Sicher fahren

– Absicherung moderner Fahrzeugsoftware

Moderne Fahrzeuge sind fahrende Computer. Damit eröffnen sich nicht nur neue Chancen, sondern auch neue Fehlerquellen und Angriffsmöglichkeiten. Die TU Darmstadt entwickelt modellbasierte Verfahren zur Überwachung der Fahrzeugsoftware, um diesen neuen Gefahren zu begegnen. Eine strikte Trennung von Überwachungs- und Fahrzeugsoftware erlaubt eine Absicherung von Steuergeräten ohne das Risiko, durch die Aktualisierung der Fahrzeugsoftware selbst neue Fehler einzuführen.

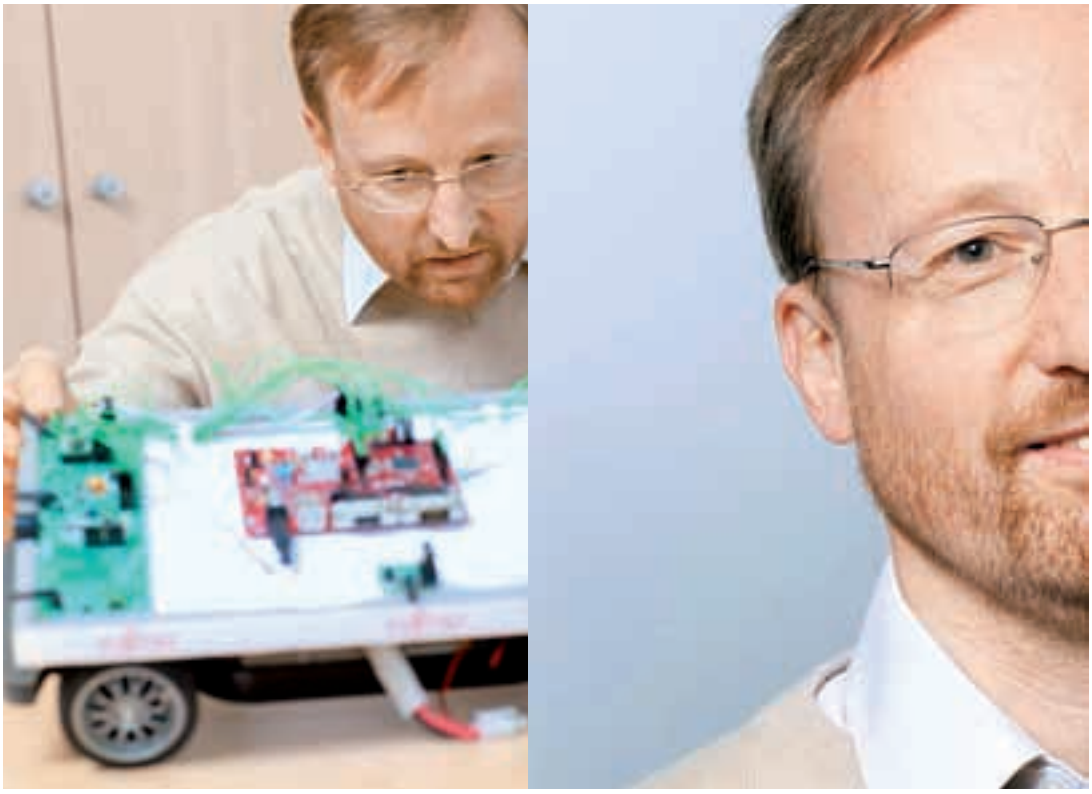
Drive safely: Securing automotive software

Modern cars are driving computers. Therewith new risks like software errors and malicious attacks arise. To address these challenges, researchers at TU Darmstadt develop a model-driven approach for monitoring automotive software. By strictly separating monitoring and automotive software the approach succeeds in securing electronic control units without running the risk of introducing new errors into the automotive software itself.

Sven Patzina, Lars Patzina, Eric Bodden, Mira Mezini, Andreas Sewe, Andy Schürr • Getrieben durch technische Innovationen werden eingebettete Systeme zunehmend stärker untereinander vernetzt und bieten Kommunikationsschnittstellen an. War früher zur Manipulation physischer Zugriff auf die Steuergeräte nötig, ist dies bei aktuellen Systemen nicht mehr erforderlich. Heutzutage können sie nicht als von der Außenwelt abgeschlossene Einheiten angesehen werden, obwohl sie oftmals als solche entwickelt wurden. Bei ihrer Spezifikation wurde häufig wenig Aufmerksamkeit auf Sicherheitsmechanismen wie Verschlüsselung und sicheres Komponenten-Design zur Abwehr von Angriffen von außen gelegt. Die Forschung hat jedoch gezeigt, dass moderne Netzwerke in Autos solche Sicherheitsmechanismen benötigen [1]. Zur nachträg-lichen Absicherung dieser Systeme ist es daher erforderlich, eine abgesicherte Kommunikation im Auto und eine sichere Architektur zur Verbesserung der Privacy und der Security zu entwickeln [3].

Selbst bei der Neuentwicklung eingebetteter Systeme, bei der alle empfohlenen Verfahren durchgeführt werden, ist es meist nicht möglich, alle Sicherheitslücken zu eliminieren und jeden möglichen Angriff vorherzusehen. Betrachtet man große heterogene Systeme oder Dienste und will

Mira Mezini



Andy Schürr

diese nachträglich absichern, ist dies meistens ökonomisch oder technisch nicht zu realisieren. Resultierend daraus kann bei keinem System davon ausgegangen werden, dass es sicher ist – sei es durch unbekannte Schwachstellen oder durch die Verwendung von Legacy-Komponenten, die nicht mehr angepasst werden können.

Wenn die bei der Spezifikation und Implementierung umgesetzten Sicherheitsmechanismen überwunden wurden, steht einem Angriff nichts mehr im Wege. Der Hersteller müsste eine Rückrufaktion starten und die Schwachstellen beheben, um seine Kunden vor der Bedrohung zu schützen. Eine Auslieferung von neuer Software, wie es vom PC über

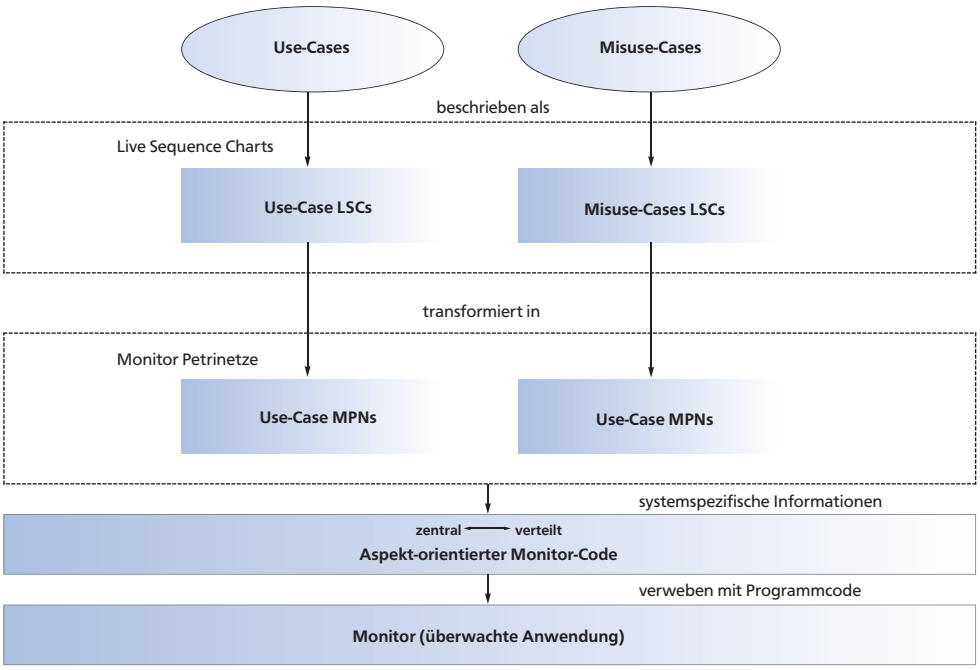


Abbildung 1
Modellbasierter Security-Monitor-Generierungsprozess. Im Rahmen der CASED-Forschungen an der TU Darmstadt wird ein modellbasierter Entwicklungsprozess entworfen, der es in Zukunft der Industrie ermöglichen wird, Monitore vollautomatisch aus grafischen Spezifikationen zu generieren. Ohne diesen Prozess ist die Entwicklung des Monitors ähnlich komplex wie Änderungen am System selbst. Der modellbasierte Ansatz hilft Fehler zu vermeiden.

einen Funkkanal bzw. das Internet bekannt ist, stellt im Bereich der eingebetteten Systeme ein hohes Sicherheitsrisiko dar, da ein fehlerhaftes Update schwerwiegende Konsequenzen haben kann.

Im Automobilbereich ist ein Update der Multimediakomponenten noch vorstellbar, der Eingriff in sicherheitskritische Steuergeräte jedoch nicht zu vertreten.

Um kostenintensive Maßnahmen bis zum nächsten regulären Service-Termin in der Werkstatt hinauszuzögern, ist eine weitere Instanz nötig: Ein Monitor, der in Software oder Hardware umgesetzt sein kann, überwacht die Kommunikation verschiedener Komponenten untereinander oder die Komponenten selbst. Dabei kann er abweichendes oder auf Angriffe schließendes Verhalten erkennen und gegebenenfalls Gegenmaßnahmen einleiten [2]. Diese Monitore könnten im Fall einer neuen nicht abgedeckten Sicherheitslücke um neue Signaturen erweitert werden und dadurch die Lücke erkennen und absichern, ohne dass eine Anpassung der kritischen Systeme notwendig wäre.

Detaillierte Beschreibung der Abläufe als Life Sequence Charts

Was passiert nun im Detail? Die Sequenzen, die die (Mis-)Use-Cases beschreiben, werden als Life Sequence Charts (LSCs) modelliert. Diese Charts bieten die Möglichkeit optionale („cold“) und erforderliche („hot“) Nachrichten zu spezifizieren. Abbildung 2b beschreibt ein Muster für das oben als Misuse-Case definierte Fehlverhalten des Tempomaten. In diesem Szenario müssen vier Instanzen - der Fahrer, der Tempomat, das Fahrzeug und der Sensor für die Abstandsbestimmung – und deren Kommunikation untereinander betrachtet werden. Während der Fahrt kann der Fahrer die Geschwindigkeit des Tempomats beliebig oft anpassen (setzeGeschwindigkeit()). Entscheidet sich der Fahrer den Tempomat zu aktivieren (starte-Regelung()), sendet dieser die vorher gesetzte Sollgeschwindigkeit an das Fahrzeug. Hat der Sensor

Literatur

[1]C. Groll, A. ; Ruland. Secure and authentic communication on existing in-vehicle networks. In Intelligent Vehicles Symposium, 2009 IEEE, pages 1093–1097, Inst. for Data Commun. Syst., Univ. of Siegen, Siegen, Germany, July 2009.

[2]Michael Müter, Tobias Hoppe, and Jana Dittmann. Decision model for automotive intrusion detection systems. In Automotive – Safety & Security 2010, pages 103–116. Shaker Verlag, 2010.

[3]P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Zhendong Ma, F. Kargl, A. Kung, and J. P. Hubaux. Secure vehicular communication systems: design and architecture. IEEE Communications Magazine, 46(11): 100–109, November 2008.

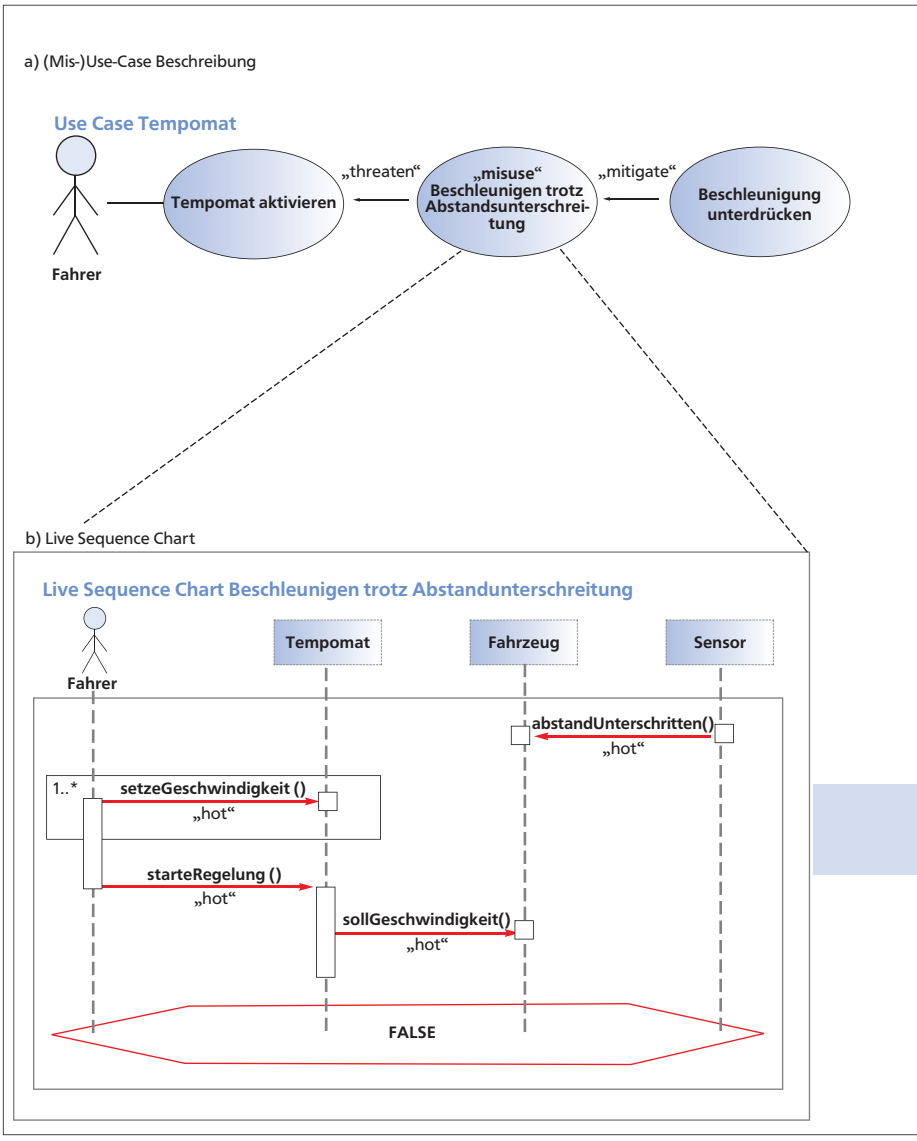
[4]Lars Patzina, Sven Patzina, Thorsten Piper, and Adny Schürr. Monitor Petri Nets for Security Monitoring. In ICPS of International workshop on S&D4RCES 2010. ACM.

eine Abstandsunterschreitung festgestellt, sendet er die Nachricht „abstandUnterschritten()“ an das Fahrzeug. Geschieht dies, wird das Muster erkannt und der Zustand „FALSE“ erreicht. Der mit der Abhängigkeit „mitigate“ dem Misuse-Case zugeordnete Use-Case „Beschleunigung unterdrücken“ wird nun ausgeführt und verhindert die Beschleunigung auf die Sollgeschwindigkeit.

Transformation zu Monitor-Petrinetzen

Obwohl das Beispiel absichtlich einfach gehalten wurde, können in der Praxis sehr leicht komplexe LSCs entstehen. Dadurch ist die Extraktion aller möglichen Abläufe, die durch das LSC beschrieben sind, sehr aufwändig. Um die Generierung von Monitoren zu vereinfachen, werden die LSCs in sogenannte Petrinetze mit einer speziellen Ausführungssemantik (Monitor-Petrinetze) transformiert [4]. Das Petrinetz für das LSC des Misuse-Cases ist in Abbildung 2c dargestellt. Für jede Instanz im LSC, wie Fahrer, Tempomat, Fahrzeug und Sensor, gibt es einen Startplatz, der eine Marke enthält.

Jede Nachricht des LSCs wird durch Transitionen, die den Übergang der Marken zwischen den Plätzen regeln, repräsentiert. Diese werden durch Ereignisse wie das Senden (s) und das Empfangen (r) einer Nachricht ausgelöst. Sind alle Plätze, die einlaufende Kanten in die Transition haben (Vorplätze), mit Marken belegt und das Ereignis der Transition tritt ein, schaltet diese. Alle Marken auf Plätzen vor der Transition werden konsumiert und auf den nachfolgenden Plätzen neu erzeugt. Auf diese Art stellt das Petrinetz alle möglichen Abläufe des LSC in einer einfach zu interpretierenden Weise dar. Aus diesen Petrinetzen wird dann mit Hilfe von systemspezifischen Informationen die Implementierung des Monitors generiert.



Verweben der Monitore

Die resultierenden Monitore müssen dann in einem letzten Schritt mit dem konkreten Programmcode verbunden (verwoben) werden. Dazu haben sich in den vergangenen Jahren Technologien der aspektorientierten Programmierung sehr bewährt. Aspektorientierte Programme erlauben es, die abstrakten Ereignisse des Monitors auf konkrete Programmereignisse abzubilden.

Viele etablierte Verfahren zur aspektorientierten Programmierung verweben die Monitore schon früh mit der Anwendung. Dies hat allerdings zur Folge, dass es zur Ausführungszeit nur schwer möglich ist, zwischen Überwachungslogik und Applikation zu unterscheiden. Die Aktualisierung eines Monitors zur Laufzeit der Applikation ist somit äußerst kompliziert, da sich die vorherige Version nicht rückstandsfrei entfernen lässt.

An der TU Darmstadt wurden daher Techniken entwickelt, um der Ausführungsumgebung zur Laufzeit alle relevanten Informationen über die Überwachungslogik zur Verfügung zu stellen. Dadurch wird die Ausführungsumgebung in die Lage versetzt, Monitore zur Applikation hinzuzufügen oder zu entfernen, ohne dass die Applikation neu gestartet werden muss.

Fachgebiet Softwaretechnik
Dr.-Inform. Eric Bodden
Tel. 06151/16-5478
E-Mail: eric.bodden@cased.de
www.stg.tu-darmstadt.de/staff/eric_bodden

Prof. Dr.-Ing. Mira Mezini
Tel. 06151/16-5311
E-Mail: mira.mezini@cased.de
www.stg.tu-darmstadt.de/staff/mira_mezini/

Dipl.-Math. Andreas Sewe
Tel. 06151/16-3608
E-Mail: andreas.sewe@cased.de
www.stg.tu-darmstadt.de/staff/andreas_sewe/

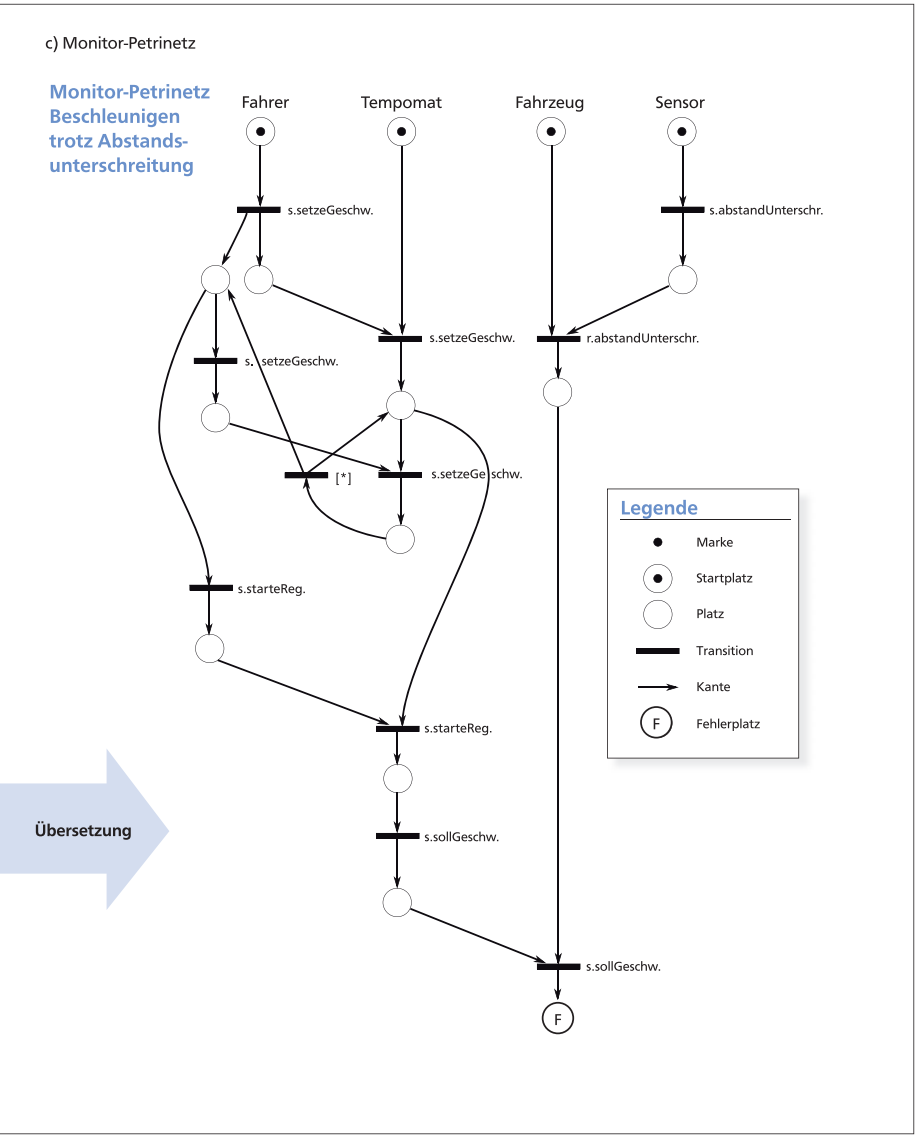


Abbildung 2
Monitor-Entwicklungsprozess am Beispiel des Tempomaten. Als adäquates Mittel zur Erfassung von funktionalen Anforderungen hat sich die Modellierung mit Use-Cases herausgestellt. Zur Darstellung nicht-funktionaler Anforderungen werden Misuse-Cases verwendet, die das Fehlverhalten des Systems beschreiben. Abbildung 2a zeigt unser Beispielszenario. Ein Auto verfügt über einen Tempomaten. Wenn der Fahrer den Tempomaten betätigt, beschleunigt das Fahrzeug auf eine vorgegebene Geschwindigkeit. Ein Angreifer könnte diese Funktionalität nutzen, um einen Auffahrunfall zu provozieren. Der Fahrzeughersteller macht sich daher die im Fahrzeug vorhandene Entfernungskontrolle zunutze und definiert den Misuse-Case „Beschleunigen trotz Abstandsunterschreitung". Dieser soll durch einen Monitor erkannt und durch den Use-Case „Beschleunigung unterdrücken" unterbunden werden.

Zusammenfassend lässt sich sagen, dass IT-Sicherheit auch im Automotive-Bereich schon heute von herausragender Bedeutung ist. Verfahren wie das vorgestellte werden daher bald unerlässlich sein, um die Sicherheit der Verkehrsteilnehmer zu gewährleisten.

Fachgebiet Echtzeitsysteme
Dipl.-Ing. Lars Patzina
Tel. 06151/16-3676
E-Mail: lars.patzina@cased.de
www.cased.de/ueber/mitarbeiter.html

Dipl.-Ing. Sven Patzina
Tel. 06151/16-3676
E-Mail: sven.patzina@es.tu-darmstadt.de
www.es.tu-darmstadt.de/mitarbeiter/sven-patzina/

Prof. Dr. rer. nat. Andy Schürr
Tel. 06151/16-6940
E-Mail: andy.schuerr@es.tu-darmstadt.de
www.es.tu-darmstadt.de/mitarbeiter/andy-schuerr/

Eric Bodden ist Post-Doc am Lehrstuhl für Softwaretechnik der TU Darmstadt. Dort befasst er sich mit statischer Programmanalyse und der Optimierung von Laufzeitmonitoren.

Lars Patzina ist seit 2007 Promotionsstipendiant bei CASED und forscht an einem durchgängigen Entwicklungsprozess zur automatischen Generierung von Security-Monitoren.

Sven Patzina ist seit 2007 wissenschaftlicher Mitarbeiter am Fachgebiet Echtzeitsysteme der TU Darmstadt. Seine Forschung umfasst die modellbasierte Spezifikation von Security-Monitoren.

Andreas Sewe studierte Mathematik an der TU Darmstadt und beschäftigt sich nach seinem Abschluss im Rahmen seiner Promotion am Fachgebiet Softwaretechnik mit virtuellen Maschinen.

Mira Mezini ist seit 2002 Professorin am Lehrstuhl für Softwaretechnik der TU Darmstadt. Ihre Forschungsinteressen sind u. a. modulare Programmierparadigmen, statische und dynamische Programmanalysen sowie intelligente Softwareentwicklungsumgebungen.

Andy Schürr ist seit 2002 Professor am Institut für Datentechnik an der TU Darmstadt. Seine Forschungsthemen sind u. a. modellbasierte Softwareentwicklung von eingebetteten Systemen sowie Modelltransformations- und Spezifikationssprachen.