



(19)
Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 10 2006 059 629 A1** 2008.06.19

(12)

Offenlegungsschrift

(21) Aktenzeichen: **10 2006 059 629.3**

(22) Anmeldetag: **14.12.2006**

(43) Offenlegungstag: **19.06.2008**

(51) Int Cl.⁸: **G06F 11/36** (2006.01)

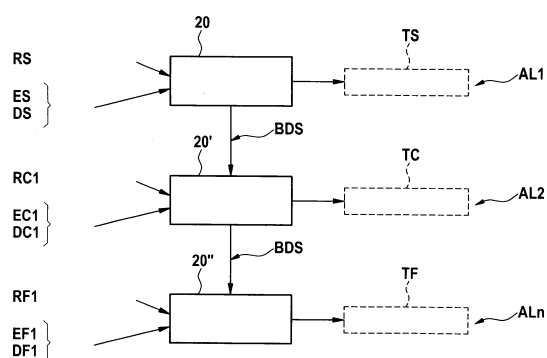
(71) Anmelder:
Robert Bosch GmbH, 70469 Stuttgart, DE

(72) Erfinder:
Liebehenschel, Jens, 65835 Liederbach, DE

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Verfahren und Werkzeug zum Ermitteln von Testfällen für ein System**

(57) Zusammenfassung: Die Erfindung betrifft ein Verfahren und ein Werkzeug zum Ermitteln von Testfällen für ein System, insbesondere für ein Softwaresystem (10), mit den Schritten: Feststellen aller Einflussfaktoren (20) auf eine erste Abstraktionsebene (AL1) des Systems, wobei für jeden dieser Faktoren (20) Äquivalenzklassenvertreter (21) für alle möglichen Ausprägungen eines Faktors (20) angegeben werden können; Zuordnen derjenigen Einflussfaktoren (20'), die für eine sich anschließende, höhere oder niedrigere, zweite Abstraktionsebene (AL2) relevant sind, zu dieser zweiten Ebene (AL2), wobei die Schritte des Feststellens und Zuordnens von Einflussfaktoren (20, 20') über die Abstraktionsebenen (AL1...ALn) des Systems (10) hinweg derart wiederholt werden, dass die jeweils zweite Ebene (AL2) zur jeweils ersten Ebene (AL1) wird, und Erstellen einer Testspezifikation (S) für wenigstens eine Abstraktionsebene (AL1...ALn), wobei einzelne Testfälle (TS, TC, TF) der Spezifikation (S) durch Kombinieren der Äquivalenzklassenvertreter (21) der relevanten Einflussfaktoren (20, 20', 20'') ermittelt werden.



Beschreibung

Offenbarung der Erfindung

Stand der Technik

[0001] Die vorliegende Erfindung betrifft ein Verfahren zum Ermitteln von Testfällen für ein System nach Anspruch 1 und ein entsprechendes Werkzeug nach Anspruch 10.

[0002] Bei der Ermittlung von Testfällen für ein System, insbesondere für ein Softwaresystem oder ein Subsystem eines Softwaresystems, ist der Testentwickler sich widerstrebenden Zielen ausgesetzt. Beispielsweise sollten einerseits die Testfälle alle Fehler finden, um die Zuverlässigkeit der Software sicherzustellen, andererseits sollte der Aufwand so gering wie möglich sein, um die Kosten gering zu halten.

[0003] Eine bekannte Methode zur Ermittlung von Testfällen ist die Classification Tree Method (CTM). Für die Anwendung der CTM existiert eine Toolunterstützung, der Classification Tree Editor (CTE). Die Methode setzt auf einer zu testenden Einheit wie z. B. System, Subsystem, Komponente, Funktion usw. auf und erfordert die Feststellung der relevanten Einflussfaktoren und deren Äquivalenzklassenvertreter für diese eine Einheit. Das Tool unterstützt den Entwickler durch graphische Repräsentation der Testfälle als Kombination der Äquivalenzklassenvertreter der Einflussfaktoren. Die Methode und der Classification Tree Editor stammen von der DaimlerChrysler AG. Die Unterstützung der Erstellung von Testfällen ist dort noch immer ein aktuelles Forschungsthema.

[0004] Eine bekannte Methode zum Risikomanagement ist die Fehlermöglichkeiten und Einflussanalyse (FMEA). Die FMEA betrachtet die Systemstruktur und Abhängigkeiten identifizierter Risiken und möglicher Fehler auf den verschiedenen Subsystemen. Die FMEA verfolgt das Ziel, Risiken zu finden und zu bewerten.

[0005] Eine Methodenauswahl für softwarebasierte Anwendungen wird im Artikel 'Modellbasierte Testspezifikation und Testgenerierung' von Dr. Oscar Slotosch, Validas AG dargestellt. Eine modellbasierte (funktionale) Testspezifikation besteht dabei aus den Elementen Schnittstellenbeschreibung der Testschnittstelle, Variantenmodellierung und Parameterauswahl, Zustandsmodellen für Systemzustände und Testsequenzen. Zusätzlich zu den Beschreibungstechniken sind Modellierungsrichtlinien notwendig, welche die Eindeutigkeit und Umsetzbarkeit der Testspezifikation sicher stellen. Eine Testgenerierung vollzieht schließlich die automatische Umsetzung der Testmodelle in Testcode für die (operative) Testumgebung. Der vorgestellte Ansatz ist unabhängig von dem entwickelten System und der Testumgebung, und basiert nur auf deren Schnittstellenspezifikation.

[0006] Es ist Aufgabe der vorliegenden Erfindung, ein Verfahren bereitzustellen, das eine durchgängige und zuverlässige Feststellung von Einflussfaktoren auf ein System zum Ermitteln von Testfällen erlaubt, das schnell durchführbar und einfach strukturiert ist, sowie kostengünstig implementiert werden kann. Es ist auch die Aufgabe der vorliegenden Erfindung, ein entsprechendes Werkzeug zur Verfügung zu stellen.

[0007] Diese Aufgabe wird zum einen durch ein Verfahren zum Ermitteln von Testfällen für ein System, insbesondere für ein Softwaresystem gelöst, mit den Schritten: Feststellen aller Einflussfaktoren auf eine erste Abstraktionsebene des Systems, wobei für jeden dieser Faktoren Äquivalenzklassenvertreter für alle möglichen Ausprägungen eines Faktors angegeben werden können; Zuordnen derjenigen Einflussfaktoren, die für eine sich anschließende, höhere oder niedrigere, zweite Abstraktionsebene relevant sind, zu dieser zweiten Ebene, wobei die Schritte des Feststellens und Zuordnens von Einflussfaktoren über die Abstraktionsebenen des Systems hinweg derart wiederholt werden, dass die jeweils zweite Ebene zur jeweils ersten Ebene wird, und Erstellen einer Testspezifikation für wenigstens eine Abstraktionsebene, wobei einzelne Testfälle der Spezifikation durch Kombinieren der Äquivalenzklassenvertreter der relevanten Einflussfaktoren erzeugt werden.

[0008] Das erfindungsgemäße Verfahren stellt damit eine Erweiterung der schon eingangs beschriebenen CTM dar. Diese Erweiterung durch die hier beschriebene Vorgehensweise besteht darin, dass im ersten Schritt zunächst alle Einflussfaktoren des Gesamtsystems festgestellt werden und im zweiten Schritt aus der Menge der Einflussfaktoren die für die Subsysteme relevanten ausgewählt werden. Dadurch werden logische Zustände der Software besonders gut abgedeckt.

[0009] Der Unterschied des Verfahrens zur CTM ergibt sich daraus, dass alle Einflussfaktoren bekannt sind und auf ihre Relevanz geprüft werden, wodurch ein reduziertes Risiko besteht, dass Einflussfaktoren vergessen werden. Sofern bei der Ermittlung der Testfälle oder beim Test eines Subsystems ein weiterer Einflussfaktor gefunden wird, kann dieser direkt in die Menge der Einflussfaktoren aufgenommen werden. Dadurch ist sichergestellt, dass dieser Einflussfaktor bei den anderen Subsystemen ebenfalls berücksichtigt wird, und nicht nur bei dem System, bei dem er gefunden wurde.

[0010] Der Unterschied des Verfahrens zur FMEA besteht dabei darin, dass bei der FMEA Risiken aufgedeckt werden sollen und daher Fehlerbäume und -abhängigkeiten betrachtet werden. In dem hier vorgestellten Verfahren sollen Testfälle gefunden wer-

den. Mit den Fehlern aus der FMEA liegt noch keine Information vor, wie Testfälle gefunden werden können, die letztendlich zum Entdecken von Fehlern führen. Dazu ist es erforderlich, die Systeme und Subsysteme mit einem anderen Fokus – den zu Testfällen kombinierbaren Einflussfaktoren – zu betrachten. Im Gegensatz zur FMEA sollen durch das Testen Fehler aufgedeckt werden, damit diese nicht mehr im Feld auftreten. Während bei der FMEA Fehler festgestellt werden, werden im hier vorgestellten Verfahren Einflussfaktoren ermittelt. Die Kombination von Einflussfaktoren kann zur Aufdeckung von Fehlern führen, die nur unter speziellen Umständen zu Tage treten und dadurch schwerer vorherzusehen sind. Den Fragestellungen der FMEA 'Welche Fehler breiten sich wie aus ?' und der Fehlerbaumanalyse 'Welche Fehler können entstehen ?' steht beim vorgestellten Ansatz die Frage 'Was hat Einfluss auf das Verhalten ?' im Vordergrund.

[0011] Die Zuordnung von Einflussfaktoren zu Subsystemen bewirkt dabei, dass die Vorgehensweise explizit – und nachvollziehbar – sichtbar macht, welche Einflussfaktoren sich auf welche Subsysteme auswirken. Das bedeutet zunächst, dass während der Testfallerstellung nicht über weitere Einflussfaktoren nachgedacht werden muss, da für die zu testenden Subsysteme die in den Testfällen berücksichtigten Einflussfaktoren bekannt sind. Der Rahmen für die Testfälle ist durch die Einflussfaktoren und deren Werte gegeben. Bei Änderungen der Einflussfaktoren sind zudem die betroffenen Subsysteme bekannt. Dies kann wiederum dazu verwendet werden, den zu erwartenden Testaufwand bei notwendigen Änderungen abzuschätzen. Weiterhin kann die Zuordnung von Einflussfaktoren auf Subsysteme bereits begleitend zum Systemdesign vorgenommen, und damit das System hinsichtlich des späteren Testaufwandes optimiert werden. Wird erkannt, dass der Einfluss einzelner Faktoren sich auf eine große Zahl von Subsystemen erstreckt, kann beispielsweise versucht werden, diesen Einfluss durch eine Änderung der Systemstruktur auf einen kleineren Teil des Systems zu beschränken. So kann die später erforderliche Zahl von Testfällen bereits im Rahmen des Designs reduziert werden (Design for Testability).

[0012] Bei der Entwicklung einer Softwarekomponente wurde das Verfahren zum Erzeugen der Tests verwendet. Es hat sich gezeigt, dass durch die Abdeckung der logischen Zustände eine sehr gute Testabdeckung erreicht werden kann. Diese bezieht sich sowohl auf den Kontrollfluss als auch auf den Datenfluss. Die gute Testabdeckung ist eine Konsequenz aus der Abdeckung logischer Zustände. Dabei wurden einerseits keine Einflussfaktoren vergessen und andererseits im Code nicht mehr implementiert ist als benötigt.

[0013] Wie vorstehend beschrieben, berücksichtigt

das Verfahren die Struktur des zu testenden Softwaresystems und beginnt damit einen Schritt früher, d. h. bei der Menge der Einflussfaktoren für alle zu testenden Subsysteme. Die Testfälle für das zu testende System werden dabei grundsätzlich in drei Schritten ermittelt, auf die nachfolgend detailliert eingegangen werden soll.

[0014] Nach dem Design des Systems werden in einem ersten Schritt zunächst die Einflussfaktoren auf das System festgestellt. Als Einflussfaktoren werden alle Dinge bezeichnet, die einen Einfluss auf das System haben. Dies sind unter anderem die Parameter für das System, Variablen aus der Umgebung des Systems, Ergebnisse der Interaktion des Systems mit seiner Umwelt wie Rückgabewerte von Subsystemen, Vorbedingungen für die Verwendung des Systems und/oder interne Zustände. Für jeden dieser Einflussfaktoren werden Vertreter für die Äquivalenzklassen aller möglichen Ausprägungen des Einflussfaktors angegeben. Durch die Betrachtung aller Möglichkeiten für den Einflussfaktor können beispielsweise die Grenzwerte – die gerade noch zulässigen und gerade nicht mehr zulässigen Werte – als die geeigneten Vertreter ausgewählt werden. Die Einflussfaktoren werden mit dem Wissen über die Anforderungen an das System und die Kenntnis der internen Logik des Systems identifiziert. Bei der Feststellung wird auch die Kenntnis der internen Struktur berücksichtigt. Die Aufteilung des Systems in Subsysteme beeinflusst die Menge der Einflussfaktoren. Durch die Strukturierung entstehen neue Schnittstellen zwischen dem System und den Subsystemen. An diesen Schnittstellen entsteht zum Beispiel ein Informationsaustausch z. B. durch die Rückgabewerte verwendeter Funktionen und Dienste, der ebenfalls als Einflussfaktor zu berücksichtigen ist.

[0015] Im nächsten Schritt erfolgt die Zuordnung von Einflussfaktoren zu Subsystemen. Dabei werden den einzelnen Subsystemen wie z. B. Komponenten, Modulen, Funktionen usw. des zu testenden Systems diejenigen Einflussfaktoren aus der im ersten Schritt festgestellten Gesamtmenge von Einflussfaktoren zugeordnet, die für das jeweilige Subsystem relevant sind. Dabei ist es möglich, dass einzelne Einflussfaktoren für mehrere Subsysteme relevant sind.

[0016] Im dritten Schritt wird schließlich eine Identifikation der Testfälle vorgenommen. Dabei wird für jedes Subsystem eine Testspezifikation erstellt. Die Testfälle ergeben sich aus den relevanten Einflussfaktoren für das zu testende Subsystem. Jede Kombination der Äquivalenzklassenvertreter der relevanten Einflussfaktoren führt zu einem Testfall. Die Testspezifikation enthält Testfälle, die detailliert den zu testenden Sachverhalt beschreiben, jedoch vom verwendeten Testwerkzeug abstrahieren.

[0017] Tests werden für das System aber auf ver-

schiedenen Abstraktionsebenen benötigt, z. B. für das System, dessen Subsysteme, wiederum deren Subsystem bis hin zu den kleinsten zu testenden Subsystemen. Dabei existieren Abhängigkeiten der Einflussfaktoren auf das System bis hin zu den Einflussfaktoren auf die kleinsten zu testenden Subsysteme.

[0018] Ein wesentlicher Punkt des vorgestellten Verfahrens besteht deshalb in seiner Rekursivität, bei der die gefundenen Einflussfaktoren auf die Subsysteme des Systems heruntergebrochen werden, und auch eine Spezialisierung eines heruntergebrochenen Einflussfaktors für das entsprechende Subsystem durchgeführt werden kann. Die Einflussfaktoren werden für das zu testende System analog zu der Struktur des Systems ermittelt. Dazu werden die ersten beiden Schritte für das System durchgeführt. Wenn die Einflussfaktoren identifiziert und den Subsystemen zugeordnet worden sind, können die ersten beiden Schritte auf diese Subsysteme rekursiv angewendet werden. Dies setzt sich fort bis hin zu den kleinsten zu testenden Einheiten. Dazu müssen die Einflussfaktoren für das jeweilige Subsystem in der gleichen Weise auf dessen Subsysteme heruntergebrochen werden. Die ersten beiden Schritte des Verfahrens können dabei für die Subsysteme durchgeführt werden, bis schließlich die kleinsten zu testenden Einheiten erreicht sind.

[0019] Obwohl das erfindungsgemäße Verfahren – und Werkzeug – an Hand eines Softwaresystems vorgestellt und erläutert werden, soll sich der hier verwendete Begriff 'System' nicht nur auf ein Softwaresystem beschränken. Das erfindungsgemäße Vorgehen und seine werkzeugseitige Unterstützung sind grundsätzlich auf jedes System anwendbar, ob es sich dabei um ein kombiniertes Software- und Hardwaresystem, oder ein reines Hardwaresystem handelt.

[0020] Im Allgemeinen kommen dabei mit jedem Verfeinerungsschritt weitere Einflussfaktoren hinzu. Diese weiteren Einflussfaktoren resultieren sowohl aus der Umwelt und dem Design des Subsystems, als auch aus den – auf dieses Subsystem heruntergebrochenen – Anforderungen. Auf unterster Ebene hinzukommende Einflussfaktoren sind beispielsweise Rückgabewerte anderer Systemteile (aus der Umwelt) oder Eigenschaften der verwendeten Datenstrukturen (aus dem Design). Durch das Verfahren ist die Struktur der Einflussfaktoren von gleicher Gestalt wie das System, was insofern sinnvoll ist, als die Tests für die entsprechenden Subsysteme benötigt werden. Für jede der zu testenden Einheiten, vom gesamten System bis hin zu den kleinsten zu testenden Subsystemen, können für die Subsysteme die jeweiligen Testfälle ermittelt, d. h. der Schritt des Erstellens der Testspezifikation durchgeführt werden. Dies kann auf verschiedenen Abstraktionsniveaus ge-

schehen. So ist es beispielsweise möglich, lediglich Tests für das ganze System und die kleinsten zu testenden Einheiten zu erzeugen. Für alle Abstraktionsebenen dazwischen müssen die Testfälle nicht erzeugt werden. Natürlich kann das Verfahren auf allen Abstraktionsebenen zwischen einem Gesamtsystem und den kleinsten identifizierbaren Einheiten des Systems wie z. B. Funktionen beginnen und enden. Mit anderen Worten stellt sich zu Beginn des Verfahrens zunächst die Frage, was das zu testende System ist. An dieser Stelle setzt das Verfahren an. Für die kleinsten zu testende Einheiten kann es dann beispielsweise erforderlich sein, Komponententests zu erstellen. Sollten Tests der in den Komponenten enthaltenen Funktionen nicht gewünscht sein, kann das Verfahren auf Komponentenebene abgebrochen werden.

[0021] Das Verfahren muss dabei nicht notwendigerweise top-down, d. h. von oben nach unten, angewendet werden, sondern kann beispielsweise auch bottom-up, d. h. von unten nach oben durchlaufen werden. In diesem Fall müssen möglicherweise manche Einflussfaktoren für einen Systemteil auf einer höheren Ebene verallgemeinert werden oder entfallen, wenn sie auf dieser Ebene nicht sichtbar sind. Werden vergessene Einflussfaktoren nachträglich gefunden, können diese mit Hilfe des Verfahrens durch alle Ebenen propagiert werden, also sowohl nach oben als auch nach unten.

[0022] Bei beiden Ansätzen, also dem top-down wie dem bottom-up-Ansatz, muss der Startpunkt des rekursiven Verfahrens nicht notwendigerweise auf der obersten (top-down) oder auf der untersten Systemebene (bottom-up) angesiedelt sein, sondern kann auch auf einer Systemebene liegen, die zwischen diesen Extremen angesiedelt ist. Wo allerdings auch immer das Verfahren startet, ist auch eine kombinierte Strategie des zunächst nach oben/unten und anschließend nach unten/oben Laufens möglich, so dass die betroffenen Ebenen unter wechselnden Abhängigkeiten mehrfach betrachtet und damit die Einflussfaktoren komplettiert werden.

[0023] Neben den schon vorstehend erwähnten grundsätzlichen Vorteilen steht damit ein durchgängiges Verfahren zur Ermittlung von Testfällen auf Basis der Einflussfaktoren zur Verfügung. Es reicht von Systemtests bis hin zu Tests für die kleinsten separat zu testenden Subsysteme. Die Einflussfaktoren werden dabei beginnend beim System bis zum kleinsten zu testenden Subsystem heruntergebrochen. Durch die Wiederverwendung der Einflussfaktoren kann der Gesamtaufwand gegenüber der separaten Ermittlung der Testfälle reduziert werden.

[0024] Die Durchgängigkeit des Verfahrens vermindert das Risiko, dass Testfälle vergessen werden, da die Testfälle für alle Abstraktionsebenen eines Sys-

tems auf der gleichen Datenbasis basieren. Die Einflussfaktoren werden von den Anforderungen unter Berücksichtigung des Designs und der Implementierung schrittweise heruntergebrochen. Sofern Einflussfaktoren anfänglich vergessen und nachträglich gefunden worden sind, können die relevanten Systemteile durch Propagierung der Einflussfaktoren direkt identifiziert werden. Nachträglich gefundene Einflussfaktoren können sowohl eine Auswirkung nach unten (Übernahme oder Konkretisierung) als auch nach oben (Übernahme oder Abstraktion) haben. Dadurch reduziert sich das Risiko des Vergessens von Einflussfaktoren noch weiter.

[0025] Ein zusätzlicher Vorteil des Verfahrens besteht darin, dass das Verfahren frühzeitig im Entwicklungsprozess, d. h. parallel zur Entwicklung des Systems angewendet werden kann. Es lässt sich damit gut in Entwicklungsprozesse integrieren. In der Anfangsphase arbeiten wenige Mitarbeiter an der groben Strukturierung des Systems. Dabei werden für die bearbeiteten Systemteile Einflussfaktoren identifiziert und möglicherweise Testfälle erzeugt. Wenn in einer späteren Entwicklungsphase bei der weiteren Strukturierung des Systems mehr Mitarbeiter involviert sind, können diese auf diesen Einflussfaktoren aufsetzen, diese verfeinern (nach unten propagieren), abstrahieren (nach oben propagieren) und ergänzen. Somit kann das Verfahren schon beginnende mit der Designphase des Systems das Erstellen von Tests unterstützen. Die Struktur des Systems und der Einflussfaktoren und Testfälle sind gleichförmig.

[0026] Bei der Ermittlung wird auch die Kenntnis der internen Struktur des Systems berücksichtigt. Die Aufteilung des Systems in Subsysteme beeinflusst die Menge der Einflussfaktoren. Durch die Strukturierung entstehen neue Schnittstellen zwischen dem System und den Subsystemen. An diesen Schnittstellen entsteht zum Beispiel ein Informationsaustausch (Rückgabewerte verwendeter Funktionen und Dienste), der ebenfalls als Einflussfaktor zu berücksichtigen ist.

[0027] Ein weiterer Vorteil besteht zudem darin, dass bereits während der Designphase anhand der Einflussfaktoren erkennbar wird, dass bestimmte Strukturierungen einen hohen Testaufwand nach sich ziehen werden. Bereits in einer frühen Phase der Entwicklung können Alternativen anhand der Einflussfaktoren evaluiert werden und Designentscheidungen entsprechend getroffen werden. Das Verfahren unterstützt deshalb ein Design for Testability. Während der Designphasen auf unterschiedlichen Abstraktionsebenen wird durch die Ermittlung der Einflussfaktoren und deren Zuordnung zu kleineren Einheiten dem Designer ein direktes Feedback gegeben. Es ist sofort ersichtlich, wenn Information nicht richtig gekapselt ist oder Verantwortlichkeiten über

mehrere Einheiten verteilt ist. Das Verfahren kann also auch ausgenutzt werden, um die Struktur der gerade betrachteten Einheit zu verbessern. Dies kann ebenfalls zu besserer Wartbarkeit und Portabilität führen.

[0028] Testfälle müssen auch nur für zu testende Einheiten erstellt werden. Der dritte und vierte Schritt des Verfahrens muss nur für die Einheiten durchgeführt werden, die auch wirklich getestet werden sollen. Das Verfahren hängt nicht von der Erstellung der Testfälle ab, sondern nur von der Ermittlung und Zuordnung von Einflussfaktoren.

[0029] Oft werden zugelieferte Systemteile oder eigenentwickelte Komponenten in ein System integriert. Für diese ist es ebenfalls möglich, Einflussfaktoren in das Verfahren einzubringen, um so die gleichförmige Struktur der Einflussfaktoren und Testfälle mit der Systemstruktur zu erhalten.

[0030] Bevorzugte Weiterbildungen des Verfahrens sind in den Unteransprüchen 2 bis 8 angegeben.

[0031] Danach ist in einer bevorzugten Ausführungsform des Verfahrens vorgesehen, dass die solchermaßen ermittelten abstrakten Testfälle in konkrete Testfälle umgesetzt werden, die durch ein Testwerkzeug ausführbar sind. Für diesen Schritt ist es erforderlich, die im dritten Schritt erhaltenen Testfälle in die Sprache des Testwerkzeugs umzusetzen. Damit werden potentielle Fehlerquellen ausgeschlossen und der Ersteller von Testfällen durch eine (Teil)Automatisierung seiner Tätigkeit unterstützt. Die (wiederverwendbare) Testspezifikation ist zudem hinreichend abstrakt, so diese alle wesentlichen Informationen für die Tests enthält. Damit kann die Spezifikation als Grundlage zur Umsetzung durch verschiedene Testwerkzeuge verwendet werden. Damit ist es gelungen, die Spezifikation der Testfälle von deren Implementierung zu trennen. Sofern beim Komponententest essentielle Codeteile nicht durchlaufen werden, ist dies ein Indiz für das Fehlen mindestens eines Einflussfaktors. Damit ist eine Plausibilisierung der Menge von Einflussfaktoren möglich. Nicht essentielle Codeteile in diesem Sinne sind z. B. Code für andere Varianten als die gerade betrachtete und Code zur Unterstützung der Entwicklung wie beispielsweise die Reaktion des Systems auf entdeckte Inkonsistenzen. Es kann sich aber um 'toten Code' handeln, der nie zur Ausführung kommen kann.

[0032] Verfahrensbedingt kann es unter Umständen vorkommen, dass ein oder mehrere Einflussfaktoren für manche Tests irrelevant sind, wodurch Duplikate von Testfällen entstehen. Bevorzugt ist es deshalb, wenn Duplikate von Testfällen erkannt und eliminiert werden. Dies ist bereits in der Testspezifikation möglich. Dadurch kann die Anzahl von Testfällen reduziert und die Tests können somit beschleunigt durch-

geführt werden.

[0033] Die Einflussfaktoren entstehen bevorzugt aus den Anforderungen an das Softwaresystem und/oder durch das Treffen von Entscheidungen während des Komponentendesigns und der Implementierung. Aus diesem Grund kann sich die Menge der für den Test relevanten Einflussfaktoren im Laufe der Entwicklungszeit verändern, gleichzeitig aber auch aktuell gehalten werden.

[0034] Den Einflussfaktoren und/oder den Testfällen wird vorzugsweise ein Kommentar zugeordnet. Damit ist auch die Dokumentation und Nachvollziehbarkeit von doppelten Testfällen und deren Elimination z. B. in einer Tabelle gegeben.

[0035] Bevorzugt ist es weiterhin, wenn bei dem Verfahren die ausgeführten Schritte und/oder die Ausgangspunkte für die Einflussfaktoren dokumentiert werden. Damit ist die Durchführung einzelner Schritte nachvollziehbar, insbesondere bei der Entwicklung komplexer Produkte mit vielen Entwicklern.

[0036] Wegen der Nachvollziehbarkeit ist es bevorzugt, wenn Versionen unterschiedlicher Testspezifikationen angelegt werden. Dies bedeutet aber nicht, dass das Werkzeug eine eigene Versionsverwaltung benötigt, sondern im Idealfall Vergleichsmöglichkeiten zur Verfügung stellen muss. Im einfachsten Fall sind Kommentare der im Projekt eingesetzten Versionsverwaltung ausreichend.

[0037] Das hier vorgestellte Verfahren soll bevorzugt zum Testen von Softwaresystemen für Steuergeräte von Fahrzeugen Verwendung finden, deren sichere Funktion bei gleichzeitig hoher Komplexität sichergestellt werden muss.

[0038] Die eingangs genannte Aufgabe wird auch durch ein Werkzeug zum Ermitteln von Testfällen für ein Softwaresystem gelöst, das zur automatisierten Unterstützung des erfindungsgemäßen Verfahrens angepasst ist.

[0039] Ein wesentlicher Punkt des erfindungsgemäßen Werkzeugs besteht insbesondere darin, dass Fehler bei der Durchführung der einzelnen Schritte und bei deren Interaktion ausgeschlossen sind. Dadurch wird das Risiko verringert, dass Testfälle vergessen werden. Gleichzeitig wird der Ersteller der Testfälle entlastet und das Testverfahren insgesamt beschleunigt.

[0040] Bevorzugte Weiterbildungen des Erfindungsgemäßen Werkzeugs sind in den Unteransprüchen 9 bis 16 angegeben, bei denen die automatisierte Unterstützung einzelner oder aller Verfahrensschritte sowie die Möglichkeit im Vordergrund steht, eine nachvollziehbare Dokumentation der Testfälle zu er-

möglichen. Beides erleichtert eine sichere und schnelle Durchführung des Verfahrens genauso wie die zuverlässige Verwaltung der Testfälle. In einem adäquaten Datenmodell zur Umsetzung des Verfahrens lässt sich eine Hierarchie der Abstraktionsebenen grundsätzlich durch ein verhältnismäßig einfaches Modell beschreiben. Für jeden Systemteil lassen sich die Einflussfaktoren mittels einer Tabelle zuordnen. Eine Verfeinerung kann über Verweise zwischen den Tabellen vorgenommen werden.

Kurze Beschreibung der Zeichnungen

[0041] Das erfindungsgemäße Verfahren wird im folgenden anhand eines Ausführungsbeispiels näher erläutert. Gleiche oder gleichwirkende Teile sind mit gleichen Bezugszeichen versehen. Es zeigen:

[0042] [Fig. 1](#) ein Softwaresystem zum Verwalten von nichtflüchtigen Daten über einen Dispatcher in zwei unterschiedlichen Speichertypen;

[0043] [Fig. 2](#) die erfindungsgemäße Zuordnung von Einflussfaktoren zu den Subsystemen des Systems der [Fig. 1](#);

[0044] [Fig. 3](#) die erfindungsgemäße Identifikation der Testfälle für ein beispielhaft gewähltes Subsystem des Systems der [Fig. 1](#), und

[0045] [Fig. 4](#) Abstraktionsebenen des Systems der [Fig. 1](#) zur Verdeutlichung des erfindungsgemäßen Verfahrens.

Ausführungsformen der Erfindung

[0046] Die [Fig. 1](#) zeigt ein – stark vereinfachtes – Softwaresystem **10** zum Verwalten von nichtflüchtigen Daten über einen Dispatcher **11** in zwei unterschiedlichen Speichertypen, einem EEPROM-Dateisystem **12** und einem Flash-Dateisystem **13**. Das System **10** wird auch als Persistent Data Manager (PDM) bezeichnet. Zwei Anwendungen A1 (Application 1) und A2 (Application 2) können die Daten der Subsysteme **12** und **13** lesen, schreiben und löschen. Dazu stehen die Operationen read, write und delete zur Verfügung. Außerdem gibt es eine Funktion init zur Initialisierung des Systems. Das zu testende System **10** besteht dabei aus drei Komponenten. Jede dieser Komponenten besitzt die vier oben erwähnten Funktionen. In [Fig. 1](#) ist eine statische Sicht auf die drei Komponenten dargestellt.

[0047] Die Verantwortlichkeiten der zu testenden Komponenten werden im folgenden zunächst beschrieben.

[0048] Der Dispatcher **11** nimmt die Aufträge der Anwendungen A1 und A2 entgegen und speichert sie in einer Auftragswarteschlange einer festen Größe,

sorgt dafür, dass die Aufträge `init`, `read`, `write` und `delete` an die richtige Dateisystem-Komponente weitergegeben werden, stellt sicher, dass zu jedem Zeitpunkt jedes Dateisystem nur höchstens einen Auftrag bearbeitet, und sorgt dafür, dass nur die zur Integrationszeit als gültig konfigurierten Daten an die entsprechenden Dateisystem-Komponenten weitergegeben werden.

[0049] Die beiden Dateisystem-Komponenten **12** und **13** nehmen die vom Dispatcher **11** erhaltenen Aufträge entgegen und setzen die Aufträge `init`, `read`, `write` und `delete` mit Kenntnis der entsprechenden Speicher-Hardware um.

[0050] In dem ersten Schritt des Verfahrens werden für das Beispiel der [Fig. 1](#) nun exemplarisch einige Einflussfaktoren **20** festgestellt. Diese sollen in einem zweiten Schritt den zu testenden Subsystemen **11**, **12** und **13**, d. h. den Funktionen der Komponenten Dispatcher, EEPROM-Speicher und Flash-Speicher zugeordnet werden. In einem dritten Schritt sollen schließlich einige abstrakte Testfälle für den Dispatcher **11** aufgeführt werden.

[0051] Die [Fig. 2](#) zeigt Abstraktionsebenen des Systems der [Fig. 1](#) zur Verdeutlichung des erfindungsgemäßen Verfahrens die erfindungsgemäße Zuordnung der Einflussfaktoren **20** zu den Subsystemen **11**, **12** und **13** des Systems **10** der [Fig. 1](#), die für das jeweilige Subsystem relevant sind. Die Einflussfaktoren **20** und die Vertreter der Äquivalenzklassen **21** sind in der [Fig. 2](#) in den Spalten 'Einflussfaktoren' und 'Werte für Test' zu finden. In diesem Schritt werden auch die separat testbaren Funktionen `init`, `read`, `write` und `delete` der drei Komponenten aufgeführt. Die Einflussfaktoren **20** aus dem ersten Schritt werden den Funktionen, also den zu testenden Subsystemen **11**, **12** und **13**, zugeordnet. Ein 'x' in einer Zelle der Tabelle bedeutet, dass ein Einflussfaktor **20** (in einer Zeile) für eine Funktion (in einer Spalte) relevant ist. Zum Beispiel muss für die Funktion `init` der Komponente Dispatcher der Einflussfaktor 'Dateisystem result' berücksichtigt werden. Es ist ebenfalls erkennbar, dass beispielsweise für die Funktion `read` des Dispatchers Eigenschaften von Daten wie die 'Länge des Datums' oder die 'Länge des gespeicherten Datums' keinen Einfluss haben und somit auch für den Test dieser Funktion nicht berücksichtigt werden müssen. Die zuvor dargestellte Tabelle ist bezüglich dieses Aspekts bereits vollständig ausgefüllt.

[0052] Die [Fig. 3](#) zeigt die erfindungsgemäße Identifikation von Testfällen für ein Subsystem des Systems **10** der [Fig. 1](#). Dabei sollen die Testfälle TF für zwei Funktionen `init` und `read` des Dispatchers **11** ermittelt werden. Dazu ist es erforderlich, alle Kombinationen der Werte der relevanten Einflussfaktoren **20** aufzuführen. Ein implizites Zusammenfassen von Testfällen TF ist in der Tabelle bereits durchgeführt

worden. Beispielsweise ist bei 'Test 5' die ID (Identification) beliebig und das 'Dateisystem result' nicht relevant. Daher wurden nicht alle möglichen acht Kombinationen dieser Einflussfaktoren **20** aufgeführt, um sie im Anschluss wieder zusammenzuführen. Für diesen Testfall ist lediglich relevant, dass kein Auftrag mehr angenommen werden kann. Jedoch wurden beispielsweise die vier Testfälle mit 'siehe Test 5' in der Kommentarspalte **22** nicht zusammengefasst. Darüber hinaus ist an den Testfällen TF der Funktion `read` erkennbar, dass Duplikate eliminiert werden können. Diese Duplikate sind in der Tabelle daran erkennbar, dass sie in der ersten Spalte keine Identifikation als Testfall TF besitzen und in einer Kommentarspalte **22** einen Verweis auf den Testfall, der ersten abdeckt. Als Beispiel wird eine Leseanforderung bei voller Queue betrachtet. Wenn vom Dispatcher kein weiterer Auftrag mehr entgegen genommen werden kann, wird 'read fehler' zurückgegeben. Dies ist unabhängig von der ID. Daher können die Testfälle mit 'siehe Test 5' in der Kommentarspalte **22** und der Testfall 'Test 5' zusammengefasst werden. Für das Zusammenfassen von Testfällen TF und die Elimination von Duplikaten ist es erforderlich, den Code des entsprechenden Subsystems zu kennen. Für die Elimination von Duplikaten im letzten Aufzählungspunkt war es beispielsweise notwendig, die Logik der Funktion `read` zu kennen. Die Überprüfung, ob die Warteschlange voll ist, erfolgt vor der Überprüfung der Korrektheit der ID.

[0053] Die Implementierung der Testfälle TF aus einer derart erzeugten Testspezifikation S (Specification) in einem anschließenden – vierten Schritt – setzt das Vorhandensein eines Testwerkzeuges voraus und hängt stark von dem Testwerkzeug ab. Das Verfahren kann aber in seiner werkzeugimplementierten Form schon eine entsprechende Schnittstelle aufweisen.

[0054] Die [Fig. 4](#) zeigt Abstraktionsebenen des Systems **10** der [Fig. 1](#) zur Verdeutlichung des erfindungsgemäßen Verfahrens. Wie schon beschrieben, besteht der PDM aus den drei Subsystemen, einem Dispatcher und zwei Dateisystemen. Diese wiederum besitzen jeweils die vier Funktionen `init`, `read`, `write` und `delete`. Auf der obersten Abstraktionsebene AL1 (Abstraction Layer 1) sind die Einflussfaktoren **20** auf das System **10** angegeben, die sich aus den spezifischen Systemanforderungen RS (Requirements System), der Umwelt ES (Environment System) und dem Design DS (Design System) ergeben. Aus den Einflussfaktoren **20** können entsprechende Tests TS (Test System) abgeleitet werden. Auf der mittleren Abstraktionsebene AL2 sind entsprechend die Einflussfaktoren **20** auf die Komponenten, also den Dispatcher und die beiden Dateisysteme angegeben, die wiederum von den Systemanforderungen RC1 (Requirement Component) an diese Komponenten, der Umwelt EC1 und dem Design DC1 her-

rühren. Wiederum aus den Einflussfaktoren **20'** können entsprechende Tests TC ermittelt werden. Auf der untersten Ebene ALn schließlich sind die Einflussfaktoren **20''** auf die Funktionen init, read, write und delete wiedergegeben, resultierend aus den Systemanforderungen RF1 (Requirement Function), der Umwelt EF1 und dem Design DF1. Generell können die Abstraktionsebenen AL2 bis ALn, also die unterhalb der Systemebene liegenden Ebenen, mehrere Komponenten umfassen, die den entsprechenden Einflussfaktoren **20'** und **20''** unterliegen. Einflussfaktoren auf Komponenten einer Ebene ergeben sich dabei nicht nur aus den jeweiligen Systemanforderungen, der Umwelt und dem Design einer Komponente, sondern umfassen auch die Einflüsse, die sich aus der funktionalen Interaktion der Komponenten dieser Ebene ergeben. Testfälle können dabei jeweils für alle oder einzelne Komponenten innerhalb einer Ebene erzeugt werden.

[0055] Anhand der [Fig. 4](#) ist erkennbar, dass Einflussfaktoren auf eine Einheit aus folgenden Gründen entstehen können. Zum einen aus Anforderungen an die Einheit, die sich gegebenenfalls aus heruntergebrochenen Anforderungen an das Gesamtsystem ergeben. Zum anderen aus der Umwelt der Einheit, die sowohl durch Design übergeordneter Systemteile entstanden sein kann als auch aus der Systemumwelt resultiert. Anforderungen ergeben sich aber auch aus dem Design der Einheit. Weitere Anforderungen ergeben sich aus der Propagierung von Einflussfaktoren, die heruntergebrochen und gegebenenfalls spezialisiert werden. Ein Beispiel für einen heruntergebrochenen Einflussfaktor kann das Vorhandensein eines Datums in einem Dateisystem sein. Ob das Datum vorhanden ist oder nicht, ist dabei für den gesamten PDM testrelevant, aber auch für die Dateisysteme. Ein Beispiel für einen spezialisierten Einflussfaktor kann darin bestehen, dass beim Dispatcher getestet werden muss, ob ein Datum gelesen werden kann oder nicht. Bei den Dateisystem-Komponenten ist der Test von verschiedenen Ursachen für fehlgeschlagene Lesevorgänge erforderlich. Anforderungen ergeben sich aber auch aus Einflussfaktoren, die nach oben propagiert und gegebenenfalls abstrahiert werden. Beispiele dafür ergeben sich analog zu den vorstehend genannten.

[0056] Die [Fig. 4](#) veranschaulicht das Ergebnis des erfindungsgemäßen Verfahrens gemäß einem top-down-Ansatz. Ein Rechteck mit durchgezogenem Rahmen steht für die Durchführung und das Ergebnis der ersten beiden Schritte des Verfahrens, also der Feststellung der Einflussfaktoren und der Zuordnung zu den zu testenden Einheiten. Das Rechteck mit gestricheltem Rahmen steht für die Durchführung der beiden letzten Schritte des Verfahrens, also Schritt drei und vier, und als Ergebnis die erzeugten Testfälle TS (Test System), TC (Test Component) und TF (Test Function). Die Pfeile stehen für

ein Eingehen der jeweiligen Artefakte in die entsprechenden Schritte des Verfahrens, und sind mit BDS (Break-Down and Specialization) bezeichnet. Im Falle eines bottom-up-Ansatzes des Verfahrens würden die Pfeile nach oben deuten und eine Propagierung und Abstraktion der Einflussfaktoren in die Schritte des Verfahrens bezeichnen.

[0057] Anhand der [Fig. 4](#) ist erkennbar, dass für die ersten beiden Schritte sowohl die Anforderungen an diese Einheit, im Fall des Dispatchers und der zwei Dateisysteme z. B. RC1, EC1 und DC1, als auch die Einflussfaktoren **20** der nächst höheren Ebene AL1 eingehen, die sich wiederum aus RS, ES und DS ergeben. Eine Ausnahme ist die höchste Ebene AL1, das zu testende System **10**. Die jeweiligen Anforderungen werden auf die entsprechenden Einheiten heruntergebrochen und können zu Einflussfaktoren **20'**, **20''** führen. Soweit die Feststellung der Einflussfaktoren **20'**, **20''** aus den Anforderungen RS, RC1, RF1, der Umwelt ES, EC1, EF1 und dem Design DS, DC1, DF1 der Einheit nicht automatisiert durchgeführt werden können, sollte dieser Schritt in einem kleinen Team durchgeführt werden und/oder durch ein Review überprüft werden. Die Einflussfaktoren der nächst höheren Ebene können übernommen bzw. müssen angepasst werden. Weiterhin ist erkennbar, dass die Tests TS, TC und TF aus der Zuordnung der Einflussfaktoren zu den zu testenden Einheiten entstehen und keine weitere Information benötigt wird. Dieser Schritt muss nur dann ausgeführt werden, wenn Tests tatsächlich benötigt werden. Auf den Tests einer bestimmten Einheit basieren keine weiteren Schritte des Verfahrens, deshalb ist es ohne Nachteile möglich, diese nicht zu ermitteln.

[0058] Entsprechend dem vorgestellten Verfahren würde die Tabelle zur Ermittlung der Einflussfaktoren **20 ... 20''** und Zuordnung zu Subsystemen der [Fig. 2](#) und [Fig. 3](#) konzeptionell in mehrere Tabellen für die unterschiedlichen Ebenen AL1 ... ALn zerfallen. Die Testspezifikation S, also die Tabelle mit den abstrakten Testfällen T, wäre für jede Funktion zu erstellen, in diesem Beispiel init, read, write und delete. Zur Unterstützung dieses Vorgehens kann ein Werkzeug entsprechende Übersichten, d. h. Zusammenfassungen von Tabellen auf mehreren Hierarchieebenen AL1 ... Mn darstellen, z. B. für die Tabelle der [Fig. 2](#).

[0059] Wie aus dem geschilderten Beispiel erkennbar ist, lässt das erfindungsgemäße Verfahren eine durchgängige und zuverlässige Feststellung von Einflussfaktoren zum Ermitteln von Testfällen zu, ist zudem schnell durchführbar und einfach strukturiert, somit auch kostengünstig implementierbar. Da das Verfahren zudem ganz oder teilweise automatisierbar ist, wird eine rasche Testfallermittlung möglich.

Patentansprüche

1. Verfahren zum Ermitteln von Testfällen (TS, TC, TF) für ein System, insbesondere für ein Softwaresystem (**10**), mit den Schritten:

Feststellen aller Einflussfaktoren (**20**) auf eine erste Abstraktionsebene (AL1) des Systems, wobei für jeden dieser Faktoren (**20**) Äquivalenzklassenvertreter (**21**) für alle möglichen Ausprägungen eines Faktors (**20**) angegeben werden können;

Zuordnen derjenigen Einflussfaktoren (**20'**), die für eine sich anschließende, höhere oder niedrigere, zweite Abstraktionsebene (AL2) relevant sind, zu dieser zweiten Ebene (AL2), wobei die Schritte des Feststellens und Zuordnens von Einflussfaktoren (**20, 20'**) über die Abstraktionsebenen (AL1 ... ALn) des Systems (**10**) hinweg derart wiederholt werden, dass die jeweils zweite Ebene (AL2) zur jeweils ersten Ebene (AL1) wird, und Erstellen einer Testspezifikation (S) für wenigstens eine Abstraktionsebene (AL1 ... ALn), wobei einzelne Testfälle (TS, TC, TF) der Spezifikation (S) durch Kombinieren der Äquivalenzklassenvertreter (**21**) der relevanten Einflussfaktoren (**20, 20', 20''**) ermittelt werden.

2. Verfahren nach Anspruch 1, bei dem die solchermaßen ermittelten abstrakten Testfälle (TS, TC, TF) in konkrete Testfälle umgesetzt werden, die durch ein Testwerkzeug ausführbar sind.

3. Verfahren nach Anspruch 1 oder 2, bei dem Duplikate von Testfällen (TS, TC, TF) in der Testspezifikation (S) erkannt und eliminiert werden.

4. Verfahren nach einem der vorstehenden Ansprüche, bei dem die Einflussfaktoren (**20, 20', 20''**) aus den Anforderungen an das Softwaresystem (**10**) und/oder durch das Treffen von Entscheidungen während des Komponentendesigns und der Implementierung entstehen.

5. Verfahren nach einem der vorstehenden Ansprüche, bei dem den Einflussfaktoren (**20, 20', 20''**) und/oder den Testfällen (TS, TC, TF) ein Kommentar (**22**) zugeordnet wird.

6. Verfahren nach einem der vorstehenden Ansprüche, bei dem die ausgeführten Schritte und/oder die Ausgangspunkte für die Einflussfaktoren (**20, 20', 20''**) dokumentiert werden.

7. Verfahren nach einem der vorstehenden Ansprüche, bei dem Versionen von unterschiedlichen Testspezifikationen (S) angelegt werden.

8. Verwendung des Verfahrens nach einem der vorstehenden Ansprüche zum Testen von Softwaresystemen für Steuergeräte von Fahrzeugen.

9. Werkzeug zum Ermitteln von Testfällen (TS, TC, TF) für ein Softwaresystem (**10**), das zur automatisierten Unterstützung des Verfahrens nach einem der vorstehenden Ansprüche ausgebildet ist.

10. Werkzeug nach Anspruch 9, das zur automatisierten Unterstützung des Erstellens der Testspezifikation (S) für jede Abstraktionsebene (AL1 ... ALn) ausgebildet ist.

11. Werkzeug nach einem der Ansprüche 9 oder 10, das zum automatischen Zuordnen relevanter Einflussfaktoren (**20, 20', 20''**) zu den einzelnen Abstraktionsebenen (AL1 ... ALn) ausgebildet ist.

12. Werkzeug nach einem der Ansprüche 9 bis 11, das zum Speichern der Einflussfaktoren (**20, 20', 20''**) und deren jeweiliger Äquivalenzklassenvertreter (**21**) ausgebildet ist.

13. Werkzeug nach einem der Ansprüche 9 bis 12, das zum Speichern eines Kommentars (**22**) zu den Einflussfaktoren (**20, 20', 20''**) und/oder den Testfällen ausgebildet ist.

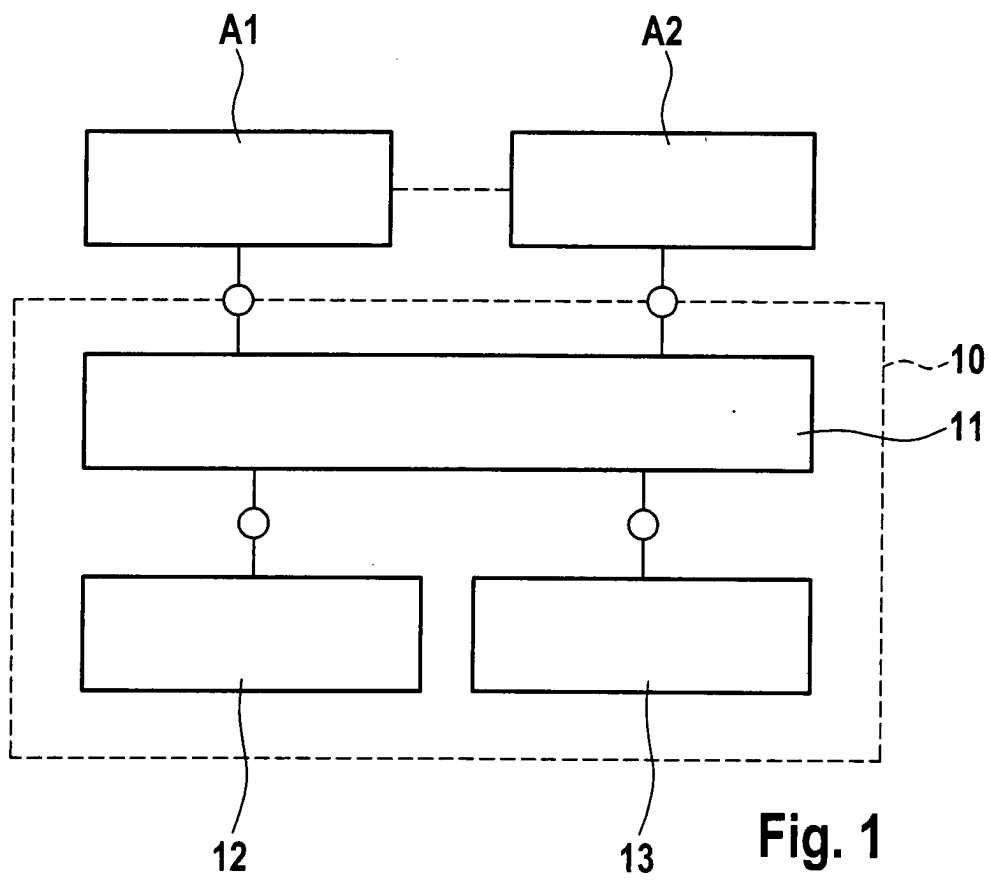
14. Werkzeug nach einem der Ansprüche 9 bis 13, das zum Dokumentieren der ausgeführten Schritte und/oder der Ausgangspunkte für die Einflussfaktoren (**20, 20', 20''**) ausgebildet ist.

15. Werkzeug nach einem der Ansprüche 9 bis 14, das eine Versionsverwaltung für Testspezifikationen (S) aufweist.

16. Werkzeug nach einem der Ansprüche 9 bis 15, das eine Schnittstelle zum Umsetzen der abstrakten Testfälle (TS, TC, TF) in konkrete Testfälle für ein Testwerkzeug aufweist.

Es folgen 4 Blatt Zeichnungen

Anhängende Zeichnungen



10				11				12				13				20'				21				22			
Dispatcher				EEPROM Dateisystem				Flash Dateisystem				Dateisystem				Einflussfaktoren				Werte f. Test				Kommentar			
Init	read	write	delete	Init	read	write	delete	Init	read	write	delete	Init	read	write	delete					ja, nein		(0, 1, max, max+1)		Zul.:(1,...,max)			
																Queue voll											
	x	x	x													ID											
	x	x	x													Dateisystem result											
x																Fehler Checksumme Verw.											
																Fehler Checksumme Daten											
																Datum vorhanden											
																Länge Datum											
																Länge gespeichertes Datum											
																Freier Speicher vorhanden											
																Umorganisation abgebrochen											

Fig. 2

S

Testfälle Dispatcher		Queue voll	ID	Dateisystem result	Ergebnis	Kommentar
Testfälle init						
Dateisystem result						
Test 1	EEPROM=ok,Flash=ok			EEPROM=ok,Flash=ok	init ok	
Test 2	EEPROM=ok,Flash=fehler			EEPROM=ok,Flash=fehler	init fehler	
Test 3	EEPROM=fehler,Flash=ok			EEPROM=fehler,Flash=ok	init fehler	
Test 4	EEPROM=fehler,Flash=fehler			EEPROM=fehler,Flash=fehler	init fehler	
Testfälle read						
Queue voll						
Test 5	ja	ja	beliebig	nicht relevant	read fehler	
Test 6	nein	nein	beliebig	EEPROM=ok	read ok	
ID						
0		ja	0	nicht relevant	read fehler	siehe Test 5
0		nein	0	nicht relevant	read fehler	
0		nein	0	EEPROM=fehler	read fehler	siehe Test 11
1		ja	1	nicht relevant	read fehler	siehe Test 5
1		nein	1	EEPROM=ok	read ok	
1		nein	1	EEPROM=fehler	read fehler	siehe Test 11
maxID		ja	maxID	nicht relevant	read fehler	siehe Test 5
maxID		nein	maxID	EEPROM=ok	read ok	
maxID		nein	maxID	EEPROM=fehler	read fehler	siehe Test 11
maxID+1		ja	maxID+1	nicht relevant	read fehler	siehe Test 5
maxID+1		nein	maxID+1	nicht relevant	read fehler	
maxID+1		nein	maxID+1	EEPROM=fehler	read fehler	siehe Test 11
Dateisystem result						
EEPROM=ok		nein	beliebig	EEPROM=ok	read ok	siehe Test 7
EEPROM=fehler		nein	beliebig	EEPROM=fehler	read fehler	
Test 11						

TF

Fig. 3

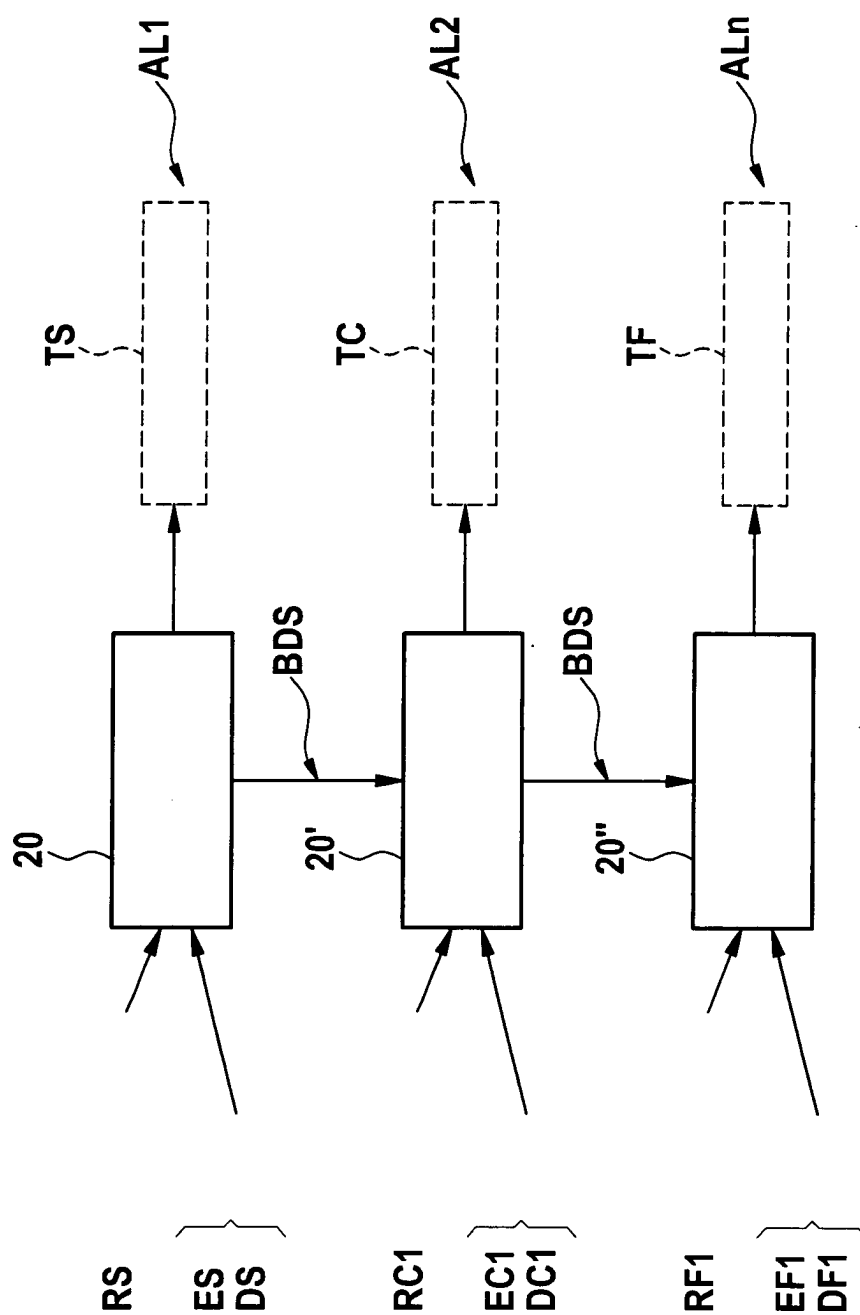


Fig. 4