# Aspekte von Software-Validationsverfahren und ihrer Automatisierbarkeit

**P. Puhr-Westerheide,** Garching

Zusammenfassung

Der vorliegende Aufsatz befaßt sich mit einem Überblick über einige Verfahren und Werkzeuge, die zur Validation von Software eingesetzt werden. Anhand einer Veröffentlichung des NBS/U.S. Department of Commerce wird zunächst ein Überblick über gängige Validationsverfahren gegeben, die in 96 Validationswerkzeugen praktisch realisiert wurden. Die Kurzbeschreibungen dieser Werkzeuge sind in einer Datenbank des NBS/U.S. Department of Commerce abgelegt und in [1] wiedergegeben. Die verwendeten Verfahren sind von unterschiedlicher Komplexität; ihre Bandbreite reicht von einfachen Datenhaltungsverfahren zur Dokumentation bis zum anspruchsvollen Programmbeweis.

Der letzte Teil des Aufsatzes befaßt sich mit Problemen, die die Automatisierung dieser Verfahren aufwirft, und schildert einige Gesichtspunkte zu ihrer Bewertung.

Abstract

The presented paper deals with an overview on some methods and tools which are in use for the validation of software. By means of a publication of the NBS/U.S. Department of Commerce a survey on current valiation methods is given which are implemented in 96 validation tools. Th abstracts of these tools are stored in a database of the NBS/U.S. Department of Commerce, and are listed in [1]. The utilized methods are spread over a wide range of complexity, starting at simple database systems for documentation purposes up to pretentious program proofs. The latter part of the paper deals with problems arising with the automation of these methods, and outlines some viewpoints on their qualification.

## Einleitung

Die wachsende Komplexität und der steigende Umfang von Software, die sich rasch vergrößernde Anzahl von Software-Projekten und die anwachsende Zahl von Menschen, die an solchen Projekten mitarbeiten, hat in der jüngeren Vergangenheit die Entwicklung von Verfahren zur Bewältigung von Software-Problemen gefördert. Die Zeiten, in denen hochqualifizierte Fachleute im Alleingang Codes geschrieben haben und danach Codefehler beseitigt haben, scheinen dem Ende zuzugehen. Anstelle dieser Vorgehensweise treten zunehmend Verfahren, die eine systematische Behandlung von Entwurf, Codierung, Validierung und Dokumentation von Software bewirken. Wegen der Vielschichtigkeit von Software-Problemen scheint es jedoch keine einfache und universelle Vorgehensweise zu geben, mit der alle anstehenden Probleme lösbar wären.

Deshalb ist die Anzahl unterschiedlicher Verfahren groß, entsprechend den verschiedenen Blickwinkeln, unter denen Software-Probleme betrachtet werden können.

Der Anteil der ursprünglich vorhandenen Anstrengungen zur Software-Validation hat sich zu Gunsten von Entwurfsmethoden verringert, gemäß der Erkenntnis, daß ungemachte Fehler nicht beseitigt werden müssen. Auf die Validation kann jedoch schon deshalb nicht verzichtet werden, weil nur mit ihr gewisse Entwurfsfehler entdeckt werden können. Darüberhinaus zwingt der Einsatz von Software an Orten, an denen sich eine Gefährdung von Menschen oder Sachwerten aus ihrem Fehlverhalten ergeben kann, zu ihrer sorgfältigen Validation.

Es hat sich gezeigt, daß gewisse Maßnahmen zur Fehlererkennung, -beseitigung und Validation stets vorgenommen werden müssen. Jedes moderne Übersetzungssystem liefert beispielsweise eine Anzahl von Kreuz-Referenz-Listen, die Aufschluß über die Verwendung von Variablen, externals, o.ä. geben.

Die existierenden Validations-Werkzeuge sind Programme, die automatisch gewisse Untersuchungen an Software vornehmen, deren Durchführung per Hand zu zeitraubend und fehleranfällig wäre. Darüberhinaus gibt es noch eine Reihe von Validationsverfahren, die bisher nicht automatisiert werden konnten.

In der Praxis verwendete Verfahren

Das NBS (National Bureau of Standards/U.S. Department of Commerce) hat Ende 1980 eine Sammlung von Kurzbeschreibungen von 257 Software-Werkzeugen herausgegeben [1]. Der Bericht beinhaltet den Abzug einer Datenbank, in dem die Werkzeug-Kurzbeschreibungen abgelegt sind, sowie einige Querverweis-Listen die die Klassifizierung der Werkzeuge, verwendete Verfahren, Literaturverweise, die verwendete Sprache, die Zielsprache der zu untersuchenden Software, u.a. enthält. Die Verfasser des NBS-Berichts betonen ausdrücklich, keine Wertung der Werkzeuge vorzunehmen, und rufen dazu auf, weitere Werkzeuge bekannt zu machen, behalten sich aber das Recht der Ablehnung vor, ohne auf die von ihnen benutzten Bewertungsmaßstäbe einzugehen.

In diesem Bericht werden die Werkzeuge wie folgt klassifiziert:

| Gruppe | Klasse | Anzahl der Werkzeuge |
|---|---|---|
| 1 | Software-Unterstützungssysteme/ Programmier-Umfeld (Software Support System/Programming environment) | 2 |
| 2 | Software-Management und Wartung (Software Management, Control, and Maintenance) | 110 |
| 3 | Anforderungs/Entwurfsspezifikation, Analyse und Programm-Erzeugung (Requirements/Design-Specification, Analysis and Program Generation) | 42 |
| 4 | Software-Modellierung und Simulation (Software Modeling and Simulation) | 7 |
| 5 | Quellprogramm-Testen und Analyse (Source Program Testing and Analysis) | 96 |

Tabelle 1: Klassifizierung der Werkzeuge

Der Großteil der Werkzeuge gehört zur Klasse "Software-Management, Kontrolle und Wartung" (110 Werkzeuge), die die Unterstützung von Software-Aktivitäten über den ganzen Lebenszyklus betrifft. Die verwendeten Verfahren sind meist wenig speziell und betreffen häufig die Dokumentationsverwaltung.

Die zweitstärkste Gruppe (96 Werkzeuge) betrifft die Klasse "Quellprogramm-Testen und Analyse", von der hier die Rede sein soll und die hier kurz als "Programm-Validation" bezeichnet werden soll.

Um eine grobe Übersicht der verwendeten Verfahren in dieser Klasse zu erhalten, wurden die Schlüsselwörter der Verfahren hier nach der Anzahl der Werkzeuge aufgelistet, in denen sie verwendet werden. Dabei können mehrere Verfahren zu einem Werkzeug gehören und sich auch inhaltlich überschneiden, wie z.B. die Verfahren "Pfadausdruck-Erzeugung" und "Pfadbedingungs-Auflösung". Da die verwendeten Begriffe von den Entwicklern angegeben wurden und für sie noch keine verbindliche Nomenklatur besteht, kann die Auflistung nur einen tendenziellen Überblick geben.

Verfahren, die nur ein- oder zweimal genannt wurden, wurden aus Platzgründen in der Tabelle nicht aufgelistet. Die Kürzel "S" und "D" in der letzten Spalte geben an, ob es sich um ein dynamisches Verfahren (d.h. mit Ausführung des zu testenden Programms) oder ein statisches Verfahren (ohne Ausführung) handelt.

Die aufgelisteten Verfahren sind untereinander keinesfalls bezüglich ihrer Aussagekraft gleichwertig oder von ähnlicher Komplexität; dennoch können einige Schlüsse gezogen werden. Zunächst ist auffällig, daß kein Werkzeug zur Unterstützung statistischer Tests existiert. Weiterhin hat nur ein Werkzeug (in obiger Tabelle nicht aufgelistet) als Verfahren den Programm-Beweis angegeben, das als schwierigstes Verfahren bekannt ist. Nur fünf Werkzeuge prüfen die Einhaltung von Programmier-Standards, und mit Ausnahme der Verfahren Zeit-Analysen (Lfd.Nr.14) und Laufzeit-Analysen (Lfd. Nr.4) werden keine Verfahren zur Validation von speziellen Echtzeit-Problemen angegeben (z.B. Aufzeigen verflochtener Zugriffe auf geschützte Resourcen).

Tabelle der verwendeten Verfahren

| Lfd. Nr. | Schlüsselwort des Verfahrens | Anzahl der Werkzeuge, die das Verfahren enthalten | |
|---|---|---|---|
| 1 | Kreuz-Referenz-Listen | 33 | S |
| 2 | Dokumentationserzeugung | 25 | S/D |
| 3 | Test-Effizienz-Messung | 23 | D |
| 4 | Überdeckungs-Analyse | 23 | S |
| 5 | Code-Inspektion (code auditing) | 20 | S |
| 6 | Flußdiagramm-Erzeugung | 20 | S |
| 7 | Global Data Management | 17 | |
| 8 | Report-Erzeugung | 15 | D |
| 9 | Zuverlässigkeitsmaße | 13 | S/D |
| 10 | Datenfluß-Analyse | 12 | S/D |
| 11 | Programmfluß-Verfolgung (Trace) | 9 | D |
| 12 | Testdaten-Management | 9 | S/D |
| 13 | Konsistenz-Prüfungen | 8 | S |
| 14 | Zeit-Analysen | 8 | D |
| 15 | Versions-Überprüfung | 7 | |
| 16 | Testfall-Erzeugung | 7 | S |
| 17 | Profil-Erzeugung | 7 | D |
| 18 | Änderungs-Validation | 7 | |
| 19 | Ausführungs-Überwachung (execution monitoring) | 7 | D |
| 20 | Interface-Analyse | 7 | S |
| 21 | Pfadausdruck-Erzeugung | 5 | S |
| 22 | Programmstruktur-Untersuchung | 5 | S |
| 23 | Durchsetzung von Standards | 5 | S |
| 24 | Symbolische Exekution (symbolic evaluation) | 5 | S |
| 25 | Komplexitätsmaße | 4 | S |
| 26 | Laufzeit-Analysen | 4 | D |
| 27 | Zusicherungs-Überprüfung (assertion checking) | 3 | S |
| 28 | Fehler-Analyse | 3 | S/D |
| 29 | Pfadbedingungs-Auflösung | 3 | S |

Tabelle 2: Verwendete Verfahren nach ihrer Häufigkeit

Beurteilung der Automatisierbarkeit von Validationsverfahren

Einige Validationsaufgaben können nur mit Werkzeuguntersützung sinnvoll gelöst werden. Eine Programm-Instrumentierung, deren Ergebnisse während eines Testlaufs zur Fortsetzung des Tests benötigt werden, ist nur dann sinnvoll, wenn ein Dialogsystem existiert, mit dem der Fortlauf des Tests beeinflußt werden kann. Solche Dialog-Test- und Bediensysteme gehören deshalb auch zu den ältesten Werkzeug-Entwicklungen. Auch die Instrumentierung selbst oder gewisse Dokumentationen zur Laufzeit des Testlings (Lfd.Nr.10,11,19,26) (Trace-Verfahren oder post-mortem-dumps) werden seit langem mit automatischen Werkzeugen durchgeführt.

Werkzeuge eignen sich sehr gut zu allgemeinen Dokumentationsaufgaben, etwa der Protokollierung von Testläufen (Lfd.Nr.8,17) mit den zugehörigen Eingangsdaten oder zur Generierung von Kreuz-Referenz-Listen (Lfd.Nr.1) und anderen lexikalischen Untersuchungen. Hierbei liegen die Schwierigkeiten eher in einer sinnvollen Ordnung und Begrenzung der erzeugten Datenmengen. Sie können aber auch weitaus schwierigere Aufgaben bewältigen, wie bei dynamischen Testhilfen die Überwachung des Testvorgangs zur Laufzeit mit Einbau von Haltepunkten, Steuerung der Uhr, die Kontrolle von Prozessen und Zugriffe auf Objekte im Dialog (Lfd.Nr.11,19, 26).

Statische Analysewerkzeuge enthalten häufig komplexere Verfahren zur Strukturaufklärung (Lfd.Nr. 6,12) und zur Testdatenerzeugung (Lfd.Nr.16).

Die Resultate der Strukturaufklärung müssen angemessen protokolliert werden. Dies geschieht häufig unter Verwendung eines Graphik-Software-Pakets.

Bei der Validation von Software ist die Gewinnung von Testdaten von großer Bedeutung. Günstig ist es, wenn bereits in der Spezifikations- und Codierungsphase Testfälle angegeben werden. Häufig steht man jedoch vor der Aufgabe, relevante Testdaten aus dem zu prüfenden Programm abzuleiten oder die Vollständigkeit vorhandener Testdaten zu prüfen. Da ein erschöpfendes Testen meist nicht möglich ist, müssen sinnvolle Test-Strategien ermittelt und befolgt werden. Dabei zeichnet sich ab, daß bestimmte Überdeckungs-Strategien (Lfd.Nr.4) allgemein als sinnvoll anerkannt werden [2]. Sie haben zum Ziel, alle Programmteile mindestens einmal auszuführen (C1-Überdeckung [2]) oder darüberhinaus alle Kombinationen von Ein- und Ausgängen von Einzelverzweigungen mindestens einmal zu durchlaufen.

Die Überdeckungsermittlung baut auf dem Kontrollflußgraphen des Testlings auf und liefert eine Menge von Pfaden, die getestet werden sollen. Für diese Pfade müssen dann die Testdaten gefunden werden, die die Pfad-Durchläufe bewirken. Das Verfahren der Pfad-Testdaten-Erzeugung (Lfd.Nr.21, 29) ähnelt dem der symbolischen Exekution (Lfd.Nr. 24) und ist in der Praxis nur schwer zu realisieren. Das Ergebnis des Verfahrens ist ein dem Pfad zugeordneter mathematischer Ausdruck ("Pfad-Ausdruck") aus dem direkt die Eingangsdaten abgelesen werden können, die zur Ausführung des Pfades dienen. Da dieses Verfahren z.Zt. nicht für beliebige Pfade praktisch mit vertretbarem Aufwand realisierbar ist, erfordert die Handhabung eines solchen Werkzeugs eine detaillierte Kenntnis seiner Wirkungsweise. Diese Problematik liegt auf der Hand, wenn man bedenkt, daß die Ermittlung aller vollständigen Pfadausdrücke eines beliebigen Programms es ermöglichen würden, das Programm so umzuschreiben, daß es außer einer Anfangs-Verzweigung keine weiteren inneren Verzweigungen enthält!

Werkzeuge können jedoch durch eine sinnvolle Darstellung der Quellprogramm-Informationen zur Lösung des Problems der Testdaten-Ermittlung beitragen.

Zwei weitere Fragen der Validation wurden bisher noch nicht vollständig beantwortet.

- Läßt sich ein quantitatives Maß für die Validation oder den Grad der Austestung angeben?

und

- läßt sich ein quantitatives Maß für die Komplexität eines Testlings angeben?

Es gibt mehrere Ansätze zur Klärung dieser Fragen; Einer wird an anderer Stelle dieses Heftes besprochen. Eine Bewertung der Validation wird von vielen Werkzeugen durchgeführt (Lfd.Nr.3), auch die Bestimmung von Komplexitätsmaßen (Lfd.Nr.25) wird von einigen Werkzeugen vorgenommen.

Zur Lösung der anfallenden Probleme bei der Software-Entwicklung und -Pflege reicht die Schaffung einer attraktiven Sprache und eines guten Übersetzungssystems allein nicht aus. Software-Entwicklung und -Pflege sollen nach Möglichkeit in ein Umfeld eingebettet sein, das in allen Phasen des Software-Lebenszyklus Hilfsmittel zur Bewältigung der Probleme anbietet. Im Zusammenhang mit den jüngsten Sprachentwicklungen wird dementsprechend die Schaffung eines Umfelds von Software-Werkzeugen angeregt, z.T. sogar gefordert. Daher wurden für die Sprache PEARL mehrere solcher Werkzeuge geschaffen, wie etwa Entwurfs-Hilfsmittel und Test-und Bediensysteme. In einer der letzten Ausgaben der PEARL-Rundschau wurde bereits ein bei der Gesellschaft für Reaktorsicherheit (GRS) entwickeltes Validationswerkzeug vorgestellt, der PEARL-Analysator [3]. An dieser Stelle sei kurz die Einordnung dieses statischen Analysewerkzeugs in die Reihe der Validationswerkzeuge dargestellt (Abb. 1-3). Die im Analysator verwendeten Verfahren betreffen die Lfd.Nr.1,2,6,10,13,16,21,22, und 29 aus Tabelle 2.

Eine detaillierte Beschreibung des PEARL-Analysators mit dem Titel "Der PEARL-Analysator, ein Software-Werkzeug zur Analyse von PEARL-Programmen" [4] kann auf Anfrage über die GRS bezogen werden.

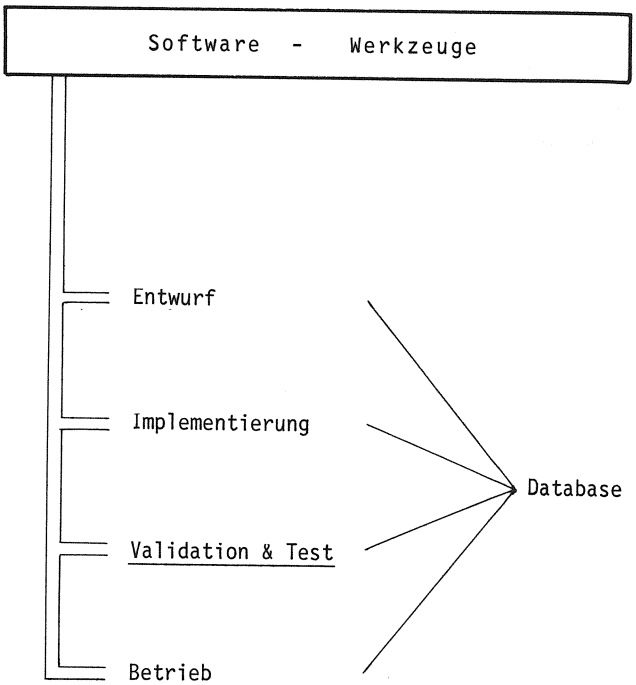


Abb. 1: Übersicht über Software-Werkzeuge. Unterstrichen: Anwendungsbereich des PEARL-Analysators

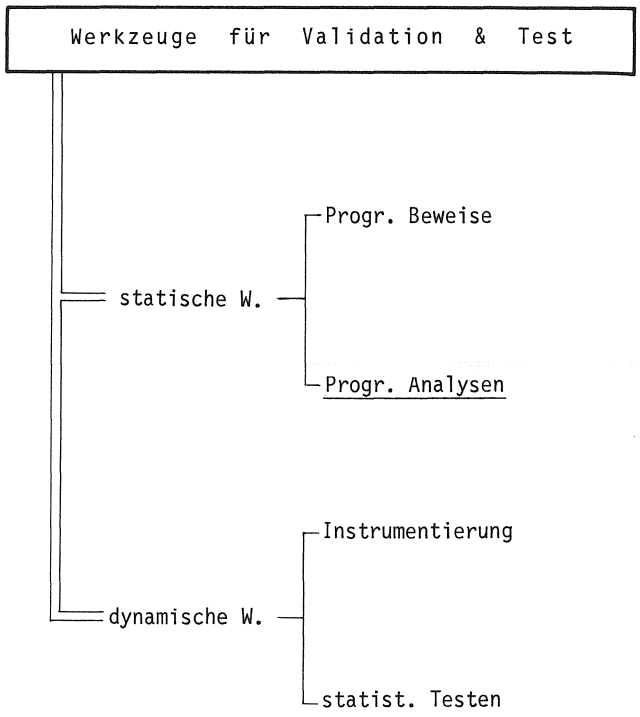


Abb. 2: Übersicht über Validations- und Testwerkzeuge.
Unterstrichen: Anwendungsbereich des PEARL-Analysators

Zusammenfassung

Validationswerkzeuge sind unterstützende Hilfsprogramme zur Analyse und zum Test von Programmen zum Zweck der Qualitätssicherung. Sie basieren auf verschiedenen Verfahren, angefangen bei einfachen Verfahren zur Verwaltung der Dokumentation in einem DV-System bis zur Implementierung von Programmbeweisverfahren. Ein Teil der Verfahren analysiert die zu validierenden Programme

- ohne sie auszuführen (statische Verfahren),

ein anderer basiert auf der

- Ausführung der zu validierenden Programme (dynamische Verfahren).

Werkzeuge und Verfahren werden eingesetzt, um

- die "Handarbeit" zu reduzieren
- die Validation zu formalisieren
- die Validation managebar zu gestalten
- die Validation besser dokumentieren zu können.

Die verwendeten Verfahren sind sehr vielfältig und betreffen meist nur bestimmte Gesichtspunkte der Validation. Ein allgemein anerkanntes, praktisch realisierbares und durchgängiges Verfahren zur Validation ist z.Zt. werder existent noch absehbar, die Entwicklung der Verfahren ist jedoch noch nicht abgeschlossen. Einige einfache Verfahren scheinen sich als Standard-Verfahren herauszukristallisieren, wie etwa das dynamische Testen nach bestimmten Kontrollflußgraphüberdeckungen. Die Dokumentation von Analysen und Tests ist bereits durchgängig in den Werkzeugen vorhanden.

Der Einsatz von Werkzeugen zur Validation von Software wird in Zukunft nötiger als je zuvor, nicht zuletzt aufgrund der steigenden zu bewältigenden Datenmengen.

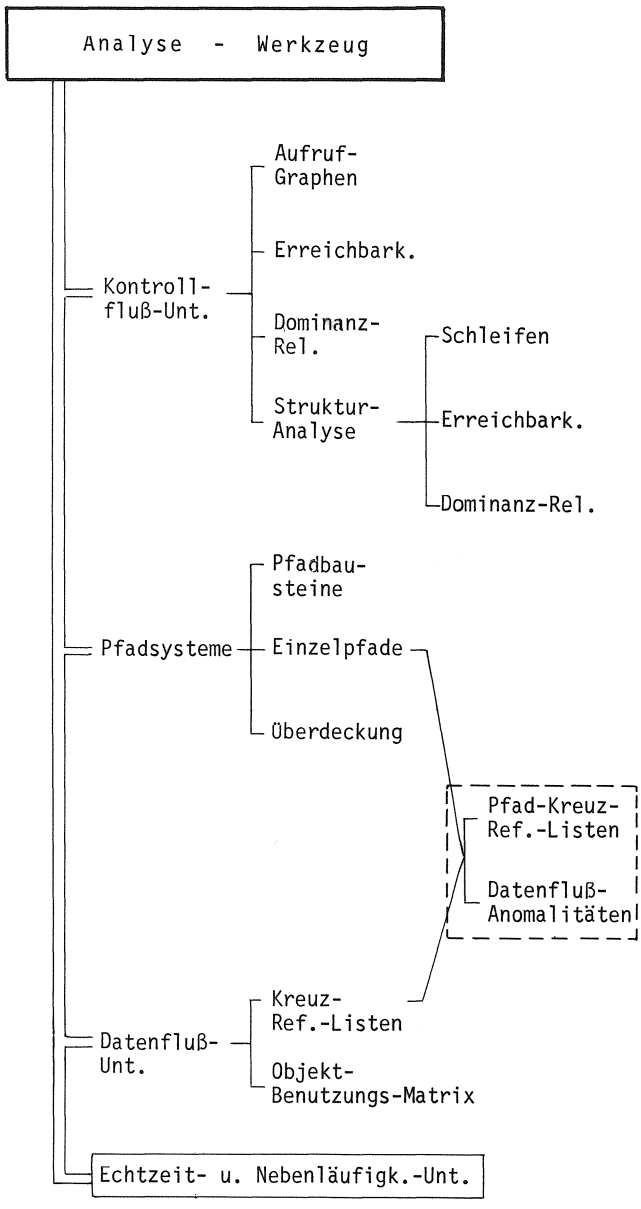


Abb. 3: Von Analyse-Werkzeugen abgedeckte Untersuchungen
[ ] : im PEARL-Analysator nicht realisiert
[- - -] : z.Zt. in Arbeit befindlich

LITERATUR

[1] Houghton, R.C., Oakley, K.A. (Editors)
"NBS Software Tools Database"
U.S. Department of Commerce, Whashington D.C. 20234, NBSIR 80-2159, October 1980

[2] Miller, E., Howden, W.E.
"Tutorial: Software Testing & Validation Techniques"
IEEE Computer Society New York, 1978
IEEE Catalog No.: EHO 138-8

[3] Puhr-Westerheide, P.
"Die statische Analyse von PEARL-Moduln mit dem PEARL-Analysator"
PEARL-Rundschau, Heft 6, Band 2, Dezember 1981

[4] Puhr-Westerheide, P., Märtz, J.
"Der PEARL-Analysator, ein Software-Werkzeug zur Analyse von PEARL-Programmen"
GRS-Bericht, GRS-A-669, Garching/München, Januar 1982

Dr. Puhr-Westerheide, Peter
Gesellschaft für Reaktorsicherheit (GRS) mbH
Forschungsgelände

8046 Garching bei München