## GRAND CHALLENGES DER INFORMATIK

# Verlässlichkeit von Software

### 1. Koordinator des Vorschlags

Prof. Dr. Michael Goedicke, Universität Duisburg-Essen, paluno, Gerlingstraße 16, 45117 Essen

Prof. Dr. Gregor Engels, Paderborn

Prof. Dr. Dirk Nowotka, Kiel

Prof. Dr. Wolfgang Thomas, Aachen

Prof. Dr. Thomas Wilke, Kiel

### 2. Titel der Grand Challenge

**Verlässlichkeit von Software**

### 3. allgemein verständliches Abstract der Grand Challenge

Software ist heutzutage ein Kernbestandteil nahezu aller Systeme im privaten und beruflichen Leben. Bei jedem Telefonanruf, jeder Autofahrt, jeder Banküberweisung, jedem Arztbesuch ist Software an vielen Stellen im Einsatz. Aber können wir uns als *Anwender und Nutzer von Software* auch darauf verlassen, dass die Software genau die Anforderungen erfüllt, die wir von ihr erwarten? Können wir sicher sein, dass die Software in allen Einsatzszenarien verlässlich ist?

Hierbei umfasst der Begriff der *Verlässlichkeit* nicht nur die funktionale Korrektheit der Software, sondern auch alle nicht-funktionalen Aspekte wie z.B. Sicherheit, Datenschutz, Performanz, Ressourcenverbrauch und Usability.

Software wird heutzutage auf unterschiedlichste Art und Weise entwickelt – sei es von einem professionellen Softwareentwicklungsunternehmen, sei es durch Weiterentwicklung existierender, oft unzulänglich dokumentierter Software, sei es durch Domänenexperten, für die Softwareentwicklung keine erlernte Schlüsselfähigkeit ist oder sei es sogar durch Laien als Endanwender, die z.B. eine von ihnen entwickelte Excel-Anwendung als Grundlage für wichtige wirtschaftliche Entscheidung verwenden. Auch für diese *Entwickler von Software* stellt sich die Frage, ob die verwendete Software genau die Anforderungen erfüllt, die an sie gestellt werden. Können Entwickler sicher sein, dass ihre Software in allen Einsatzszenarien verlässlich ist? Mit anderen Worten: Kann ein Entwickler Garantien geben, dass ein Benutzer einer Software sich auf die Funktionsweise und weitere Eigenschaften verlassen kann.

Bisher existieren keine Methoden und Werkzeuge, die es erlauben, die Verlässlichkeit von Software im Allgemeinen zu überprüfen und zu garantieren. Aufgrund der großen Bedeutung und der zentralen Rolle von Software können wir uns das aber aus vielerlei Gründen nicht leisten. Software muss nachweisbar verlässlich arbeiten! Methoden und Werkzeuge, die Verlässlichkeit einer Software zu jedem Zeitpunkt und in jeder Umgebung garantieren können, müssen bereitgestellt werden!

Es ist klar, dass Verlässlichkeit von Software nicht einfach durch ein vollautomatisiertes Werkzeug erzielt werden kann. Grundsätzliche Überlegungen, Aufwands- und Ressourcengründe, unsichere Informationen, aber auch Aspekte der Nutzbarkeit durch menschliche Anwender sprechen dagegen. Für einzelne Aspekte ist eine Automatisierung jedoch machbar. Für andere Aspekte müssen nachvollziehbare Methoden und Prozesse entwickelt werden, um die Verlässlichkeit der Software zu garantieren. Hierzu gehören etwa Anwenderstudien, um den Usability-Aspekt einer Software zu garantieren.

# Verlässlichkeit von Software

Insgesamt müssen Konzepte, Methoden, Sprachen und Werkzeuge (weiter-)entwickelt werden, die es erlauben, die Verlässlichkeit von Software zu garantieren. Dort wo diese Forderung aus prinzipiellen Gründen nicht umgesetzt werden kann, muss die Vorhersagbarkeit von Aspekten der Verlässlichkeit von Softwaresystemen bzw. die Vorhersagebarkeit bei Änderungen – auch im Betrieb – unterstützt werden.

## 4. Beschreibung der Grand Challenge

Die hier beschriebene Herausforderung ist in zweierlei Hinsicht eine Grand Challenge:

- Erstens, weil die Erarbeitung einer erfolgreichen Lösung nicht ohne den Einsatz von inter- als auch transdisziplinärem Wissen erfolgen kann.
- Zweitens, weil aufgrund der hohen Vernetzung von Softwaresystemen und der damit verbundenen hohen Abhängigkeit vom Nutzungskontext die Grenzen eines Softwaresystems kaum noch festgelegt werden können.

Insbesondere für den ersten Punkt muss eine neue Form der Kooperation von Fach- und Anwendungsexpertinnen und -experten erzielt werden, die sich gemeinsam den vielfältigen Herausforderungen stellen. Hierzu zählen im interdisziplinären Sinne u.a. Expertinnen und Experten aus den Bereichen

- Requirements Engineering (vollständige, nachvollziehbare Anforderungen),Algorithm Engineering (Entwurf und Entwicklung von korrekten, effizienten, effektiven Algorithmen),
- Method Engineering (situationsbezogene Entwicklungsmethoden),
- Konfigurations- und Variantenmanagement,
- End-user Software Development (unterstützende Methoden und Werkzeuge, die auch durch einen End-Anwender genutzt werden können),
- Usability Engineering (ergonomische Nutzungsschnittstellen),
- Compilerbau (Software-Analysetechniken, Kontroll-/Datenfluss),
- Testen (Testautomatisierung, Testüberdeckung, Testeffizienz),
- Verifikation (Model Checking, unsicheres Wissen, komplexe Modelle),
- Selbst-adaptive Systemen (selbst-heilende Systeme),
- Fehlertolerante Systeme (Mehrfachsysteme).

Aber auch im transdisziplinären Sinne müssen Software-Expertinnen und -Experten mit den Expertinnen und Experten der Anwendungsdomäne zusammenarbeiten und hierbei ein gemeinsames Verständnis von Begrifflichkeiten (Ontologien) und domänenspezifischem Wissen erlangen.

## 5. Warum ist das Problem schwierig?

- Software wird auf der Basis unpräziser Anforderungen entwickelt, im Laufe der Zeit häufig geändert und erweitert, ohne dokumentierte Entwurfsentscheidungen erstellt, oft verteilt und vernetzt sowie mit unklaren Grenzen ausgeführt und überdies mit unerwarteten Einsatz- und Kontextbedingungen konfrontiert. Konkrete Herausforderungen sind hierbei Größe der Systeme,
- Abwägung der Ausdruckstärke von Modellierungssprachen gegen die Komplexität von entsprechenden Analyseverfahren,
- Abstraktions- und Granularitätsgrad von Beschreibungen,
- Mehrdeutigkeit und Widersprüchlichkeit von informellen Beschreibungen,
- Verwendung unterschiedlicher Ontologien, Sprachen und Darstellungsformen in der informellen Anforderungsbeschreibung,
- Wiederverwendung von Beschreibungen oder auch Softwarekomponenten, die nicht ausreichend klar und eindeutig sind,
- sich verändernde Kontextbedingungen auch bei laufenden Softwaresystemen,
- Integration von diskreten und kontinuierlichen Systemen.

Insgesamt ist die Kombination von informellen und formalen, mehrdeutigen und eindeutigen Beschreibungen eine große Schwierigkeit, um Softwaresysteme zu erstellen und zu pflegen, die eher strengen und eindeutigen Regeln folgen sollen. Teilbereiche von Softwaresystemen können mit Hilfe starker formaler Methoden verifiziert werden. Dies muss sinnvoll mit Methoden kombiniert werden, die für die Entwicklung von (Teilen von) Softwaresystemen angewendet werden, die sich einer solchen strengen Prüfung bzw. Vorhersage von Eigenschaften entziehen.
Insbesondere ist die quantitative Größe von heutigen Softwaresystemen sowie die vielen gegenseitig von einander abhängigen Qualitäten, die das gewünschte Softwaresystem besitzen soll, die größte Schwierigkeit in dem derzeitigen Prozess, bessere Entwicklungsmethoden bzw. Analysemethoden zu finden.

## 6. In welchem Zeithorizont erwarten Sie eine Lösung?

Für Software mit festgelegten Rahmenbedingungen und in bekannten Domänen werden Lösungen in Form von Methoden und Werkzeugen in 5 -10 Jahren erreichbar sein. Inwieweit eine umfassende, allgemeine Lösung überhaupt erreichbar ist, ist ein weiterer wichtiger Forschungsbereich.

## 7. Welche Anstrengungen wurden bereits in die Lösung investiert und welche Disziplinen müssten zusammenarbeiten?

Das Thema der Verlässlichkeit insbesondere in Hinsicht auf Korrektheit und Zuverlässigkeit von Software wird in vielen Teildisziplinen der Informatik seit Jahrzehnten bearbeitet. Hierzu gehören Arbeiten im Bereich der Algorithmenentwicklung und –analyse, der formalen Spezifikationssprachen, der automatischen Verifikation, der Semantik- und Simulationstechniken, der Entwicklungs-, Architektur- und Testmethoden der Softwaretechnik sowie der Mensch-Maschine-Schnittstellen-Unterstützung. Viele dieser Teildisziplinen arbeiten isoliert. Hier sind gemeinsame Anstrengungen erforderlich. Hinzu kommen die Disziplinen der Anwendungsdomänen sowie Hardware-nahe Disziplinen, um die immer häufiger entwickelten Cyber-physikalischen Systeme mit abzudecken. Die damit skizzierte Challenge geht also über die „formale Verifikation“ (vgl. die „Verified Software Initiative“ von Hoare und anderen) weit hinaus.

Einzelne Aspekte dieser Grand Challenge wurden in der Vergangenheit bereits in verschiedenen, von der DFG geförderten Forschungsvorhaben studiert. Hierzu gehören etwa

- das Schwerpunktprogramm "Mathematische Methoden zur Extraktion quantifizierbarer Information aus komplexen Systemen" (SPP 1324),
- das Schwerpunktprogramm “Reliably Secure Software Systems – RS3” (SPP 1496),
- das Schwerpunktprogramm “Design for Future” (SPP 1593).

In allen zugehörigen Projekten war aber eine grundsätzliche und umfassende Behandlung der Fragestellungen dieser Grand Challenge nicht das Ziel.

## 8. Bitte beschreiben Sie die mögliche Lösung der Grand Challenge in einem leicht verständlichen Ziel-Szenario.

Software wird in der Zukunft unser tägliches Leben mehr und mehr beeinflussen und unterstützen. Hierzu gehören etwa Softwareanwendungen („Apps“) auf dem Smartphone, aber auch die vielfältigen Softwarebestandteile etwa in den Geräten einer Hausautomatisierung (Licht, Heizung, Rolläden, Küchen- und Reinigungsgeräte, Serviceroboter…). Diese werden in der Zukunft nicht nur für sich alleine stets intelligenter werden, sondern durch ihre Vernetzung und Anbindung ans und ihre Vernetzung durch das Internet eine große Komplexität erreichen.
Aber sind diese Softwareanwendungen auch verlässlich? Kann ich mir sicher sein, dass durch die Nutzung einer App auf meinem Smartphone keine Aktivitäten im Hintergrund passieren, die meine persönlichen Daten an andere weiterleiten? Analoge Fragen stellen sich in der Hausautomatisierung: Kann ich mir sicher sein, dass der

Staubsaugerroboter nicht andere Systeme im Haus unkontrolliert beeinflusst oder mich flächendeckend überwacht und so einen größeren Schaden verursacht?

Wir brauchen eine genaue Vorhersagbarkeit der Verlässlichkeit von Software – von der Modellierung bis zum Betrieb des Softwaresystems. Hierzu ist es nötig, dass nachvollziehbare, präzise und damit automatisch analysierbare Beschreibungen eingesetzt werden. Auf dieser Basis muss insbesondere im Falle von (sicherheits-) kritischen Anwendungen die Software (wie jedes andere technische Produkt) zertifiziert werden. Dies muss, um den Marktanforderungen gerecht zu werden, soweit wie möglich automatisiert sowohl bei der Entwicklung als auch beim Betrieb der Software möglich sein. Die Beschreibungen müssen – um die sehr wichtige Forderung der Nachvollziehbarkeit zu erfüllen – auch bei komplexen Problemstellungen für die menschlichen Entwickler und Betreiber einfach zu verstehen sein. Abstraktionsformen wie z.B. hierarchische Darstellungen, Kapselungs- und Parametrisierungskonzepte, aber auch die Benutzbarkeit der entsprechenden Werkzeuge und Methoden tragen zu der Lösung wesentlich bei. Im Falle eines erkannten Fehlers müssen selbst-heilende Mechanismen in Gang gesetzt werden, um den Fehler zu beheben, also um Auswirkungen lokaler Änderungen im globalen Kontext zu beherrschen. Zum Beispiel werden in der Steuerung eines Hausautomatisierungssystems lokale Fehler (Ausfall eines Sensors im Staubsaugerroboter) kompensiert unter Bewahrung der globalen Konsistenz (Stabilität der gesamten Hausautomatisierung). Insgesamt muss das Wechselspiel der drei Aspekte Technologie-Mensch-Anwendungsdomäne funktionieren, durch geeignete Konzepte, Methoden und Werkzeuge.

## 9. Bitte benennen Sie eine Ziellinie, anhand derer die Herausforderung als gelöst betrachtet werden kann.

Die hier dargestellte Herausforderung wäre allgemein gelöst, wenn für jede beliebige Software in jeder beliebigen Domäne Verlässlichkeit von Software garantiert werden könnte. Es ist zunächst eher davon auszugehen, dass dies nur in eingegrenzten Domänen und eingegrenzten Nutzungskontexten garantiert werden kann. Ein geeigneter Benchmark wäre etwa im Beispielszenario der Hausautomatisierung, dass in 5 Jahren angestrebt wird, mit geeigneten Methoden und Werkzeuge ausgewählte Aspekte der Verlässlichkeit isolierter Anwendungen zu garantieren. In 10 Jahren sollte dann die Verlässlichkeit vernetzter Anwendungen garantiert werden können.

## 10. Welche sozialen / gesellschaftlichen / ökonomischen Probleme lassen sich mit der Grand Challenge adressieren (mit Beispielen)?

Die Lösung zu dieser Grand Challenge wird dazu führen, dass in allen Bereichen des täglichen Lebens die Verlässlichkeit von Software gegeben ist. Hierdurch werden Katastrophen durch Softwarefehler in sicherheitskritischen Bereichen vermieden, etwa in der Hausautomatisierung, in der Medizintechnik durch Fehler in der Operationssoftware oder im Management von kritischer Infrastruktur. Zudem werden in allen Bereichen des täglichen Lebens Ressourcen in großem Umfang gespart, wenn z.B. Fehler in industriell genutzter Software eliminiert sind (denn Fehlerkorrekturen wie zum Beispiel Rückrufaktionen im Automobilbereich verursachen extreme Kosten).

## 11. Weiterführende Literatur

[1] C.A.R. Hoare, J. Misra, G. T. Leavens, and N. Shankar. 2009. The verified software initiative: A manifesto. ACM Comput. Surv. 41, 4, Article 22 (October 2009)[2] http://www.dfg-spp1324.de/
[3] http://www.spp-rs3.de/
[4] http://www.dfg-spp1593.de/

## 12. Unterstützende GI-Gliederungen (Fachbereiche, Fachgruppen), Mitautoren

Bei den Autoren handelt es sich um Mitglieder der GI Fachbereiche *Grundlagen der Informatik* und *Softwaretechnik.*