Ein Framework zur automatisierten Ermittlung der Modellqualität bei eingebetteten Systemen

Jan Scheible (jan.scheible@daimler.com)
Daimler AG - Group Research and Advanced Engineering

Abstract: Die Modelle, die in der Automobilindustrie zur Modellbasierten Softwareentwicklung von eingebetteten Systemen verwendet werden, werden immer größer und komplexer. Zwar findet die Modellbasierte Entwicklung auf einer höheren Abstraktionsebene als die herkömmliche, handcodierte Softwareentwicklung statt, jedoch resultieren im Modell gemachte Fehler direkt in Fehler im Quellcode. Dieses Research Abstract stellt einen Ansatz zur automatisierten Ermittlung der Modellqualität von Matlab Simulink-Modellen vor. Dadurch wird nicht nur eine detaillierte Aussage über die momentane Qualität der Modelle ermöglicht, sondern auch eine Verbesserung der Modellqualität durch Befolgen der abgeleiteten Handlungsempfehlungen erreicht.

1 Problemstellung

In der Automobilindustrie wird in den letzten Jahren verstärkt auf die Modellbasierte Softwareentwicklung von eingebetteten Systemen gesetzt (vergleiche [KCFG05]). Die Modelle nehmen dabei den Stellenwert ein, den zuvor der handgeschriebene Quellcode eingenommen hat. Sie sind das zentrale Entwicklungsartefakt und werden (über die Codegenerierung und Compilierung) direkt auf den eingebetteten Systemen ausgeführt. Ein Werkzeug zur Modellbasierten Entwicklung ist z. B. Matlab Simulink mit dSPACE Targetlink. Die Modelle bestehen aus Blöcken, welche durch Signale verbunden werden. Mittels Subsystemen werden die Modelle hierarchisch gegliedert.

Der Umfang der Modelle steigt stetig. So hat momentan ein großes Modell typischerweise ca. 15.000 Blöcke, 700 Subsysteme und eine Subsystemhierarchie mit 16 Ebenen. Außerdem geben Werkzeuge wie Matlab Simulink den Entwicklern sehr viele Freiheiten. Deshalb wird in [CD06] vorgeschlagen, den erlaubten Sprachumfang für sicherheitsrelevante Systeme einzuschränken. Sowohl die großen Umfänge als auch die Freiheiten der Entwickler, die sich aus der herkömmlichen und zusätzlich aus der Modellbasierten Softwareentwicklung ergeben, führen zu vielen potentiellen Fehlerquellen.

Daher ist der Ansatz dieses Beitrags die automatisierte Ermittlung der *Modellqualität* anhand von Kriterien, welche qualitativ hochwertige Modelle auszeichnen. Die Qualitätskriterien werden dabei in einem *Qualitätsmodell* strukturiert. Die Untersuchung der Modelle auf die Erfüllung der Qualitätskriterien wird mit Hilfe von *Modellmetriken* durchgeführt. Aus dem Qualitätsmodell und den Informationen der Modellmetriken wird die *Modellqualitätsbewertung* gebildet. Falls ein Modell nicht die gewünschte Qualität hat, kann durch

das Befolgen der *Handlungsempfehlungen* die Qualität des Modells verbessert werden. Um eine Erweiterbarkeit und projektspezifische Anpassbarkeit zu erreichen, wird im Rahmen dieser Arbeit schließlich ein *Framework* definiert.

Unterstützt wird der Ansatz z. B. durch Fieber et. al [FHR08]. Sie beschreiben die automatisierte Ermittlung der Modellqualität als nächsten Schritt, da nur so der große Umfang bewältigt werden kann. Da Modelle das zentrale Artefakt der Modellbasierten Softwareentwicklung sind, wird außerdem in [FHR08] und [MA07] die Hypothese unterstützt, dass die Softwarequalität direkt von der Modellqualität abhängig ist. Deshalb ist davon auszugehen, dass qualitativ hochwertige Modelle auch zu qualitativ hochwertiger Software führen. Des Weiteren ermöglicht der Ansatz der automatisierten Qualitätsermittlung sowohl das Ermitteln eines aktuellen Ist-Stands, um die aktuelle Modellqualität einschätzen zu können, als auch eine abschließende Qualitätsbewertung bei Abgabe eines Modells.

2 Lösungsidee

Modellqualität Die Qualität eines Modells wird bewertet, indem geprüft wird, ob das Modell Kriterien mit einem positiven Einfluss auf gewünschte Faktoren wie z. B. Wartbarkeit, Lesbarkeit oder Robustheit erfüllt. Daher hat ein Modell, das alle Kriterien erfüllt, eine hohe Modellqualität. Für den Ansatz dieses Beitrags sind alle Kriterien von Interesse, die einen Rückschluss auf gewünschte Faktoren und somit die Modellqualität zulassen.

Qualitätsmodell Zur Strukturierung der Kriterien wird ein Qualitätsmodell mit einem Aufbau ähnlich [CM78] verwendet. Dazu werden Faktoren definiert, welche Einfluss auf die Modellqualität haben. Die Erfüllung der Faktoren wird an Hand von den Kriterien festgemacht. Inwieweit die Kriterien erfüllt sind, wird mit Hilfe von Metriken gemessen. Zusätzlich besitzt jedes Kriterium eine Beschreibung, wie es sich auf seinen übergeordneten Qualitätsfaktor auswirkt. Dadurch wird der Zusammenhang zwischen Faktoren und Kriterien explizit dokumentiert. Das Qualitätsmodell wird im Folgenden FCM-Modell (Factors-Criteria-Metrics-Modell) genannt.

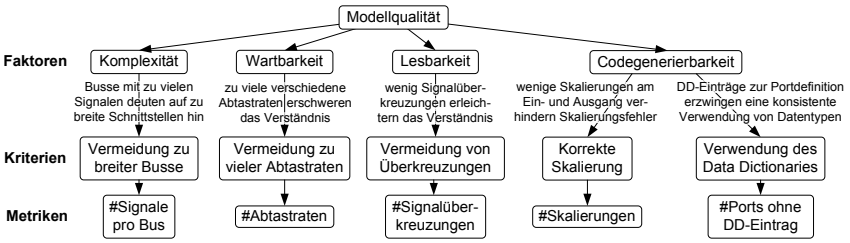

Abbildung 1: Auszug aus dem FCM-Modell

Abbildung 1 zeigt einen Auszug aus dem FCM-Modell zur Modellqualitätsbewertung. Bisher enthält das FCM-Modell 10 Faktoren, 26 Kriterien und 43 Metriken. Die Fakto-

ren ähneln denen in [CM78], sind aber um neue, modellspezifische Faktoren wie z. B. Codegenerierbarkeit (vergleiche Abbildung 1) erweitert worden.

Wenn beispielsweise das Kriterium „Verwendung des Data Dictionaries“ verletzt wird, hat das einen negativen Einfluss auf den Faktor Codegenerierbarkeit, da z. B. der Datentyp der betroffenen Ports nicht mehr zentral an einer Stelle änderbar ist. Dies ist ein Indikator für unsaubere Modellierung, führt jedoch nicht zwingend zu Fehlern. Daher kann solch eine Verletzung einerseits als Anlass zur Korrektur bei einer Ist-Stand-Ermittlung dienen, anderseits kann sie unter Umständen, z. B. aus Zeitmangel ignoriert werden.

Modellmetriken Im Gegensatz zur Ermittlung der Softwarequalität in [CM78] messen die Metriken bei der Ermittlung der Modellqualität nicht auf dem Quellcode, sondern auf dem Simulink-Modell. Die Aufgabe der Metriken ist, die Erfüllung von Qualitätskriterien zu messen, daher wird folgende Definition aus [BBL76] übernommen: „The term 'metric' is defined as a measure of extent or degree to which a product (...) possesses and exhibits a certain (quality) characteristic.“

Eingeordnet nach den Eigenschaften von [Mil99] sind die Metriken sowohl *direkt* als auch *indirekt*. Direkte Metriken messen auf dem Modell, indirekte verwenden eine externe Datenquelle, welche aufbereitete Informationen über das Modell bereitstellt. Ein Beispiel für eine externe Datenquelle sind die Ergebnisse eines Prüfers für Modellierungsrichtlinien wie MXAM von MES, welche Rückschlüsse auf die Standardkonformität eines Modells zulassen. Die Metriken können außerdem *primitiv* (z. B. „#Blöcke“, um abzuschätzen, ob der Umfang des Modells gerechtfertigt ist) oder *berechnet* (z. B. „#Blöcke pro Subsystem“, um auf zu große Subsysteme schließen zu können) sein. Zusätzlich müssen die Messwerte der Metriken mindestens auf einer Ordinalskala liegen.

Zusätzlich gibt jede Metrik einen erlaubten Minimal- und Maximalwert für ihre Messergebnisse vor. Sie messen entweder einen Messwert pro Subsystem oder einen Messwert für das gesamte Modell. Jede Metrik hat einen Typ, welcher bestimmt, wie die Messwerte erhoben werden. Für Metriken vom Typ *atomar* wird ein einzelner Messwert pro Modell gemessen. Bei Metriken vom Typ *akkumulierend* oder *arithmetisches Mittel* hingegen wird ein Messwert pro Subsystem gemessen und aufaddiert. Beim Typ arithmetisches Mittel wird zusätzlich noch durch die Anzahl der Subsysteme geteilt. Schließlich besitzt jede Metrik eine Priorität (1 = normal, 2 = hoch).

Modellqualitätsbewertung Die Messwerte der Metriken und das FCM-Modell bilden die Grundlage der Modellqualitätsbewertung. In [CM78] werden dazu alle Messwerte der Metriken normalisiert und dann eine gewichtete Summe für jeden Qualitätsfaktor berechnet.

In diesem Beitrag werden die Messwerte dadurch normiert, dass sie in Bezug zur Problemgröße gesetzt werden. Wenn beispielsweise der Umfang der Spezifikation als Problemgröße verwendet wird, dann kann der Bezug z. B. als kontinuierliche Funktion wie $Messwert_{normiert} = Messwert/Problemgröße$ formuliert oder durch die Einteilung in diskrete Klassen (z. B. kleines, mittleres oder großes System) hergestellt werden.

Zur Berechnung der endgültigen Modellqualität wird jede Metrik gewichtet. Das Gewicht

ist 0, falls der skalierte Messwert zwischen den von der Metrik vorgegebenen Grenzen liegt. Andernfalls entspricht das Gewicht der Priorität der Metrik. Pro Qualitätsfaktor werden alle Gewichte addiert und durch die Summe der Prioritäten geteilt. Das Ergebnis ist eine Prozentzahl pro Qualitätsfaktor, welche angibt, zu wie viel Prozent die Kriterien des Qualitätsfaktors nicht erfüllt sind. Somit besteht eine Modellqualitätsbewertung aus einem n-Tupel, wobei n die Anzahl der Qualitätsfaktoren ist.

Handlungsempfehlung Um die Modellqualität auf Basis der Modellqualitätsbewertung verbessern zu können, besitzt jede Metrik eine Handlungsempfehlung. Die Handlungsempfehlung gibt an, wie die Modellqualität verbessert werden kann, wenn der Messwert einer Metrik außerhalb der vorgegebenen Grenzen liegt. Bei indirekten Metriken, die externe Datenquellen verwenden, wird zur detaillierten Auswertung auf die externe Datenquelle verwiesen. Die gesamte Menge der Handlungsempfehlungen bildet eine Wissensdatenbank.

Framework Das FCM-Modell ist nicht monolithisch, sondern wird durch verschiedene *Qualitätsaspekte* gebildet, die in einem Framework verwaltet werden. Dadurch können einzelne Qualitätsaspekte an- oder abgeschaltet sowie neue Qualitätsaspekte hinzugefügt werden. Ein Qualitätsaspekt besteht aus einem Teilbaum mit Faktoren, Kriterien und Metriken. Qualitätsaspekte sind z. B. die Codegenerierung mit Targetlink oder die Einhaltung von Modellierungsrichtlinien. Abbildung 2 zeigt, wie Qualitätsaspekte zu einem FCM-Modell vereinigt werden. Bei der Vereinigung werden Knoten mit gleichen Namen als identisch angesehen.

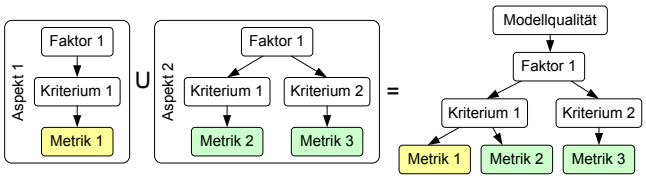

Abbildung 2: Schematische Darstellung der Vereinigung zweier Aspekte

Prototyp Ein in Java implementierter Prototyp setzt bereits das Framework um und enthält erste einfache Metriken. Die Anbindung an Matlab Simulink wird durch die Nutzung des Simulink Parsers der TUM realisiert. Auch die Visualisierung der Messwerte mit ihren Minimal- und Maximalwerten in einem Kiviat-Diagramm (vergleiche [Dra96]) und eine Visualisierung der Modellqualitätsbewertung wurde bereits prototypisch umgesetzt.

3 Verwandte Arbeiten

Fey und Stürmer führen in [FS07] folgende Maßnahmen zur Qualitätssicherung von Simulink-Modellen auf: Modellbasiertes Testen, Modell-Review, Code-Review und stati-

sche Code-Analyse. Der Ansatz dieses Beitrags kann als statische Modellanalyse eingeordnet werden, da die Analyse direkt auf dem Modell stattfindet und das Modell dabei nicht ausgeführt wird. Zusätzlich zu den aufgeführten Maßnahmen hat sich das Aufstellen und Prüfen von Modellierungsrichtlinien als Standard etabliert [SDP08, WDTG09]. Allerdings lassen sich zum einen nur ein Teil der Modellierungsrichtlinien automatisiert prüfen und zum anderen können viele Qualitätskriterien, wie z. B. eine sinnvolle Aufteilung in Subsysteme, nicht als formale Richtlinien definiert werden. In der Automobilindustrie sind die MAAB-Richtlinien [MAA07] weit verbreitetet. Sie geben herstellerunabhängige Regeln vor, um den Austausch von Modellen zu vereinfachen und die Anzahl der potentiellen Fehler zu reduzieren.

Deissenboeck et al. [DWP+07] stellen ein Qualitätsmodell für die Wartbarkeit auf. Sie modellieren nicht nur die Qualitätskriterien, sondern auch ihren Einfluss auf die nötigen Aktivitäten. In einer Fallstudie erweitern sie ihr Qualitätsmodell um Simulink-spezifische Kriterien und Aktivitäten. Automatisierte Qualitätsbewertung steht nicht in ihrem Fokus und das Qualitätsmodell konzentriert sich auf die Wartbarkeit. Die Aktivitäten sind vergleichbar mit den Handlungsempfehlungen im Ansatz dieses Beitrags.

Bobkowska stellt in [Bob09] einen generischen Ansatz vor, der Qualitätskriterien und die Wahl der Methode zur Bewertung berücksichtigt. In einer Fallstudie zur Vorhersage der Qualität eines Softwareprojekts, welches UML verwendet, wird von ihr ein FCM-Modell nach Cavano und McCall [CM78] verwendet.

Ein mit dem Ansatz dieses Beitrags vergleichbares Framework für UML-Modelle beschreiben Lange und Chaudron [LC05]. Sie unterscheiden zwischen Entwicklung und Wartung, da bei UML-Modellen in den jeweiligen Phasen andere Qualitätskriterien relevant sind. Sie verwenden zum Großteil OO-Metriken und nur einige modellspezifische Metriken wie z. B. Anzahl der überkreuzenden Linien.

4 Zusammenfassung und Ausblick

In diesem Research Abstract wurde ein Ansatz zur automatisierten Qualitätsbewertung anhand von Qualitätskriterien vorgestellt. Die Qualitätskriterien wurden in einem Qualitätsmodell in den Kontext der Modellqualität gesetzt. Die Erfüllung der Kriterien wird mit Hilfe von Modellmetriken gemessen. Handlungsempfehlungen für nicht erfüllte Qualitätskriterien bilden eine Wissensdatenbank zur qualitätsorientierten Modellierung. Abschließend wurde gezeigt, wie verschiedene Qualitätsaspekte ein Framework zur projektspezifischen Anpassbarkeit und Erweiterbarkeit bilden.

Der nächste Schritt besteht vor allem in der Vervollständigung des Prototyps, welcher zur Evaluierung des bestehenden FCM-Modells verwendet werden soll. Mit Hilfe der Evaluierungs-Ergebnisse soll dann geprüft werden, in welcher Hinsicht das Qualitätsmodell noch angepasst werden muss. Ziel ist, ein Qualitätsmodell für eine robuste Ermittlung der Modellqualität zu finden und zutreffende Handlungsempfehlungen zur Verbesserung der Modelle geben zu können.

Literatur

[BBL76] B. W. Boehm, J. R. Brown und M. Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering*, Seiten 592–605, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

[Bob09] A. E. Bobkowska. *Model-driven software development*, Kapitel Integrating quality criteria and methods of evaluation for software models, Seiten 78–93. Information Science Reference, 2009.

[CD06] M. Conrad und H. Dörr. Einsatz von Modell-basierten Entwicklungstechniken in sicherheitsrelevanten Anwendungen: Herausforderungen und Lösungsansätze. In *Proceedings Dagstuhl Workshop Modellbasierte Entwicklung eingebetteter Systeme*, 2006.

[CM78] J. P. Cavano und J. A. McCall. A framework for the measurement of software quality. In *Proceedings of the software quality assurance workshop on Functional and performance issues*, Seiten 133–139, 1978.

[Dra96] T. Drake. Measuring Software Quality: A Case Study. *Computer, IEEE Computer Society Press*, 29(11):78–87, 1996.

[DWP+07] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert und J.-F. Girard. An Activity-Based Quality Model for Maintainability. In *Proc. of the 23rd IEEE International Conference on Software Maintenance*. IEEE CS Press, 2007.

[FHR08] F. Fieber, M. Huhn und B. Rumpe. Modellqualität als Indikator für Softwarequalität: eine Taxonomie. *Informatik-Spektrum*, 31(5):408–424, October 2008.

[FS07] I. Fey und I. Stürmer. Quality Assurance Methods for Model-based Development: A Survey and Assessment. *SAE SP*, NUMB 2126:69–76, 2007.

[KCFG05] T. Klein, M. Conrad, I. Fey und M. Grochtmann. Modellbasierte Entwicklung eingebetteter Fahrzeugsoftware bei DaimlerChrysler. *Inform., Forsch. Entwickl.*, 20(1-2):3–10, 2005.

[LC05] C. F. J. Lange und M. R. V. Chaudron. Managing Model Quality in UML-Based Software Development. In *Proc. of the 13th IEEE International Workshop on Software Technology and Engineering Practice*, Seiten 7–16. IEEE Computer Society, 2005.

[MA07] P. Mohagheghi und J. Aagedal. Evaluating Quality in Model-Driven Engineering. In *Proceedings of the International Workshop on Modeling in Software Engineering*, Seite 6. IEEE Computer Society, 2007.

[MAA07] MAAB. The MathWorks Automotive Advisory Board, "Controller Style Guidelines for Production Intent Using MATLAB, Simulink and Stateflow", Version 2.1, 2007.

[Mil99] E. E. Mills. Metrics in the software engineering curriculum. *Ann. Softw. Eng.*, 6(1-4):181–200, 1999.

[SDP08] I. Stürmer, C. Dziobek und H. Pohlheim. Modeling Guidelines and Model Analysis Tools in Embedded Automotive Software Development. In *Proceedings Dagstuhl Seminar Modellbasierte Entwicklung eingebetteter Systeme*, 2008.

[WDTG09] S. Wagner, F. Deissenboeck, S. Teuchert und J.-F. Girard. *Model-driven software development*, Kapitel Assuring Maintainability in Model-Driven Development of Embedded Systems, Seiten 353–373. Information Science Reference, 2009.