 2004. S. 265ff.

[3] Geisberger; Wußmann: Requirements eingebetteter Systeme. (Softwaretechnik-Trends, 23 / 1)

[4] Kapeller: Strukturierte Spezifikation von elektronischen Kfz-Steuergeräten. (MID-Whitepaper, in Vorb.)

[5] Kapeller; Krause: Eingebettete Elektroniksysteme natürlichsprachig modellieren. (Zeitschriftenartikel, in Vorb.)

[6] v. d. Beeck; Braun; Rappl; Schröder: Modellbasierte Softwareentwicklung für automobilspezifische Steuergerätenetzwerke. (www.forsoft.de/automotive/BadenBaden 01.pdf)