Software-Entwicklung und Zertifizierung im Umfeld sicherheitskritischer und hochverfügbarer Systeme: Bedeutung modellbasierter und formaler Ansätze für effiziente Entwicklung und Zertifizierung

Hardi Hungar[1], Erwin Reyzl[2]

[1] OFFIS, Bereich Sicherheitskritische Systeme,
Escherweg 2, 26121 Oldenburg
hungar@offis.de

[2] Siemens AG, Corporate Technology, Bereich Software&Engineering,
Otto-Hahn-Ring 6, 81730 München
erwin.reyzl@siemens.com

Abstract: Angesichts der zunehmenden Verwendung von Software bei der Realisierung sicherheitskritischer und hochverfügbarer Systeme wächst die Bedeutung der Nachweisbarkeit nicht-funktionaler Eigenschaften wie der Funktionssicherheit und der Zuverlässigkeit – sowohl bei der Entwicklung wie auch bei der Zertifizierung derartiger Systeme. Dem daraus resultierenden Anspruch an Nachvollziehbarkeit, an Rigorosität von Tests und anderen Prüfungen einerseits stehen innovative Trends der Software-Entwicklung wie modellbasierte Entwicklung, formale Ansätze für Design und Verifikation, Komponenten- und Plattformorientierung, Code-Generierung und Test-Automatisierung gegenüber, die ohne spezifische Adaptionen diesem Anspruch häufig nicht genügen. Hier viel versprechende Techniken auszuwählen, Einsatzperspektiven zu eruieren und dabei den Abgleich zwischen Forschungsrichtungen und industriellem Bedarf zu suchen soll Thema des Workshops sein.

1 Ausgangssituation

Software, insbesondere eingebettete Software, spielt eine zunehmend tragende Rolle im Umfeld sicherheitskritischer und hochverfügbarer Systeme. Traditionell beheimatet in der Luft- und Raumfahrt sowie dem militärischen Bereich, steigt die Bedeutung derartiger Systeme zunehmend auch im zivilen Bereich, sei es in der Verkehrstechnik, der Industrieautomatisierung, der Medizintechnik, der Kraftwerks- und Energietechnik [Lev95]. Die Erfüllung und der Nachweis nicht-funktionaler Eigenschaften wie Funktionssicherheit, der Zuverlässigkeit und Verfügbarkeit werden bestimmend in der Entwicklung, aber auch bei der Zertifizierung derartiger Systeme. Relevante Standards sind der allgemeine Standard IEC 61508, aber auch domänenspezifische Normen wie EN50128, DO178, ISO WD 26262 und FDA-Vorschriften (siehe etwa [BB+06]). Besonderheiten dabei, die sich in diesem Zusammenhang mit dem zunehmenden Softwareanteil ergeben, gilt das besondere Interesse dieses Workshops.

Kostentreiber liegen erwiesenermaßen in Aktivitäten zum Nachweis durch Test, Verifikation und Validierung, aber auch im wachsenden Dokumentationsbedarf zur Nachvollziehbarkeit von Änderungen und ihrer Auswirkungen. Effizienten Ansätze zur Verbesserung von Nachweisen und zur Vereinfachung der Nachvollziehbarkeit wird gesteigerte Bedeutung zukommen. Von speziellem Interesse sind einerseits bewährte Lösungen und Erfahrungen aus dem industriellen Umfeld, andererseits Ansätze aus der akademischen Forschung mit realer Anwendungsperspektive.

2 Trends im Software-Engineering

In der Software-Entwicklung haben sich in den letzten Jahren verschiedene Ansätze und Techniken weit verbreitet. Dazu gehören modellbasierte Entwicklung, formale Ansätze für Design und Verifikation, Komponenten- und Plattformorientierung, Code-Generierung und Test-Automatisierung [Lyu96]. Im folgenden sprechen wir einige dieser Ansätze kurz an und weisen jeweils auf hier relevante Fragestellungen hin.

2.1 Modellbasierte Entwicklung

Die Erstellung von Modellen in frühen Phasen der Entwicklung hilft, Anforderungen klar, konsistent und umfassend zu formulieren und Konzepte der Realisierung greif- und damit überprüfbar zu machen. Besonders, aber nicht nur, bei objektorientierter Umsetzung haben modellbasierte Methoden eine mittlerweile recht große Verbreitung erlangt – hier sei nur die viel verwendete UML [UML] genannt. Im Zusammenhang mit standardisierten Anforderungen an den Entwurfsprozess stellen sich verschiedene Fragen an die Belastbarkeit des Ansatzes, etwa:

1. Können die Spezifikationsmodelle auf ein ausreichend präzises Niveau gehoben werden, so dass sie textuelle Beschreibungen mit Vorteilen für spätere Validationsschritte ersetzen?

2. Lassen sich (modellbasiert) Lösungskonzepte in einem verbindlichen Sinn überprüfen, bzw., unter welchen Voraussetzungen wäre dies möglich?

2.2 Formale Methoden

Das Versprechen, durch Einsatz formaler Methoden zu sicheren und umfassenden Aussagen über das Verhalten von Entwicklungsartefakten zu gelangen, ist oft gemacht worden. Mittlerweile sind auch in der Praxis verschiedenste Ansätze aus diesem Bereich erfolgreich umgesetzt worden. Gleichwohl ist der Anwendungsbereich bisher noch schmal und oft lückenhaft, und durchgängige Anwendungen sind selten zu finden [HB+07]. Wir listen einige Beobachtungen auf:

1. Um den Erfordernissen einer Norm zu genügen, müssen in der Regel die Verifikationswerkzeuge sowie alle Übersetzer oder Codegeneratoren qualifiziert sein.

2. Eine vollständige formale Analyse (etwa Modelchecken) ist nach wie vor zumeist nur bei sehr kleinen Systemen möglich. An welcher Stelle lassen sich solche Werkzeuge heute erfolgreich einsetzen, oder welche Änderungen, beispielsweise an Systementwurfssprachen, würden einen weitergehenden Einsatz erlauben?

2.3 Testgenerierung

Heutzutage basieren die meisten Verifikationen und Validationen auf ausführlichen Tests, die nach mehreren Gesichtspunkten durchzuführen sind, etwa Tests auf Basis von Fehlererwartung und Fehlereinstreuung, Berücksichtigung von Grenzwertanalysen, strukturabhängige Tests etc. Um dem hohen Aufwand der Erstellung von Sätzen von Testmustern, welche den jeweiligen Erfordernissen genügen, abzuhelfen, können etwa Verfahren der automatischen Testgenerierung in Erwägung gezogen werden. Zu beachten sind dabei u.a.:

1. Grundlage der Testgenerierung muss eine formale Beschreibung des Testobjektes sein. Neben Programmen in späten Entwicklungsphasen kommen in früheren Phasen Modelle in Frage.

2. Das Erreichen der Anforderungen muss separat nachprüfbar sein, oder das Werkzeug zur Testgenerierung ist in dieser Hinsicht zu qualifizieren.

2.4 Codegenerierung

Der Einsatz automatischer Codegenerierung, bei der etwa aus Modellen direkt Zielcode gewonnen wird, entspricht der Einführung einer weiteren, höheren Ebene der Programmierung, für die sich Fragen der Codierungsregeln und Übersetzerkorrektheit neu stellen. Erstrebenswert wäre es, für Verifikation und andere Entwurfspflichten nicht den erzeugten Code, sondern die Quelle der Generierung heranziehen zu können. Das aber erfordert aber in der Regel die Ergänzung der etablierten Normen auf bisher nicht erfasste Bereiche.

2.5 Weitere Ansätze

Die oben angeführten Ansätze und Techniken stellen nur einen Ausschnitt aus der großen Menge an Methoden dar, welche für eine Anwendung in Betracht kommen. Viele relevanten Fragen, etwa der Plattformorientierung, der Wiederverwendung, des Konfigurations- und Änderungsmanagements, der Anforderungsnachverfolgung und so weiter haben wir nicht genauer angesprochen. Alle diese Aspekte müssen in der Praxis berücksichtigt werden, und auch auf diesen Gebieten stellen sich alte Fragestellungen neu und verlangen nach innovativen Lösungen.

3 Zusammenfassung

Im Spannungsfeld zwischen kosteneffizienter Entwicklung, steigender Systemkomplexität, bei gleichzeitig hohen Qualitätsanforderungen und fundierter Zertifizierung besteht Bedarf nach übergreifenden Methodiken, welche auf Techniken wie die oben genannten zurückgreifen und sie mit geeigneter Werkzeugunterstützung in durchgängigen Prozessketten integrieren. Verschiedene Anforderungsprofile, teils auf Grund anwendungsspezifischer Vorschriften (Bahn, Luftfahrt, Medizin etc.), teils wegen Unterschieden in Kritikalität oder Systemumgebung, werden in Zukunft eine vielgestaltige Landschaft an Lösungen bedingen. Bei aller Vielgestaltigkeit jedoch lässt sich voraussagen, dass allein schon die Notwendigkeit, zunehmend komplexere Lösungen noch wirtschaftlich tragbar beherrschen zu müssen, den integrierten Einsatz entwurfsunterstützender Technologie in großem Umfang bedingen wird. Zum Explorieren dieser Landschaft beizutragen ist zentrales Anliegen des Workshops.

Literaturverzeichnis

[UML] Object Management Group: Unified Modeling Language, Version 2.1.1:, http://www.uml.org/#UML2.0, 2007.

[BB+06] Braband, J.; Brehmke, B.-E.; Griebel, S.; Peters, H.; Suwe, K.-H.: Die CENELEC-Normen zur funktionalen Sicherheit. Eurailpress, Hamburg, 2006.

[HB+07] Hungar, H.; Bruhns, G.; Plan, O.; Lemke, O.: „OpRail – Normenkonforme Entwicklung sicherheitsrelevanter Software unter Einsatz der UML“, Signal + Draht 09/2007, S. 6ff.

[Lev95] Leveson, Nancy G.: Safeware, System Safety and Computers, Addison-Wesley Publishing Company, Inc, 1995.

[Lyu96] Lyu, Michael R.: Handbook of Software Reliability Engineering, IEEE Computer Society Press, Mc Graw-Hill Comp, Inc, 1996.