Ergebnisse des Workshops

„Software-Entwicklung und Zertifizierung im Umfeld sicherheitskritischer und hochverfügbarer Systeme“

Hardi Hungar[1], Erwin Reyzl[2]

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

[2] Siemens AG, CT SE 1,
Otto-Hahn-Ring 6, 81730 München
erwin.reyzl@siemens.com

1 Generelles

Etwa 30 Teilnehmer setzten sich in reger Diskussion mit neun Beiträgen sowie sechs Positionsstatements zu Erfahrungen und Herausforderungen der Software-Entwicklung und Zertifizierung im Umfeld sicherheitskritischer und hochverfügbarer Systeme auseinander. Vertreten waren vor allem Verkehrs- und Automatisierungstechnik, dabei kamen die Teilnehmer aus Industrie und Forschung.

2 Vorträge

Nach der Einführung und dem Vortrag von Hr. Prof. Parnas zum Thema „Certification: What, how, and how Confidently“ wurden in drei Sitzungen die Themen „Werkzeuge und Programmierung“, „Entwurfsparadigmen“ sowie „Formale Methoden“ angesprochen.

Bei „Werkzeuge und Programmierung“ zeigte sich, wie bereits heute modellbasierte Werkzeugen mittels qualifizierter/zertifizierter Codegeneration und –übersetzung zumindest Teile der Entwicklung sicherheitskritischer Software verbessern. Mit der Spezifizierung und Standardisierung eines echtzeit- und sicherheitsfähigen Java dürfte in Zukunft auch Java neben C/C++ und Ada Anwendung finden. Die Wichtigkeit domänenspezifischer Modellierung in der Praxis der Stellwerkstechnik sowie die Unverzichtbarkeit eines sicheren (secure) Codings für sichere (safe) Systeme war Thema unter „Entwurfsparadigmen“. Die abschließenden Vorträge zeigten nicht nur die Perspektiven „Formaler Methoden“ im Umfeld sicherheitsrelevanter Systeme, sondern belegten, wie diese bereits heute in der Praxis der Zertifizierung Einsatz finden können.

3 Positionsstatements

In den Positionsstatements zeigten sich unterschiedliche, teilweise konträre Einschätzungen insbesondere der Tragweite modellbasierter und formaler Ansätze. Bereits heute in speziellen ausgewählten Situationen praktisch genutzt, wurde ihnen einerseits das Potential zugesprochen, eine ingenieursgerechte und industrielle Softwareentwicklung zu befähigen. Benannte Perspektiven waren fundierte Nachvollziehbarkeit, frühere und damit kostengünstigere Prüfung auf höherer Abstraktionsebene, Ersatz traditioneller Tests durch Korrektheitsnachweise, sowie zuletzt die Überwindung juristischer Argumentation und betriebswirtschaftlicher Überlegung durch mathematisch belegbare Gewissheit.

Andererseits wurden derzeit, und möglicherweise zukünftig weiter ungelöste, Beschränkungen und Unzulänglichkeiten gesehen, die einer wirklich generellen Anwendung derartiger Ansätze entgegenstehen. Unzulänglichkeiten der vorwiegenden Programmiersprachen, Semantikschwächen gängiger Modellierungssprachen, Hürden bei Einführung und Anwendung formaler Sprachen, Unbeherrschbarkeit von Variabilität und Komplexität. Darüberhinaus stelle die Langlebigkeit der Systeme mit lange Wartungsverpflichtungen neue Werkzeuge und Methoden vor ebenso hohe Verpflichtungen.

4 Diskussion

In der Diskussion wurden auch die Positionsstatements nochmals aufgegriffen. Die Vielfalt der oft konträren Stellungnahmen findet sich in der Folge frei und unkommentiert aufgelistet.

Vor- und Nachteile der Programmiersprachen waren erneut Thema: sicheres Ada, performantes und weitverbreites C/C++, programmierfreundliches Java.

- Java wird bereits bei der Mars Rover Mission Einsatz finden. Gute Gründe für JAVA bestehen z.B. bei sicherheitskritischen Komponenten, deren Umfeld selbst in JAVA realisiert ist.

- Weiterhin limitierte Rechenressourcen erlaubten auch heutzutage oft nur C; ein ADA Frachtsystem für das Spaceshuttle musste schlicht in C nachprogrammiert werden.

- Inakzeptabel ist andererseits, dass grundlegende Probleme mit Integer-Overflow und Arraygrenzen bei C/’C++ im Gegensatz zu Ada nicht abgefangen werden, wo dies doch mit heutigen leistungsfähigen Prozessoren leicht lösbar sein sollte.

- Nur durch strikte Beschränkung des Sprachumfangs werden C/C++ (MISRA) sowie Java (SCSJ) safety-tauglich werden.

Bisher fanden formale Methoden wenig Eingang in die Zertifizierung selbst. Durch formale Verifikation wird eher eine zusätzliche Prüfungsmöglichkeit als eine Eliminierung von Tests erwartet. Verschiedene Prüfungen werden damit erst machbar, allerdings kann dadurch ein leichterer Sicherheitsbeweis trotzdem nicht erwartet werden. Ein weit verbreiteter Irrtum ist zudem, dass durch wie auch immer bessere Prüftechniken Sicherheit selbst hineingeprüft wird.

- „Sicherheit kann nicht in ein System hineingetestet werden." Sicherheit ist ein Konzept, das im Entwurf behandelt werden muss. Sicherheit kann nicht durch noch so massives Testen erreicht werden.

- Formale Methoden bringen zusätzliche Qualität. Dennoch bleiben Tests unverzichtbar. Niemand würde sich mit gutem Gefühl in ein Fahrzeug (Auto, Zug, Flugzeug) setzen, das nicht auch getestet wurde.

- Formale Prüfungen basieren auf Modellen, beruhen auf Abstraktion und erfassen damit naturgemäß nicht das gesamte System. Abstraktion und Bewertung erfordern selbst Sorgfalt und Sachverstand, um nicht Wesentliches unberücksichtigt zu lassen.

- Anforderungen und ihre vollständige Erhebung sind die Grundlage, auf dem das Sicherheitskonzepts und seine Prüfung basiert. Neben Domänenkompetenz bedarf es eines systematischen Vorgehens.

- Mitunter sind Systeme so komplex, dass sie sich einem vernünftigen und vollständigen Spezifizieren entziehen. Am Ende der Entwicklung (Systemintegration) treten oft eine Diskrepanz zwischen Kundenintention und Resultat zu Tage.

Techniken und Tools allein werden nicht zum Erfolg führen, ohne den Faktor Mensch zu integrieren. Werkzeuge und Methoden werden aber oft überschätzt. Fundierte Ausbildung und industrielle Erfahrung, Kompetenz und Sorgfalt, sowie organisierte Abläufe sind unverzichtbare Vertrauensgrundlage:

- Man sollte nie vergessen, dass Zertifizierung dem Vermitteln von Vertrauen dient und berechtigtes Vertrauen voraussetzt. Alle Regulierungen setzen Sorgfalt voraus und sollen hauptsächlich sicherstellen, dass keine Aspekte übersehen werden.

- Eine der großen Herausforderung ist es, berechtigtes Vertrauen zukünftig durch nachvollziehbare Gewissheit zu ersetzen; kurz gesagt Mathematik statt Jura.

- Weitgehend formale Beweise im mathematischen Sinne könnten in fernerer Zukunft (20 Jahre?) die juristischen Argumentationen ersetzen.

- Wichtig ist eine präzise Kommunikation zwischen den Beteiligten vom Kunden/Auftraggeber bis zur Abnahme/Zertifizierung.

- Der Drang nach Tools wie DOORS kann mit dem Griff nach dem Strohhalm verglichen werden. Man hat damit ein Werkzeug, mit dem man Anforderungen aufschreiben kann, aber mehr auch nicht.

- UML wird als interdisziplinäres Verständigungsmittel oft weit überschätzt. Fachrichtungsübergreifend funktioniert es weit weniger als beim Austausch zwischen Softwareexperten.

- Formale Methoden werden oft überbewertet. Ohne ingenieurstechnischen Sachverstand wird sich ihr Potential nicht erschließen.

- Generierung und Automatisierung manueller Schritte werden Softwareentwicklung von ihrem derzeitigen Manufaktur-/Kunststatus auf den Level industrieller Fertigung heben.

- Angesichts katastrophaler Fehler stellt sich die Frage, wie Experten befähigt werden können, Konsequenzen aus Ketten- und Mehrfachfehlern früher und besser vorherzusehen und damit im Sicherheitskonzept wirklich berücksichtigen zu können.

- Die Erzeugung direkter persönliche Betroffenheit bewirkt erstaunliche Fortschritte. China verpflichtete Manager des Jahr-2000-Problems zum Fliegen zum Jahreswechsel, und bewältigte damit den schleppenden Fortschritt.

- Auch das Space-Shuttle-Programm setzt auf das persönliche Element. Hier müssen sich die Manager den direkten Vertrauensfragen der Astronauten stellen: „Haben Sie wirklich alles für meine sichere Rückkehr getan?“.

- Systems Engineering kann man genauso selten studieren wie Software Engineering, oder noch seltener.

Unumstritten ist, dass es eines konsequenten System- und Software-Engineerings bedarf, beginnend mit den Anforderungen bis hin zur Freigabe, um Qualität und Vertrauenswürdigkeit kritischer Systeme auch in Zukunft sicherzustellen. Der Trend zur Integration auch Web-basierter Techniken in kritischen Anlagen macht fachübergreifende Zusammenarbeit zwischen funktionaler (Safety) und informationstechnischer Sicherheit (Security) unumgänglich.

Insbesondere im Zuge wachsender Komplexität und eines zunehmenden Softwareanteils werden modellbasierte Entwicklung, formale Ansätze für Design und Verifikation, Code-Generierung und Test-Automatisierung an Bedeutung gewinnen. Dies kann zwar nie Sorgfalt und Fachkompetenz ersetzen. Aber nur so werden die verantwortlichen Experten die steigenden Anforderung an Nachvollziehbarkeit, Dokumentation, Tests und Prüfungen mit vertretbarem Aufwand und Kosten erfüllen können.

5 Fazit

Die sich an die Vorträge und Positionsstatements anschließende lebhafte Diskussion zeigte ein reges Interesse aus dem Teilnehmerkreis. Zutage traten viel Übereinstimmung in grundlegenden Fragen, manche abweichende, pointierte Meinung und insgesamt einen Konsens, dass sich das Gebiet in einer schnellen Entwicklung befinde, wo sich noch kein Ende abzeichne. So erscheint eine Fortführung der Veranstaltung in den kommenden Jahren, vielleicht dann auch unter Beteiligung der dieses Mal nicht vertretenen Industriesparten und insbesondere der Prüfungs- und Zulassungsbehörden, als sehr wünschenswert