(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 10 2006 047 762 A1** 2008.04.10

(12) **Offenlegungsschrift**

(21) Aktenzeichen: **10 2006 047 762.6**
(22) Anmeldetag: **06.10.2006**
(43) Offenlegungstag: **10.04.2008**

(51) Int Cl.[8]: ***H04L 12/26*** (2006.01)
H04L 12/28 (2006.01)

(71) Anmelder:
Siemens AG, 80333 München, DE

(72) Erfinder:
Meissner, Thomas, 91074 Herzogenaurach, DE

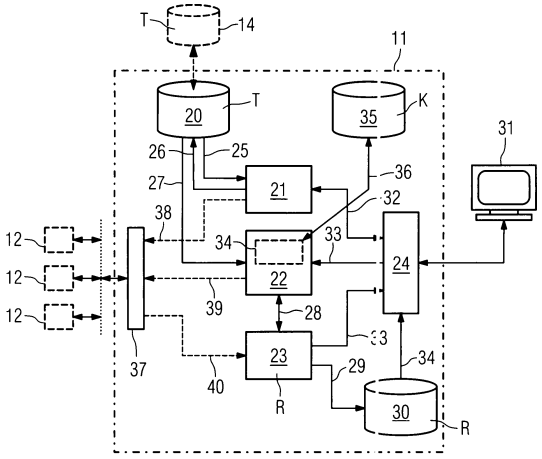
(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:
DE10 2005 031245 A1
JP 06-0 35 823 A

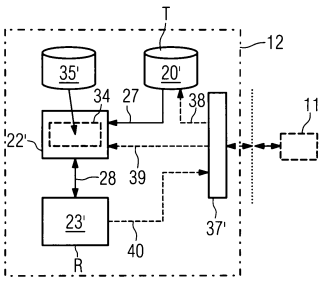
Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gemäß § 44 PatG ist gestellt.

(54) Bezeichnung: **System zum Testen der Funktion eines Computernetzwerkes**

(57) Zusammenfassung: Es wird ein System zum Testen der Funktion eines Computernetzwerkes angegeben, das auf effektive und einfach handhabbare Weise eine Fehlererkennung und Fehleranalyse in einem Computernetzwerk ermöglicht. Das System (10) umfasst eine Testdatenbank (14, 20), in der eine Anzahl von Testszenarien (T) ablegbar oder abgelegt sind, und ein Diagnose-Framework (11). Hierbei umfasst das Diagnose-Framework (11) eine Laufzeitumgebung (22), auf der das oder jedes Testszenario (T) ausführbar ist, eine Auswerteeinheit (23), die die Ausführung des oder jedes Testszenarios (T) überwacht und einen Testbericht (R) zur Anzeige für einen Benutzer erstellt, sowie einen Testeditor (21), mittels welchem zur Ausführung auf der Laufzeitumgebung (22) geeignete Testszenarien (T) erstellbar sind.





Beschreibung

[0001] Die Erfindung bezieht sich auf eine System zum Testen der Funktion eines Computernetzwerkes. Als Computernetzwerk wird eine lokal begrenzte Struktur von zum Datenaustausch miteinander vernetzten Rechnern, insbesondere ein so genanntes LAN (Local Area Network) bezeichnet. Ein Computernetzwerk in diesem Sinne ist insbesondere auch ein so genanntes PACS (Picture Archiving and Communication System), wie es im klinischen Bereich zur arbeitsteiligen Erzeugung, Bearbeitung und Auswertung medizinischer Bilddaten verwendet wird.

[0002] Ein solches Computernetzwerk umfasst in der Regel eine Vielzahl von datenübertragungstechnisch miteinander verbundenen Netzwerkknoten, wobei jeder Knoten durch einen Rechner realisiert ist. Der Begriff "Rechner" ist hierbei allgemein zu verstehen und umfasst neben Servern für Daten und Datenverarbeitungsdienste, Befundungsstationen oder sonstigen Arbeitsstationen insbesondere auch computergestützte medizinische Untersuchungsgeräte (Modalitäten) wie z.B. Computertomographie (CT)-Scanner, Ultraschall-Scanner, Röntgenbildaufnahmegeräte, Magnetresonanztomographen, etc.

[0003] Ein solches Computernetzwerk neigt mit zunehmender Größe und Komplexität in steigendem Maße zu Instabilitäten und Systemausfällen, die die Funktion des gesamten Netzwerkes oder einzelner Netzwerkbereiche mitunter erheblich einschränken oder sogar ganz zum Erliegen bringen können. Solche Fehlfunktionen können vielfältige Ursachen haben. Neben Hardware-Defekten kommt insbesondere eine Fehlkonfiguration von Verbindungsparametern, Software-Fehler oder eine Inkompatibilität der im Netzwerk zusammenwirkenden Hardware- und/oder Softwarekomponenten in Frage.

[0004] Eine solche Inkompatibilität tritt besonders häufig auf, wenn nicht aufeinander abgestimmte Hardware- bzw. Softwarekomponenten verschiedener Hersteller oder verschiedener Versionen in Kombination miteinander eingesetzt werden.

[0005] Die Ursache für eine derartige Fehlfunktion ist oft nur schwer zu finden, insbesondere zumal Ursache und Symptome einer Fehlfunktion häufig in keinem offensichtlichen Zusammenhang miteinander stehen. Beispielsweise kann eine Fehlkonfiguration einer beliebigen Arbeitsstation eines Krankenhauses unter ungünstigen Umständen dazu führen, dass der Datendurchsatz eines zentralen Servers drastisch herabgesetzt ist oder der Server komplett ausfällt. Eine vergleichbare Wirkung könnte aber beispielsweise auch dadurch hervorgerufen werden, dass ein an entscheidender Stelle angeordneter Netzwerkknoten zu wenig Arbeitsspeicher aufweist, etc.

[0006] Der Erfindung liegt die Aufgabe zu Grunde, ein System zum Testen der Funktion eines Computernetzwerkes anzugeben, das auf effektive und einfach handhabbare Weise eine Fehlererkennung und Fehleranalyse in einem Computernetzwerk ermöglicht.

[0007] Diese Aufgabe wird erfindungsgemäß gelöst durch die Merkmale des Anspruchs 1.

[0008] Danach sind im Rahmen des Systems eine Testdatenbank sowie ein Diagnose-Framework vorgesehen. Die Testdatenbank kann hierbei ein Bestandteil des Diagnose-Frameworks oder ein separater Systembestandteil sein. In der Testdatenbank sind eine Anzahl von Testszenarien, d.h. mindestens ein Testszenario, bevorzugt aber eine Vielzahl von Testszenarien, ablegbar oder abgelegt. Das Diagnose-Framework umfasst eine Laufzeitumgebung, auf der das oder jedes Testszenario ausführbar ist. Weiterhin umfasst das Diagnose-Framework eine Auswerteeinheit und einen Testeditor. Die Auswerteeinheit ist hierbei dazu ausgebildet, die Ausführung des oder jeden Testszenarios zu überwachen und auf Basis dieser Überwachung einen Testbericht in einer für einen Benutzer anzeigbaren Form, insbesondere als XML-Text, zu erstellen. Der Testeditor ist seinerseits dazu ausgebildet, die Erstellung neuer, zur Ausführung auf der Laufzeitumgebung geeigneter Testszenarien durch einen Benutzer zu unterstützen.

[0009] Das Diagnose-Framework und seine Bestandteile sind bevorzugt als Softwaremodule ausgebildet, die auf einem oder mehreren Knoten eines Computernetzwerkes installierbar oder installiert sind. Abweichend hiervon können das Diagnose-Framework und seine Bestandteile aber auch in Form einer Hardware mit darauf vorinstallierter Software, z.B. einer Steckkarte, ausgebildet sein.

[0010] Als "Testszenario" wird eine Softwareroutine bezeichnet, die eine bestimmte Funktion des Computer-Netzwerkes überprüft und einen von dem Ergebnis dieser Überprüfung abhängigen Ausgabewert zurückgibt, anhand dessen erkennbar ist, ob und gegebenenfalls in welchem Ausmaß der Test erfolgreich durchgeführt wurde oder ob ein Fehler aufgetreten ist. Optional spezifiziert der Ausgabewert im Fehlerfall auch die Art des aufgetretenen Fehlers näher. Der Ausgabewert kann hierbei wahlweise eine Textmeldung (z.B. "Test erfolgreich durchgeführt") oder ein Code-Zeichen enthalten.

[0011] Als "Framework" wird allgemein eine Anordnung von Softwaremodulen bezeichnet, die die Erstellung und Durchführung von konkreter Anwendungs-Software – hier den Testszenarien – unterstützt, und somit eine "Rahmen-Software" für die Anwendungs-Software bildet. Das Framework gibt hierbei die Softwarearchitektur für die Anwendungs-Soft-

ware vor und steuert deren Kontrollfluss.

[0012] Als "Laufzeitumgebung" wird eine Softwareschicht bezeichnet, die einer Anwendungs-Software – hier also einem durchzuführenden Testszenario – und dem Betriebssystem eines Netzwerkknotens zwischengelagert ist. Die Laufzeitumgebung stellt Grundfunktionen, die von der Anwendungs-Software benötigt werden, z.B. Lesen und Schreiben von Dateien, die Steuerung von Ein- und Ausgabegeräten, etc. zur Verfügung und übersetzt die häufig einem so genannten Zwischencode vorliegende Anwendungs-Software in den unmittelbar ausführbaren Maschinencode.

[0013] Indem im Rahmen des Diagnose-Frameworks ein spezieller Testeditor mit einer daran angepassten Laufzeitumgebung kombiniert ist, wird ermöglicht, Testszenarien auf besonders einfache Weise zu erstellen, zu verwalten und auszuführen. Diesen Zweck fördert auch die dem Diagnose-Framework weiterhin zugeordnete Auswerteeinheit, indem diese eine für alle Testszenarien vereinheitlichte Plattform zur Aufbereitung der von einem jeden Testszenario im Rahmen der Ausführung zurückgegebenen Ausgabewerte darstellt. Die Testszenarien selbst können daher vergleichsweise einfach aufgebaut sein und sind daher auch schnell und flexibel erstellbar. Auf Grund des von der Auswerteeinheit in Form des Testberichts ausgegebenen vereinheitlichten Testergebnisses ist zudem eine einfache und gut handhabbare Fehlersuche und -analyse ermöglicht.

[0014] Eine besonders einfache und intuitive Handhabung des Systems wird dadurch ermöglicht, dass das oder jedes Testszenario in einer zumindest im Wesentlichen einem Petri-Netz oder einem Aktivitätsdiagramm gemäß des UML2-Standards entsprechenden Form hinterlegt ist. Als "Petri-Netz" wird ein an sich bekanntes mathematisches Modell zur Modellierung von Systemen und Transformationsprozessen bezeichnet.

[0015] Ein Petri-Netz besteht allgemein aus so genannten Stellen (bzw. Plätzen/places) und so genannten Übergängen (bzw. Transitionen/transitions). Stellen werden dabei konventionsgemäß als Kreise dargestellt, Übergänge als rechteckige Balken. Stellen und Übergänge sind durch gerichtete "Kanten" verbunden, wobei es keine direkten Verbindungen zwischen zwei Stellen oder zwei Übergängen gibt.

[0016] Jede Stelle hat eine so genannte "Kapazität" und kann entsprechend viele so genannte "Marken" (tokens) enthalten. Ist eine Kapazität nicht angegeben, steht dies – je nach Konvention – für eine unendliche Kapazität oder eine Kapazität des Wertes eins. Jeder Kante ist ein Gewicht zugeordnet, das die so genannten "Kosten" dieser Kante festlegt. Ein Übergang gilt als "schaltbereit", wenn in allen Eingangsstellen mindestens so viele Marken vorliegen, wie der Übergang Kosten verursacht, und alle Ausgangsstellen noch genug Kapazität haben, um die neuen Marken aufzunehmen. Beim Schalten eines Übergangs werden aus den Ausgangsstellen entsprechend der Kantengewichte Marken entfernt, und bei den Ausgangsstellen entsprechend der Kantengewichte Marken hinzugefügt.

[0017] Für weitere Einzelheiten zu den Eigenschaften eines Petri-Netzes wird auf http://de.wikipedia.org/wiki/Petri-Netz in der Fassung vom 29. August 2006 - 11:44 Uhr Bezug genommen.

[0018] Ein so genanntes Aktivitätsdiagramm gemäß des UML2-Standards (Unified Modelling Language Version 2.x) weist einem Petri-Netz ähnliche Eigenschaften auf. Anstelle von Stellen bzw. Plätzen weist ein Aktivitätsdiagramm Aktionen auf, die konventionsgemäß als Rechtecke mit abgerundeten Ecken dargestellt werden. Anstelle der Übergänge sind im Rahmen eines Aktivitätsdiagramms so genannte Synchronisationsbalken vorgesehen. Die Notation eines Aktivitätsdiagramms gemäß UML2 ist gegenüber dem abstrakten mathematischen Modell eines Petri-Netzes flexibilisiert. So können im Rahmen eines Aktivitätsdiagramms – anders als bei einem klassischen Petri-Netz – zwei Aktionen auch direkt ohne zwischengeschalteten Übergang durch eine gerichtete Kante miteinander verbunden werden.

[0019] Als eine "einem Petri-Netz bzw. einem Aktivitätsdiagramm gemäß UML2 entsprechenden Form" wird jedes Datenformat bezeichnet, das eine eindeutige Abbildung der Bestandteile eines Testszenarios auf ein Petri-Netz bzw. ein Aktivitätsdiagramm gemäß UML2 ermöglicht. Insbesondere liegt ein Testszenario dann in einer einem Petri-Netz oder einem Aktivitätsdiagramm entsprechenden Form vor, wenn es in eine Anzahl von Bestandteilen (Aktionen bzw. Stellen) gegliedert ist, wenn zwischen diesen Bestandteilen gerichtete Datenübertragungsregeln (Kanten) und gegebenenfalls Verzweigungs- bzw. Übergangspunkte (Übergänge, Synchronisationsbalken) definiert sind, und wenn diese Bestandteile, Datenübertragungsregeln und Verzweigungs- bzw. Übergangspunkte den für ein Petri-Netz bzw. ein Aktivitätsdiagramm gemäß UML2 festgelegten Eigenschaften genügen. Eine einem Petri-Netz oder einem Aktivitätsdiagramm entsprechende Form kann insbesondere auch dann vorliegen, wenn eine von den für ein Petri-Netz oder ein Aktivitätsdiagramm gemäß UML2 geltenden Konventionen verschiedene Nomenklatur oder Darstellung gewählt ist.

[0020] In einer zweckmäßigen Ausbildung der Erfindung umfasst das System hierbei eine bevorzugt als Softwaremodul ausgeführte Anzeigeeinheit, die dazu ausgebildet ist, das oder jedes Testszenario in Form eines Petri-Netzes oder eines Aktivitätsdiagramms

gemäß UML2 graphisch darzustellen. Um eine besonders einfache und intuitive Erstellung von Testszenarien zu ermöglichen, wirkt die Anzeigeeinheit mit dem Testeditor dabei zweckmäßigerweise dahingehend zusammen, dass das oder jedes Testszenario graphisch, d.h. unmittelbar in Form einer Graphik, erstellbar ist. Zusätzlich oder alternativ hierzu wirkt die Anzeigeeinheit mit der Auswerteeinheit dahingehend zusammen, dass das aus der Ausführung eines beliebigen Testszenarios resultierende Testergebnis anhand der graphischen Darstellung dieses Testszenarios darstellbar ist, wodurch insbesondere ermöglicht wird, dass das Testergebnis für einen Benutzer auf einen Blick erfassbar ist. Zur Vereinfachung der Fehleranalyse ist dabei vorgesehen, dass etwaige Testfehler stellen- bzw. aktionsaufgelöst dargestellt werden. Hierunter wird verstanden, dass der oder jeder Testfehler in der graphischen Darstellung des zugehörigen Testszenarios derjenigen Stelle bzw. Aktion graphisch zugeordnet wird, die den Testfehler hervorgerufen hat. Eine derartige Zuordnung erfolgt in geeigneter Weise durch testergebnisabhängige Einfärbung der Stellen bzw. Aktionen in der graphischen Darstellung des Testszenarios. Beispielsweise werden Stellen bzw. Aktionen eines Testszenarios, die erfolgreich ausgeführt wurden, grün dargestellt, während Stellen bzw. Aktionen, die einen Testfehler hervorgerufen haben, rot markiert werden.

[0021] In einer vorteilhaften Ausgestaltung der Erfindung ist die Laufzeitumgebung derart ausgebildet, dass sie in einem Simulationsmodus betreibbar ist. Der Simulationsmodus zeichnet sich dadurch aus, dass die Parameter des Computernetzwerkes lediglich temporär änderbar sind. In dem Simulationsmodus wird zu diesem Zweck eine Kopie der Parameterkonfiguration des Computernetzwerkes oder eines für ein bestimmtes Testszenario relevanten Teils davon erstellt, an dem Änderungen vorgenommen werden können, ohne dass die tatsächliche Parameterkonfiguration des Computernetzwerkes davon berührt wird. Auf diese Kopie der Parameterkonfiguration können dann im Rahmen des Simulationsmodus die Testszenarien angewendet werden. Der Simulationsmodus erlaubt, die Auswirkung von Änderungen der Parameterkonfiguration gefahrlos durchzutesten, ohne die Integrität der bestehenden Netzwerkkonfiguration zu gefährden. Bei Beendigung des Simulationsmodus ist zweckmäßigerweise vorgesehen, dass nach Wunsch des Benutzers die simulierte Parameterkonfiguration, d.h. die im Rahmen des Simulationsmodus hinterlegte Kopie, entweder auf das reale Computernetzwerk übernommen oder verworfen werden kann. In letzterem Fall wird der vor dem Eintritt in den Simulationsmodus bestehende Konfigurationszustand des Computernetzwerkes wiederhergestellt.

[0022] In einer vorteilhaften Weiterbildung ist die Laufzeitumgebung dazu ausgebildet, dass im Rahmen des Simulationsmodus nicht nur die Parameterkonfiguration der realen Knoten des Computernetzwerkes veränderbar ist, sondern dass zusätzlich neue Netzwerkknoten simulierbar, d.h. virtuell erzeugbar sind. Auf diese Weise kann insbesondere eine geplante Netzwerkerweiterung getestet werden, ohne dass die für diese Netzwerkerweiterung erforderliche Hardware tatsächlich zur Verfügung stehen müsste. Im Extremfall kann auf diese Weise auf einem einzigen physikalischen Rechner ein ganzes Computernetzwerk simuliert werden.

[0023] Das Diagnose-Framework ist zweckmäßigerweise skalierbar ausgebildet in dem Sinne, dass es in beliebigen Umfang um neue Testszenarien erweiterbar ist. Neben der Möglichkeit, neue Testszenarien mittels des Testeditors selbst zu erstellen, ist hierfür vorzugsweise die Möglichkeit vorgesehen, neue Testszenarien aus einer globalen Testdatenbank zu importieren. Der Begriff "global" ist hierbei zunächst allgemein im Sinne von "netzwerkübergreifend" zu verstehen. Eine Testdatenbank ist in diesem Sinne global, wenn sie mehreren Computernetzwerken, beispielsweise also mehreren eigenständigen PACS-Systemen, gemeinsam zur Verfügung steht. In bevorzugter Ausbildung steht das Diagnose-Framework aber mit einer netzwerk-externen Datenbank in Verbindung, die im ursprünglichen Wortsinn "global", nämlich weltweit zugänglich ist. Auf diese Weise wird eine rasche Verbreitung geeigneter Testszenarien ermöglicht, die separate Eigenentwicklungen für gängige Probleme weitgehend überflüssig macht.

[0024] Im Sinne einer möglichst flexiblen Weiterentwicklung des vorhandenen Testszenarien-Pools – somit im Sinne der oben eingeführten Skalierbarkeit – ist vorzugsweise neben der Möglichkeit, neue Testszenarien von Grund auf neu zu erstellen, auch die Möglichkeit vorgesehen, bereits bestehende Testszenarien als Bestandteil, insbesondere als Stelle bzw. Aktion eines neuen Testszenarios, herangezogen werden kann. Auf diese Weise können insbesondere verschiedene bestehende Testszenarien zu einem übergeordneten Testszenario gruppiert werden. Ein solches übergeordnetes Testszenario kann dann wiederum Bestandteil eines weiter übergeordneten Testszenarios sein. optional ist vorgesehen, dass ein Testszenario rekursiv auch sich selbst als Bestandteil enthalten kann.

[0025] In einer bevorzugten Variante des Systems ist das Diagnose-Framework, wie vorstehend beschrieben, lediglich auf einem Knoten des Computernetzwerkes installiert bzw. installierbar. Dieses Diagnose-Framework stellt dabei ein Master-Framework dar, von dem aus die Überwachung des Computernetzwerkes gesteuert wird. Auf den anderen Knoten des Computernetzwerkes sind dagegen so genannte Agent-Frameworks installiert, die gegenüber dem Master-Framework einen reduzierten Funk-

tionsumfang aufweisen. Jedes der Agent-Frameworks enthält insbesondere eine Laufzeitumgebung zur Ausführung lokaler – d.h. nur für den jeweiligen Netzwerkknoten relevanter – Testszenarien, sowie eine Auswerteeinheit zur Überwachung der Ausführung solcher lokaler Testszenarien. Die einem jeden Agent-Framework zugewiesene Auswerteeinheit gleicht insofern der Auswerteeinheit des Master-Frameworks, als sie im Zuge dieser Überwachung ein Testergebnis erstellt. Dieses Testergebnis wird von dem Agent-Framework an das Master-Framework zurückgegeben und dort, gegebenenfalls zusammen mit dem von anderen Agent-Frameworks gelieferten Testergebnissen einen Benutzer angezeigt oder zur Anzeige angeboten.

[0026] Von dem Master-Framework aus ist das oder jedes Agent-Framework ansteuerbar, indem lokale Testszenarien von dem Master-Framework aus im Rahmen des zugehörigen Agent-Frameworks ausführbar ist.

[0027] Die vorstehend beschriebene Aufteilung des Systems in ein Master-Framework und ein oder mehrerer Agent-Frameworks hat sich als guter Kompromiss zwischen einer rein zentralen, d.h. lediglich von einem Knoten ausgehenden Softwarearchitektur, und einer rein dezentralen Softwarearchitektur mit gleichberechtigten Systemkomponenten auf jedem Knoten herausgestellt. Insbesondere wird durch diese eingeschränkt dezentrale Verteilung des Systems die Vorteile einer zentralen und einer dezentralen Softwarearchitektur synergetisch erzielt, nämlich ein vergleichsweise geringer Installations- und Synchronisationsaufwand, wie er für eine zentrale Architektur typisch ist, sowie ein effektiver und schneller Testablauf, wie er für eine dezentrale Architektur typisch ist.

[0028] Im Sinne einer weiter vereinfachten Handhabung des Systems ist die Laufzeitumgebung derart ausgebildet, dass das oder jedes Testszenario nach Maßgabe einer konfigurierbaren Auslösebedingung entweder manuell oder automatisch in vorgegebenen Zeitintervallen oder automatisch bei Eintritt eines vorgebbaren Systemzustandes, z.B. bei Auftreten eines bestimmten Netzwerkfehlers ausgeführt werden.

[0029] Für eine vereinfachte Ursachenforschung im Hinblick auf eine bestimmte Fehlfunktion ist optional dem oder jedem Testszenario eine Problemlösungs- (bzw. Trouble-Shooting-)Information zugewiesen oder zuweisbar. In diesem Fall ist die Auswerteeinheit zweckmäßigerweise dazu ausgebildet, die einem Testszenario zugeordnete Problemlösungsinformation anzuzeigen, wenn die Ausführung dieses Testszenarios zu einem Testfehler führt. Alternativ zu der direkten Anzeige der Problemlösungsinformation kann die Anzeige dieser Information dem Benutzer im Fehlerfall auch zunächst nur angeboten werden, z.B. indem dem Benutzer ein Dialogfeld oder Link angezeigt wird, in dem der Benutzer spezifizieren kann, ob er die Problemlösungsinformation sehen will. Liegen die Testszenarien in einem Petri-Netz oder Aktivitätsdiagramm gemäß UML2 entsprechenden Form vor, so ist die zugewiesene Problemlösungsinformation bevorzugt stellen- bzw. aktionsorientiert hinterlegt. Hierunter wird verstanden, dass zu jeder Stelle bzw. Aktion des Testszenarios ein separater Abschnitt der Problemlösungsinformation hinterlegt ist. Dem Benutzer wird dabei zweckmäßigerweise im Fehlerfall nur diejenige Problemlösungsinformation angezeigt, die derjenigen Stelle bzw. Aktion des Testszenarios zugeordnet ist, die den Testfehler hervorgerufen hat.

[0030] In bevorzugter Ausführung umfasst das System zumindest eine (erste) Supporteinrichtung, die bei einer Funktionsstörung, die von einem Benutzer nicht selbstständig beseitigt werden kann, eine Lösung zur Beseitigung der Funktionsstörung sucht. Die Auswerteeinheit des Diagnose-Frameworks ist dabei dazu ausgebildet, im Falle eines negativen Testverlaufs den zugehörigen Testbericht automatisch oder auf Bestätigung eines Benutzers hin an die erste Supporteinrichtung zu senden. Wird durch die Supporteinrichtung eine Lösung für die Beseitigung der gemeldeten Funktionsstörung gefunden, so übermittelt die erste Supporteinrichtung diese Lösung an das Diagnose-Framework, und damit an den Benutzer zurück. Zusätzlich oder alternativ hierzu ist vorgesehen, dass die Supporteinrichtung die Funktionsstörung entsprechend der von ihr gefundenen Lösung auf dem Wege einer Fernwartung selbstständig behebt. Bei der Supporteinrichtung kann es sich um ein automatisiertes, computergestütztes Expertensystem, ein menschliches Expertengremium oder um eine Mischung aus beidem handeln.

[0031] In vorteilhafter Ausgestaltung umfasst das System ein mehrstufiges Supportkonzept, bei dem zusätzlich zu der ersten Supporteinrichtung eine oder mehrere einander jeweils übergeordnete Supporteinrichtungen vorgesehen sind. Hierbei wird jeweils, wenn die erste bzw. eine untergeordnete Supporteinrichtung keine Lösung zur Beseitigung eines negativen Testergebnisses zur Verfügung stellen kann, das Testergebnis an eine übergeordnete Supporteinrichtung gesendet, die ihrerseits eine Lösung zur Beseitigung der Funktionsstörung sucht. Die übergeordnete Supporteinrichtung wird hierbei wahlweise von der direkt untergeordneten Supporteinrichtung oder direkt von der Auswerteeinheit aus angerufen. Zweckmäßigerweise sind die Supporteinrichtungen entsprechend ihres hierarchischen Rangs zunehmend zentralisiert, so dass mehrere untergeordnete Supporteinrichtungen jeweils einer übergeordneten Supporteinrichtung zugeordnet sind. Dieses baumartige Beziehungsgeflecht der verschiedenrangigen Supporteinrichtungen sorgt für ein besonders effektives Problemmanagement, im Rahmen dessen das Gros

der einfachen oder gängigen Fehlfunktionen von der vergleichsweise hohen Anzahl untergeordneter Supporteinrichtungen abgearbeitet wird, während lediglich komplexere Problemstellungen, deren Lösung erfahrungsgemäß vergleichsweise viel Erfahrung und Zeit kostet, die aber erfahrungsgemäß lediglich in geringer Häufigkeit auftreten, zu den hierfür spezialisierten übergeordneten Supporteinrichtungen gelangen.

[0032] Anliegend wird ein Ausführungsbeispiel der Erfindung anhand einer Zeichnung näher erläutert. Darin zeigen:

[0033] Fig. 1 ein an ein übergeordnetes Netzwerk, z.B. das Internet, angeschlossenes Computernetzwerk mit einer Anzahl von Netzwerkknoten, mit einem System zum Testen der Funktion des Computer-Netzwerkes, umfassend ein auf einem Netzwerkknoten installierten Diagnose-Framework als Master-Framework und jeweils ein auf jedem weiteren Netzwerkknoten installiertes Agent-Framework,

[0034] Fig. 2 in einem schematischen Blockschaltbild das Diagnose-Framework gemäß Fig. 1, mit einer Testdatenbank mit darin abgelegten Testszenarien, einer Laufzeitumgebung, einer Auswerteeinheit sowie einem Testeditor,

[0035] Fig. 3 in Darstellung gemäß Fig. 2 ein Agent-Framework gemäß Fig. 1, mit einer Testdatenbank, einer Laufzeitumgebung sowie einer Auswerteeinheit,

[0036] Fig. 4 in Form eines Aktivitätsdiagramms gemäß UML2 ein Beispiel für ein Testszenario,

[0037] Fig. 5 in Darstellung gemäß Fig. 4 ein weiteres Beispiel für ein Testszenario,

[0038] Fig. 6 in einem schematischen Flussdiagramm ein von dem Diagnose-Framework ausgeführtes Verfahren zum Testen der Funktion des Computer-Netzwerkes unter Ausführung eines Testszenarios, und

[0039] Fig. 7 in schematischer Darstellung eine globale Datenbank zur weltweiten Hinterlegung und Zurverfügungstellung von Testszenarien, sowie die funktionale Wechselwirkung dieser globalen Datenbank mit mehreren Computernetzwerken gemäß Fig. 1.

[0040] Einander entsprechende Teile und Größen sind in allen Figuren stets mit den gleichen Bezugszeichen versehen.

[0041] In Fig. 1 ist ein Computernetzwerk **1** schematisch dargestellt, wie es im medizinischen Bereich in einer modernen Klinik eingesetzt wird. Bei dem Computernetzwerk **1** handelt es sich um ein so genanntes PACS, das der vernetzen Archivierung und Bearbeitung und Auswertung medizinischer Bilddaten, z.B. Röntgenbildern dient.

[0042] Das Computernetzwerk **1** umfasst als Netzwerkknoten mehrere Server-Rechner und Client-Rechner. Als Server-Rechner umfasst das Computernetzwerk **1** insbesondere einen so genannten Operationsmanager-Server **2**, der zentrale Funktionen und Netzwerkdienste wie DNS (Domain Name System) oder DHCP (Dynamic Host Configuration Protocol)netzwerkweit zur Verfügung stellt.

[0043] Der Operationsmanagement-Server **2** ist datenübertragungstechnisch mit einem Datenmanagement-Server **3** verbunden, der Funktionen zur Archivierung von medizinischen Bilddaten zur Verfügung stellt. Der Datenmanagement-Server **3** korrespondiert hierbei mit einem Speicher **4** zur Hinterlegung der Bild- und Textdaten. Der Speicher **4** ist (in nicht näher dargestellter Weise) gegliedert in einem Kurzzeitspeicher, einen Archivspeicher sowie einen Back-up-Speicher, wobei letzterer zur redundanten Sicherung der Bild- und Textdaten gegen einen Datenverlust dient. Letzterer kann auch durch ein auswechselbares Speichermedium, z.B. einen Bandspeicher, realisiert sein.

[0044] Der Operationsmanagement-Server **2** korrespondiert weiter mit einem so genannten Radiologieinformationssystem **5** (kurz: RIS). Das Radiologieinformationssystem **5** beinhaltet in an sich bekannter Weise insbesondere Funktionen zur Verwaltung von Patientenstammdaten, zur Terminplanung von radiologischen Untersuchungen, zur Dokumentation medizinischer Daten nach den Anforderungen der Röntgenverordnung und zur Dokumentation von abrechnungsrelevanten Leistungen. Das Radiologieinformationssystem **5** stellt weiterhin eine Schnittstelle zu digitalen Untersuchungsgeräten (z.B. einem Computertomographen oder einem Magnetresonanztomographen) bereit und unterstützt die Erstellung von radiologischen Befunden.

[0045] Als Client-Rechner umfasst das Computernetzwerk **1** eine Vielzahl von Arbeitsstationen **6**, deren jede über einen Netzwerkbus **7** mit dem Operationsmanagement-Server **2** datenübertragungstechnisch verbunden ist.

[0046] Das Computer-Netzwerk **1** ist zudem über den Operationsmanagement-Server **2** datenübertragungstechnisch mit dem Internet **8** als übergeordnetes Netzwerk verbunden.

[0047] Zum Testen der Funktion des Computernetzwerkes **1** sowie ggf. für eine einfache und effektive Fehleranalyse und – beseitigung ist nun ein System **10** vorgesehen. Dieses System umfasst netzwerkintern ein Diagnose-Framework **11**, das in Form einer

Software auf dem Operationsmanagement-Server **2** installiert ist. Das System **10** umfasst netzwerkintern weiterhin eine Anzahl von Agent-Frameworks **12**, von denen jeweils eines ebenfalls in Form einer Software auf jedem weiteren Netzwerkknoten, d.h. insbesondere dem Datenmanagement-Server **3**, dem Radiologiesystem **5** und jeder Arbeitsstation **6** installiert ist. Das Diagnose-Framework **11** ist hierbei den Agent-Frameworks **12** übergeordnet und steuert zentral von dem Operationsmanagement-Server **2** aus deren Betrieb. Das Diagnose-Framework **11** wirkt in funktioneller Hinsicht somit im Sinne dieser Überordnung als Master-Framework **13** für die untergeordneten Agent-Frameworks **12**.

[0048] Das System **10** umfasst als netzwerkexterne Komponenten ferner eine globale Datenbank **14**, auf deren – nachfolgend näher beschriebenen Inhalt – das Diagnose-Framework **11** über das Internet **8** insbesondere weltweit zugreifen kann. Als weitere netzwerkexterne Komponente umfasst das System **10** mindestens eine, bevorzugt mehrere Supporteinrichtungen **15**, mit denen das Diagnose-Framework **11** ebenfalls über das Internet **8** in datenübertragungstechnischer Verbindung steht.

[0049] Der innere Aufbau des Diagnose-Frameworks **11** ist in Fig. 2 näher dargestellt.

[0050] Danach umfasst das Diagnose-Framework **11** insbesondere eine lokale Testdatenbank **20**, einen Testeditor **21**, eine Laufzeitumgebung **22**, eine Auswerteeinheit **23** sowie eine Anzeigeeinheit **24**.

[0051] In der lokalen Testdatenbank **20** sind – lokal im Rahmen des Operationsmanagement-Servers **2** – eine Anzahl von Testszenarien T hinterlegt. Das oder jedes Testszenario T ist eine Softwareroutine, mit der eine oder mehrere Funktionen des Computer-Netzwerkes oder eines oder mehrerer bestimmter Netzwerkknoten getestet werden kann.

[0052] Die hinterlegten Testszenarien T umfassen insbesondere

– Funktionen zum Testen der Datenverbindung zu einem Netzwerkknoten oder zum Testen der Datenübertragungsgeschwindigkeit,
– Funktionen zum Testen der ordnungsgemäßen Funktion eines Netzwerkknotens oder einer auf diesem installierten Software,
– Funktionen zum Überprüfen der Installationspfade von Softwarekomponenten und Konfigurationsdateien,
– Funktionen zur Ermittlung der Version eines in dem Computer-Netzwerk **1** installierten Softwarebestandteils,
– Funktionen zum Testen verschiedener Hardware- oder Softwarebestandteile oder verschieden versionierter Softwarebestandteile auf Kompatibilität,
– Funktionen zur Überprüfung von Lizenzen auf Gültigkeit,
– Funktionen zur Überprüfung des Zustandes des Arbeitsspeichers oder Festspeichers eines Netzwerkknotens,
– Funktionen zur Überprüfung der Arbeitsgeschwindigkeit eines Softwarebestandteils, etc.

[0053] Die lokale Testdatenbank **20** steht im bidirektionalem Datenaustausch mit der globalen Testdatenbank **14**, in der weitere Testszenarien T hinterlegt sind. Im Zuge dieses Datenaustausches können einerseits Testszenarien T, die von dem Diagnose-Framework **11** benötigt werden, aber in der lokalen Testdatenbank **20** nicht enthalten sind, aus der globalen Testdatenbank **14** heruntergeladen werden. Andererseits werden Testszenarien T, die auf lokaler Ebene neu erstellt werden, durch das Diagnose-Framework **11** an die globale Testdatenbank **14** hochgeladen.

[0054] Um die lokale Testdatenbank **20** nicht mit Information zu überfrachten und so ein schnelles und einfaches Arbeiten mit dem Diagnose-Framework **11** zu ermöglichen, ist die lokale Testdatenbank **20** insbesondere nach Art eines so genannten Cache-Speichers dazu ausgebildet, die Testszenarien T lediglich, z.B. für eine bestimmte Zeitdauer, zwischenzuspeichern. Dies bewirkt, dass lediglich die häufig oder zuletzt benutzten Testszenarien T lokal vorgehalten werden, während selten oder nur einmalige benutzte Testszenarien T nach einer gewissen Zeit wieder aus der lokalen Testdatenbank **20** gelöscht werden. In einer weiteren Variante des Diagnose-Frameworks **11** kann die lokale Testdatenbank **20** auch ganz entfallen. In diesem Fall korrespondieren die Komponenten des Diagnose-Frameworks **11** direkt mit der globalen Testdatenbank **14**.

[0055] Der Testeditor **21** unterstützt die Erstellung neuer Testszenarien T durch einen Benutzer. Der Testeditor **21** stellt dabei insbesondere Funktionen zur Verfügung, mittels derer Objekte oder Bestandteile eines neuen Testszenarios T – im Folgenden als Aktion A des Testszenarios T bezeichnet (s. Fig. 4 und Fig. 5) – erzeugt und ablauf- und datenübertragungstechnisch miteinander verknüpft werden können. Der Testeditor **21** ermöglicht dabei insbesondere, Testszenarien T, die in einer der Datenbanken **14** oder **20** hinterlegt sind, als Aktion A eines neuen, übergeordneten Testszenarios T heranzuziehen. Der Testeditor **21** greift hierzu auf die lokale Datenbank **20** sowie hierüber auf die globale Datenbank **14** zu (Beziehung **25**). Im Gegenzug bietet der Testeditor **21** die Möglichkeit, neu erstellte Testszenarien in der lokalen Testdatenbank **20** abzulegen (Beziehung **26**).

[0056] Zur Ausführung können die in den Testdatenbanken **20** und **14** hinterlegten Testszenarien der Laufzeitumgebung **23** zugeführt werden (Beziehung **27**). Die Ausführungen des oder jeden Testszenarios

T wird dabei (gemäß Beziehung **28**) überwacht von der Auswerteeinheit **23**, die Ausgabewerte des jeweils ausgeführten Testszenarios T sammelt, analysiert und anhand dieser Information einen Testbericht R, insbesondere in einem XML-Format, erstellt. Die Testberichte R werden (gemäß Beziehung **29**) von der Auswerteeinheit **23** einer Berichtdatenbank **30** zur Archivierung zugeführt. Die Auswerteeinheit **23** vermerkt hierbei auch Änderungen, die im Rahmen des Diagnose-Frameworks **11** an der Konfiguration des Computernetzwerkes **1** vorgenommen werden. Die Auswerteeinheit führt diese Änderungen einem in der Berichtdatenbank **30** hinterlegten History-Archiv zu, anhand dessen die an der Konfiguration des Computernetzwerks **1** vorgenommenen Änderungen zu einem späteren Zeitpunkt nachvollzogen werden können.

[0057] Zur Interaktion mit einem Benutzer wirken der Testeditor **21**, die Laufzeitumgebung **22** und die Auswerteeinheit **23** mit der Anzeigeeinheit **24** zusammen, die ein auf einem Bildschirm **31** anzeigbare graphische Benutzeroberfläche (GUI) zur Verfügung stellt. Die Anzeigeeinheit **24** ist hierbei insbesondere dazu ausgebildet,

- in Zusammenwirkung mit dem Testeditor **21** ein im Rahmen des Testeditors **21** zu erstellendes Testszenario T anzuzeigen und Steuerbefehle eines Benutzers zur Erstellung oder Sicherung des erstellten Testszenarios T an diesen zu übertragen (Beziehung **32**),
- in Zusammenwirkung mit der Laufzeitumgebung **22** ein Testszenario aus einer der Datenbanken **20** oder **14** zur Ausführung auszuwählen und sonstige Steuerbefehle zu der Ausführung des Testszenarios T an die Laufzeitumgebung **22** zu übertragen (Beziehung **33**),
- in Zusammenwirkung mit der Auswerteeinheit **23** den von Letzterer erstellten Testbericht R anzuzeigen (Beziehung **33**). Die Anzeigeeinheit **24** ist gleichermaßen geeignet, auch die in dem Berichtspeicher **30** archivierten Testberichte R zu laden und anzuzeigen (Beziehung **34**).

[0058] Der Testeditor **21**, die Laufzeitumgebung **22**, die Auswerteeinheit **23** und die Anzeigeeinheit **24** können – wie dargestellt – in separaten Softwaremodulen implementiert sein. Eine oder mehrere der zugehörigen Funktionalitäten können aber auch in einem gemeinsamen Softwaretool zusammengefasst sein.

[0059] Die Laufzeitumgebung **22** ist wahlweise in einem Real-Modus und einem Simulations-Modus **34** betreibbar. Im Real-Modus wird das jeweils innerhalb der Laufzeitumgebung **22** ausgeführte Testszenario T auf die realen Netzwerkknoten des Computernetzwerkes **1** und deren reale Parametereinstellungen angewendet. Änderungen an den Parametereinstellungen beeinflussen die reale Konfiguration des Computernetzwerkes **1**.

[0060] Dagegen wird in dem Simulationsmodus **34** eine Kopie K der Konfigurationsparameter des Computer-Netzwerkes **1**, oder zumindest einer relevanten Auswahl dieser Konfigurationsparameter, erstellt, und das im Simulationsmodus **34** ausgeführte Testszenario T auf diese Kopie K angewendet. Die Kopie K der Konfigurationsparameter wird in einer der Laufzeitumgebung **22** beigeordneten Simulationsdatenbank **35** temporär hinterlegt und aus dieser Simulationsdatenbank **35** zur Verfügung gestellt (Beziehung **36**), solange die Laufzeitumgebung **22** in dem Simulationsmodus **34** betrieben wird. Die Erstellung der Kopie K und die Anwendung der ausgeführten Testszenarien T auf diese Kopie K hat den Vorteil, dass Änderungen an der Parameterkonfiguration des Computer-Netzwerkes **1** im Simulationsmodus **34** gefahrlos getestet werden können. Eine versehentliche Fehlkonfiguration des Computer-Netzwerkes **1** kann aufgrund der lediglich temporären Speicherung der Kopie K durch Verlassen des Simulationsmodus **34** einfach verworfen werden, ohne die ursprüngliche Parameterkonfiguration und damit die Integrität des Computer-Netzwerkes **1** zu gefährden. Führt dagegen ein im Simulationsmodus **34** durchgeführtes Testszenario T zu einem positiven Ergebnis hinsichtlich einer in der Kopie K geänderten Parameterkonfiguration, so kann die Kopie K auf entsprechende Bestätigung des Benutzers beim Verlassen des Simulationsmodus **34** anstelle der ursprünglichen Parameterkonfiguration des Computer-Netzwerkes **1** übernommen werden.

[0061] Jedes der Agent-Frameworks **12** ist – wie aus Fig. 3 erkennbar ist – eine "abgespeckte" Version des Diagnose-Frameworks **11** mit gegenüber diesem verringerten Funktionsumfang. Jedes Agent-Framework **12** umfasst – wie das Diagnose-Framework **11** – eine lokale Testdatenbank **20'**, die zur Speicherung von lokalen Testszenarien T vorgesehen ist. Die lokalen Testszenarien T testen die Funktion des jeweils zugehörigen Netzwerkknotens. Jedes Agent-Framework **12** umfasst weiterhin eine Laufzeitumgebung **22'** und eine Auswerteeinheit **23'**. Die Laufzeitumgebung **22'** und die Auswerteeinheit **23'** entsprechen im Wesentlichen den korrespondierenden Komponenten des Diagnose-Frameworks **11** und sind auch (entsprechend den Beziehungen **27** und **28** in grundsätzlich gleicher Weise miteinander sowie mit der lokalen Testdatenbank **20'** – verschaltet. Insbesondere ist auch die Laufzeitumgebung **22'** in dem Simulationsmodus **34** betreibbar und wirkt hierzu mit einer korrespondierenden Simulationsdatenbank **35'** zusammen.

[0062] Das Diagnose-Framework **11** und jedes der Agent-Frameworks **12** korrespondieren über eine (jeweils master- und agent-seitig vorgesehene) Datenschnittstelle **37** bzw. **37'** miteinander. Über diese Da-

tenschnittstelle **37**, **37'** können insbesondere in dem Testeditor **21** neue Testszenarien T für ein Agent-Framework **12** erstellt und in der lokalen Testdatenbank **20'** des Agent-Frameworks **12** hinterlegt werden (Beziehung **38**). Ebenso kann ausgehend von der Laufzeitumgebung **22** die Ausführung eines Testszenarios T innerhalb der Laufzeitumgebung **22'** gesteuert werden (Beziehung **39**). Insbesondere kann hierbei ein innerhalb der Laufzeitumgebung **22** ablaufendes Testszenario T die Durchführung eines lokalen Testszenarios T in der Laufzeitumgebung **22'** auslösen. Der von der Auswerteeinheit **23'** erstellte Testbericht R wird andererseits über die Datenschnittstelle **37'**, **37** an die Auswerteeinheit **23** zurückgemeldet (Beziehung **40**) und dort – ggf. mit den Testberichten R anderer Agent-Frameworks **12** kombiniert – über die Anzeigeeinheit **24** angezeigt und/oder in dem Berichtspeicher **30** archiviert.

[0063] Jedes der Testszenarien T ist in den Testdatenbanken **20**, **20'** und **14** in einer einem Aktivitätsdiagramm gemäß UML2 entsprechenden Form hinterlegt. Ein einfaches Beispiel für eine solche Darstellung eines Testszenarios T ist in Fig. 4 abgebildet.

[0064] Das hier dargestellte Testszenario T (nachfolgend als Testszenario T1 bezeichnet) umfasst zwei Aktionen A, die zur näheren Unterscheidung nachfolgend als A1 bzw. A2 bezeichnet sind. Die Aktionen A1 und A2 sind zwischen einem so genannten Startknoten **41**, der einen Startpunkt für die Ausführung des Testszenarios T1 markiert und einem Stoppknoten **42**, der das Programmende markiert, angeordnet. Die Aktionen A und Knoten **41**,**42** sind durch gerichtete Kanten **43** verbunden, die die Bearbeitungsreihenfolge innerhalb des Testszenarios T1 bestimmen. Die nebenläufigen Programmstränge des Testszenarios T1 werden durch den Synchronisationsbalken **44** zusammengefasst.

[0065] Jede Aktion A enthält einen in sich abgeschlossenen Programmteil, bei dem es sich – wie an dem Bespiel der Aktion A1 beispielhaft angedeutet – wiederum um ein eigenes Testszenario T handeln kann. Beispielsweise enthält die Aktion A1 gemäß Fig. 4 das in Fig. 5 abgebildete Testszenario T2.

[0066] Jede Aktion A ist dazu ausgebildet, nach beendeter Ausführung zumindest einen Ausgabewert zurückzugeben, der anzeigt, ob die Aktion erfolgreich ausgeführt wurde, oder ob ein Ausführungsfehler aufgetreten ist. Beispielsweise gibt jede Aktion A für den Fall, dass sie erfolgreich ausgeführt wurde, den Wert Null zurück, während sie im Fehlerfall einen Fehlercode zurückgibt.

[0067] Der Ausgabewert wird über die von der Aktion A ausgehende Kante bzw. Kanten **43** an die oder jede nachfolgende Aktion A oder den oder jeden nachfolgenden Synchronisationsbalken **44** übertragen. Jeder Kante **43** ist eine hiervon abweichende Ausführungsregel E zuweisbar, mit der das Verhalten der dieser Kante **43** gegenüber dem Standardverhalten modifizierbar ist. Beispielsweise ist die die Aktion A1 mit der Aktion A2 verbindenden Kante **43** durch Zuweisung der Ausführungsregel E dahingehend konfiguriert, dass sie die Aktion A2 nur dann aktiviert, wenn der entlang der Kante **43** übergebene Ausgabewert eine erfolgreiche Durchführung der Aktion A1 anzeigt, beispielsweise also den Wert Null aufweist. Infolgedessen wird die Aktion A2 nur dann aktiviert, wenn zuvor die Aktion A1 erfolgreich durchlaufen wurde.

[0068] Der Synchronisationsbalken **44** wartet standardmäßig so lange, bis an jeder eingehenden Kante **43** ein Eingangswert anliegt, und gibt in diesem Fall einen entsprechenden Ausgangswert über einen oder mehrere Kanten **43** an eine oder mehrere nachfolgende Aktionen A oder Knoten, im vorliegenden Fall an den Stoppknoten **42** aus. Auch der Synchronisationsbalken **44** kann aber durch eine Ausführungsregel E modifiziert werden. Beispielsweise ist der in Fig. 4 dargestellte Synchronisationsbalken **44** durch die Ausführungsregel E dahingehend modifiziert, dass er dann schaltbereit ist, wenn entweder an jeder eingehenden Kante **43** ein Ausgangswert anliegt, der die erfolgreiche Ausführung der vorangehenden Aktion A anzeigt, oder wenn an mindestens einer eingehenden Kante **43** ein Fehlersignal anliegt.

[0069] Die Ausführung des in Fig. 4 dargestellten Testszenarios T1 beginnt somit in dem Startknoten **41**, der die Aktion A1 aktiviert. Nach Ausführung der Aktion A1 werden Ausgabewerte über die Kanten **43** an den Synchronisationsbalken **44** und – wenn die Aktion A1 erfolgreich durchgeführt wurde – an die Aktion A2 übergeben, die hierdurch ihrerseits aktiviert wird. Nach der Ausführung der Aktion A2 wird deren Ausgabewert ebenfalls über die ausgehende Kante **43** an den Synchronisationsbalken **44** geleitet, der – wenn beide Aktionen A1 und A2 erfolgreich durchgeführt wurden oder sobald von der Aktion A1 oder der Aktion A2 ein Fehlersignal übermittelt wurde – schaltet und einen entsprechenden Ausgabewert an den Stoppknoten **42** übergibt, der die Ausführung des Testszenarios T1 beendet.

[0070] Das in Fig. 5 abgebildete Testszenario T2, das – wie vorstehend erwähnt – die Aktion A1 des Testszenarios T1 bildet, umfasst seinerseits sieben Aktionen A3 bis A9, die ihrerseits jeweils wieder für ein Testszenario T stehen können. Die Ausführung des Testszenarios T2, und damit der Aktion A1, beginnt wiederum im Startknoten **41**, der die Aktionen A3 und A4 aktiviert. Der Ausgabewert dieser Aktionen A3 und A4 wird dann dem nachfolgenden Synchronisationsbalken **44** übergeben. Dessen ausgehenden Kanten **43** ist jeweils eine Ausführungsregel E zugewiesen, nach Maßgabe welcher in Abhängig-

keit des von dem Synchronisationsbalken **44** ausgegebenen Wertes eine der Aktionen A5, A6 oder A7 aktiviert wird. Beispielsweise wird die Aktion A6 aktiviert, wenn der Ausgabewert des Synchronisationsbalkens **44** größer oder gleich 0,9 ist, während die Aktion A7 aktiviert wird, wenn der Ausgabewert des Synchronisationsbalkens kleiner als 0,9 ist. Enthält der Ausgabewert des Synchronisationsbalkens **44** dagegen einen Fehlercode, so wird die Aktion A5 aktiviert. Die Aktion A6 aktiviert ihrerseits ein nachgeschaltetes Entscheidungsfeld **45**, das den Ausgabewert der Aktion A6 mit einer hinterlegten Bedingung vergleicht und nach Maßgabe des Vergleichsergebnisses ("Wahr" oder "Falsch") eine der Aktionen A8 und A9 aktiviert.

[0071] Die Anzeigeeinheit **24** ist nun derart ausgebildet, dass sie die hinterlegten Testszenarien T in graphischer Form als Aktivitätsdiagramm gemäß UML2, d.h. in einer Darstellung analog Fig. 4 oder Fig. 5 anzeigen kann. Die Anzeigeeinheit **24** wirkt dabei derart mit dem Testeditor **21** derart zusammen, dass neue Testszenarien T von einem Benutzer durch Erzeugung graphischer Aktionssymbole und durch graphische Verbindung dieser Aktionssymbole mit den Kanten **43** mittels Drag&Drop-Methoden erstellt werden können. Die Auswerteeinheit **23** wirkt ihrerseits derart mit der Anzeigeeinheit **24** zusammen, dass die sukzessive Abarbeitung eines Testszenarios graphisch dargestellt wird. Insbesondere ist vorgesehen, dass ein Testszenario T während der Ausführung in der Laufzeitumgebung **22** an dem Bildschirm **31** dargestellt wird, wobei Aktionen A, die bereits erfolgreich abgearbeitet wurden, Aktionen A, die einen Fehler hervorgerufen haben, und Aktionen A, die noch nicht ausgeführt wurden, farblich unterschieden werden (beispielsweise grün für erfolgreich ausgeführte Aktionen A, rot für fehlerhaft ausgeführte Aktionen A und gelb für noch nicht ausgeführte Aktionen A). Die Auswerteeinheit **23** ist weiterhin dazu ausgebildet, im Fehlerfall zu einer Aktion A2, die einen Fehler hervorgerufen hat, den zugehörigen Testbericht R mit einer Beschreibung des Fehlers in Form einer Textinformation über die Anzeigeeinheit **24** anzuzeigen. Zusätzlich ist jedem Testszenario T, und insbesondere aktionsaufgelöst jeder Aktion A eines Testszenarios T eine Problemlösungsinformation (Trouble-Shooting-Information) zugewiesen, die durch die Auswerteeinheit **23** im Fehlerfall stets oder auf Anfrage des Benutzers mit der Fehlerbeschreibung angezeigt wird, und die für den Benutzer Hinweise zur Fehlerbeseitigung zur Verfügung stellt.

[0072] Für Fehlfunktionen, die der Benutzer anhand des Testberichts R und gegebenenfalls der Problemlösungsinformation nicht selbst beheben kann, stellt das System **10** ein Supportkonzept mit mehreren hierarchisch gestaffelten Supporteinrichtungen **15** zur Verfügung.

[0073] In Fig. 6 ist der Ablauf eines mit dem System **10** durchgeführten Testverfahrens unter Einschaltung dieser Supporteinrichtungen **15** schematisch dargestellt. Danach wird das Verfahren gemäß Schritt **50** durch eine vorgegebene Auslösebedingung C angestoßen. Das System **10** stellt verschiedene Varianten zur Definition der Auslösebedingung C zur Verfügung, insbesondere

- eine manuelle Auslösung durch einen Benutzer,
- eine Auslösung durch Ablauf eines Zeitintervalls, z.B. zwei Wochen nach der Installation einer Softwarekomponente oder turnusmäßig in regelmäßigen Zeitabständen,
- eine Auslösung durch den Eintritt eines bestimmten Netzwerkstatus, z.B. dem Ausfall eines Geräts oder einer Datenübertragungsverbindung, ein bestimmter Auslastungsgrad eines Speichers etc. – in diesem Fall werden ein oder mehrere Testszenarien beispielsweise von einer entsprechenden Status- oder Fehlermeldung des Computernetzwerkes **1** angestoßen.

[0074] Nachdem durch Eintritt der Auslösebedingung C das Verfahren gestartet ist (Schritt **51**), werden in Schritt **52** durch das Diagnose-Framework **11**, ggf. in Zusammenwirkung mit einem oder mehreren der Agent-Frameworks **12**, ein oder mehrere Testszenarien T ausgeführt, die hierfür aus einer der Testdatenbanken **20**, **14**, **20'** in die Laufzeitumgebung **22**, **22'** geladen werden. Nach Abschluss dieses eigentlichen Diagnoseschritts wird durch die Auswerteeinheit **23** in Schritt **53** festgestellt, ob die Diagnose erfolgreich beendet wurde. Ist dies der Fall, so wird das Verfahren beendet (Schritt **54**).

[0075] Andernfalls erstellt die Auswerteeinheit **23** auf den Testbericht R in Form eines XML-Textes, und zeigt diesen Testbericht R zusammen mit der relevanten Problemlösungsinformation zunächst dem Benutzer an (Schritt **55**). Kann der Benutzer das Problem anhand dieser Information nicht selbst lösen, so wird der Testbericht auf Bestätigung durch den Benutzer an eine erste von drei gestaffelten Supporteinrichtungen **15** (vgl. Fig. 1) gesendet (Schritt **56**), die ihrerseits eine Fehlerlösung sucht (Schritt **57**). Findet die erste Supporteinrichtung **15** eine Lösung, so beseitigt sie auf dem Wege einer Fernwartung den Fehler innerhalb des Computer-Netzwerkes **1** (Schritt **58**), worauf das Diagnoseverfahren erneut gestartet wird, um zu überprüfen, ob der Fehler tatsächlich beseitigt ist. Findet die erste Supporteinrichtung **15** dagegen keine Lösung, so leitet sie den Testbericht R in Schritt **58** an eine übergeordnete, zweite Supporteinrichtung **15** weiter, die ihrerseits eine Lösung für den Fehler zu finden sucht (Schritt **59**). Findet die zweite Supporteinrichtung **15** eine Lösung, so verfährt sie gemäß Schritt **58**. Andernfalls leitet sie den Testbericht R an eine wiederum übergeordnete, dritte Supporteinrichtung **15** weiter (Schritt **60**), die in Schritt **61** eine Lösung zur Fehlerbeseitigung erarbeitet und

das Problem gemäß Schritt **58** behebt.

[0076] Die Supporteinrichtungen **15** sind dabei nach Art einer Baumstruktur gestaffelt. So ist die dritte Supporteinrichtung **15** insbesondere herstellerseitig angeordnet und korrespondiert mit mehreren direkt untergeordneten Supporteinrichtungen zweiten Ranges, von denen wiederum jeder mit mehreren Supporteinrichtungen **15** ersten Ranges korrespondiert.

[0077] In Fig. 7 ist die Zusammenwirkung mehrerer lokaler Computer-Netzwerke **1** gemäß Fig. 1 mit der globalen Testdatenbank **14** näher dargestellt. Jedes Computer-Netzwerk **1** ist dabei beispielsweise auf den Bereich eines Krankenhauses beschränkt, wobei innerhalb eines jeden Computer-Netzwerkes **1** ein Diagnose-Framework **11** – sowie ggf. ein oder mehrere Agent-Frameworks **12** – installiert sind. Jedes Computer-Netzwerk **1** bezieht hierbei Testszenarien T aus der globalen Testdatenbank **14** und liefert Testszenarien T, die im Rahmen des jeweiligen Computernetzwerkes **1** neu erstellt werden, an die globale Testdatenbank **14** zurück. Auf diese Weise ist eine effektive Verteilung von Testszenarien zum Testen der gängigen Netzwerkfunktionen und zum Auffinden bekannter Fehlerquellen sichergestellt. Die im Rahmen eines jeden Computernetzwerkes **1** erstellten Testberichte R werden dagegen netzwerk-lokal in der einem jeden Diagnose-Framework zugeordneten Berichtdatenbank **30** archiviert.

[0078] Um eine effektive Suche nach geeigneten Testszenarien T in der globalen Testdatenbank **14** zu ermöglichen, ist jedem Testszenario T eine Meta-Information zugewiesen, die den Anwendungsbereich des jeweiligen Testszenarios T nach Maßgabe mehrerer Kriterien spezifiziert. Insbesondere enthält die einem Testszenario T zugeordnete Meta-Information Angaben darüber, welcher Komponente (z.B. Server, Drucker, Netzwerk, etc.) und welchem Netzwerkknoten (z.B. Operationsmanagement-Server, Datenmanagement-Server, Arbeitsstation, etc.) des Computer-Netzwerkes **1** das Testszenario T zugeordnet ist, für welches zu testende Problem (Verbindungstest, Lizenztest, Pfadkontrolle etc.) das Testszenario ausgebildet ist und für welche Applikationsversion das Testszenario ausgebildet ist. Die globale Testdatenbank **14** stellt ein Suchformular zur Verfügung, in dem durch Spezifizierung der oben genannten Kriterien geeignete Testszenarien T ausgewählt werden können.

Patentansprüche

1. System (**10**) zum Testen der Funktion eines Computernetzwerkes (**1**), mit einer Testdatenbank (**14**, **20**), in der eine Anzahl von Testszenarien (T) ablegbar oder abgelegt sind, und einem Diagnose-Framework (**11**), umfassend
– eine Laufzeitumgebung (**22**), auf der das oder jedes Testszenario (T) ausführbar ist,
– eine Auswerteeinheit (**23**), die die Ausführung des oder jeden Testszenarios (T) überwacht und einen Testbericht (R) zur Anzeige für einen Benutzer erstellt, sowie
– einen Testeditor (**21**), mittels welchem zur Ausführung auf der Laufzeitumgebung (**22**) geeignete Testszenarien (T) erstellbar sind.

2. System (**10**) nach Anspruch 1, wobei das oder jedes Testszenario (T) in einer einem Petri-Netz oder einem Aktivitätsdiagramm gemäß des UML2-Standards entsprechenden Form hinterlegt ist.

3. System (**10**) nach Anspruch 2, mit einer Anzeigeeinheit (**24**), die dazu ausgebildet ist, das oder jedes Testszenario (T) in Form eines Petri-Netzes oder eines Aktivitätsdiagramm gemäß des UML2-Standards graphisch darzustellen.

4. System (**10**) nach Anspruch 3, wobei der Testeditor (**21**) mit der Anzeigeeinheit (**24**) dahingehend zusammenwirkt, dass das oder jedes Testszenario (T) graphisch in Form eines Petri-Netzes oder eines Aktivitätsdiagramm gemäß des UML2-Standards erstellbar ist.

5. System (**10**) nach Anspruch 3 oder 4, wobei die Auswerteeinheit (**23**) mit der Anzeigeeinheit (**24**) dahingehend zusammenwirkt, dass der Testbericht (R) anhand der graphischen Darstellung des oder jedes Testszenarios (T) darstellbar ist, wobei etwaige Testfehler stellen- bzw. aktionsaufgelöst dargestellt werden.

6. System (**10**) nach einem der Ansprüche 1 bis 5, wobei die Laufzeitumgebung (**22**) in einem Simulationsmodus (**34**) betreibbar ist, in dem Parameter des Computernetzwerkes (**1**) lediglich temporär änderbar sind.

7. System (**10**) nach Anspruch 6, wobei im Rahmen des Simulationsmodus (**34**) ein oder mehrere Netzwerknoten (**2**, **3**, **5**, **6**) des Computernetzwerkes (**1**) virtuell erzeugbar sind.

8. System (**10**) nach einem der Ansprüche 1 bis 7, wobei als Testdatenbank eine netzwerkexterne, weltweit zugängliche Testdatenbank (**14**) vorgesehen ist, aus der Testszenarien (T) in das Diagnose-Framework (**11**) herunterladbar sind.

9. System (**10**) nach einem der Ansprüche 1 bis 8, wobei ein Testszenario (T, T1) ein oder mehrere weitere Testszenerien (T, T2) als Bestandteile enthalten kann.

10. System (**10**) nach einem der Ansprüche 1 bis 9, mit einem Diagnose-Framework (**11**) als Master-Framework (**13**) sowie mit mindestens einem

Agent-Framework (**12**), umfassend
– eine Laufzeitumgebung (**22'**), auf der mindestens ein lokal für einen Netzwerkknoten relevantes Testszenario (T) ausführbar ist,
– eine Auswerteeinheit (**23'**), die dazu ausgebildet ist, die Ausführung des oder jeden lokalen Testszenarios (T) zu überwachen und einen Testbericht (R) zur Übermittlung an das Master-Framework (**13**) zu erstellen,
wobei das Master-Framework (**13**) auf einem Netzwerkknoten (**2**) des Computernetzwerkes (**1**), und jeweils ein Agent-Framework (**12**) auf mindestens einem weiteren Netzwerkknoten (**3**, **5**, **6**) des Computernetzwerkes (**1**) installiert oder installierbar ist, und wobei von dem Master-Framework (**13**) aus das oder jedes lokale Testszenario (T) im Rahmen des zugehörigen Agent-Frameworks (**12**) ausführbar ist.

11. System (**10**) nach einem der Ansprüche 1 bis 10, wobei dem oder jedem Testszenario (T) eine Problemlösungsinformation zugewiesen oder zuweisbar ist, und wobei die Auswerteinheit (**23**) dazu ausgebildet ist, im Falle eines negativen Testergebnisses die zugeordnete Problemlösungsinformation anzuzeigen oder zur Anzeige anzubieten.

12. System (**10**) nach einem der Ansprüche 1 bis 11, wobei die Auswerteinheit (**23**) dazu ausgebildet ist, im Falle eines negativen Testergebnisses automatisch oder auf Bestätigung eines Benutzers hin den Testbericht (R) an eine erste Supporteinrichtung (**15**) zu senden.

13. System (**10**) nach Anspruch 12, mit mehreren, hierarchisch gestaffelten Supporteinrichtungen (**15**), wobei die Auswerteeinheit (**23**) oder die erste bzw. eine untergeordnete Supporteinrichtung (**15**) dazu ausgebildet sind, automatisch oder auf Bestätigung eines Benutzers hin den Testbericht (R) an eine übergeordnete Supporteinrichtung (**15**) zu senden, wenn die erste bzw. untergeordnete Supporteinrichtung (**15**) keine Lösung zur Behebung des negativen Testergebnisses zur Verfügung stellen kann.

Es folgen 5 Blatt Zeichnungen

Anhängende Zeichnungen

FIG 1

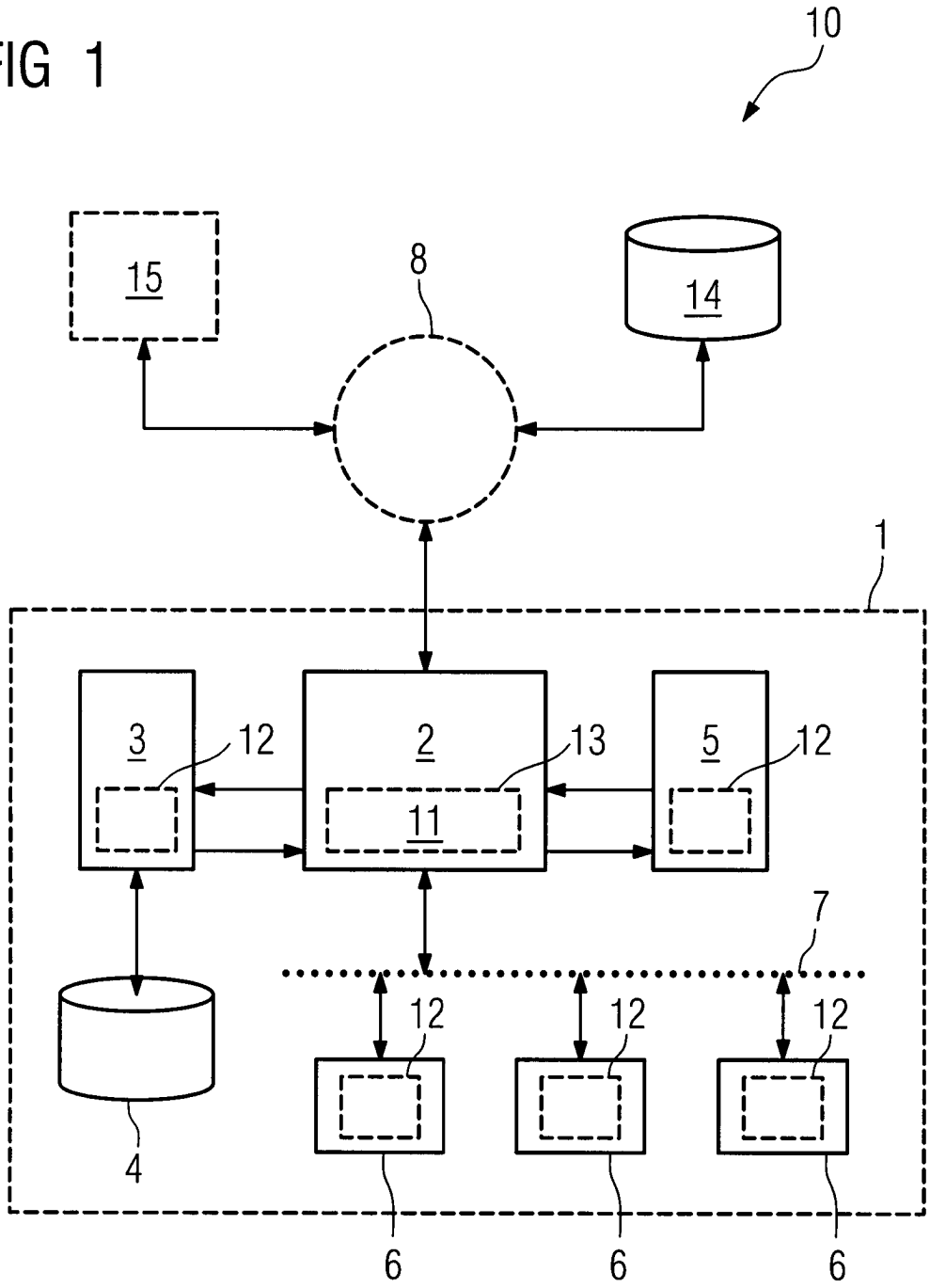


FIG 2

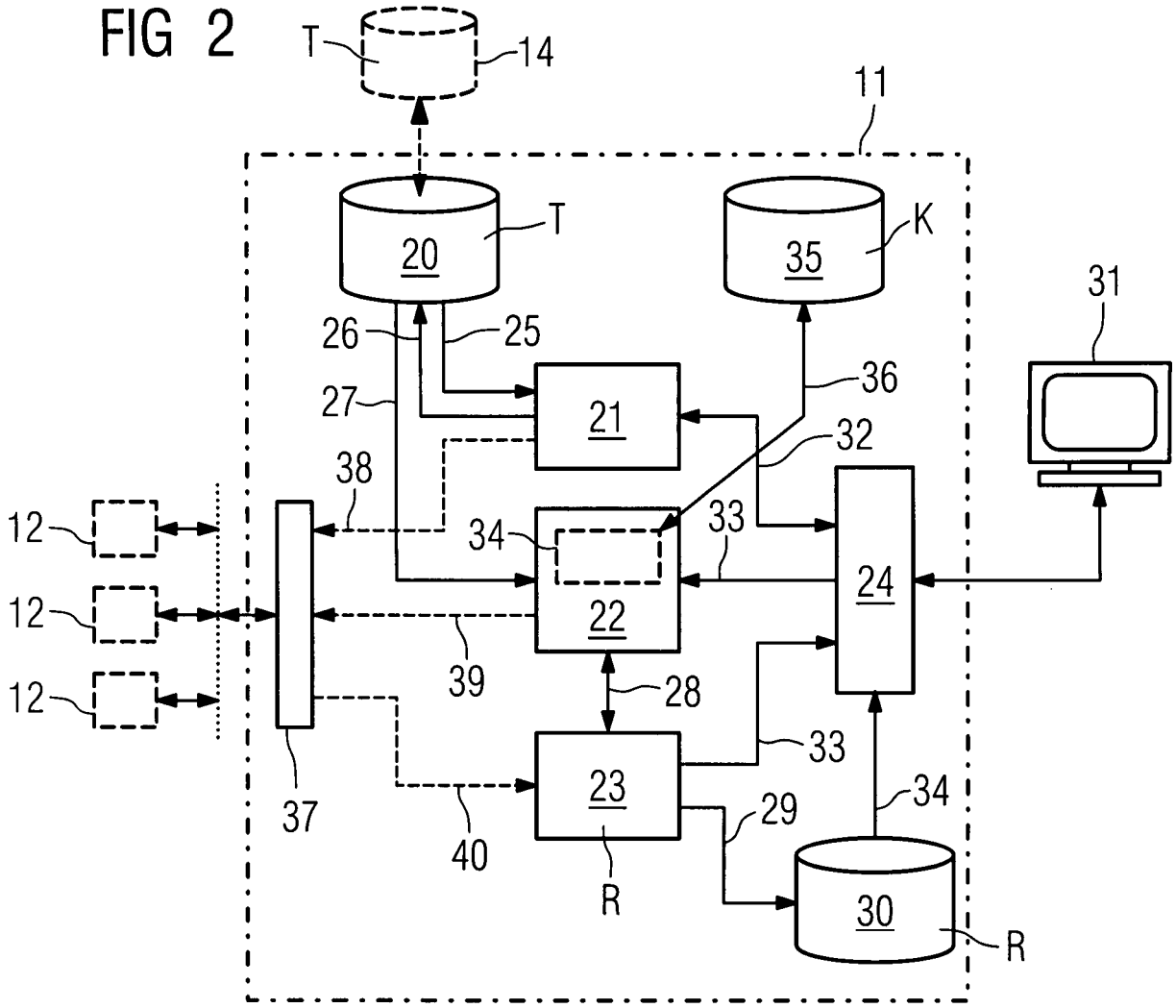

T
14
11
20
T
35
K
31
26
25
27
21
36
32
38
12
34
33
24
12
22
39
28
12
23
33
37
40
29
34
R
30
R

FIG 3

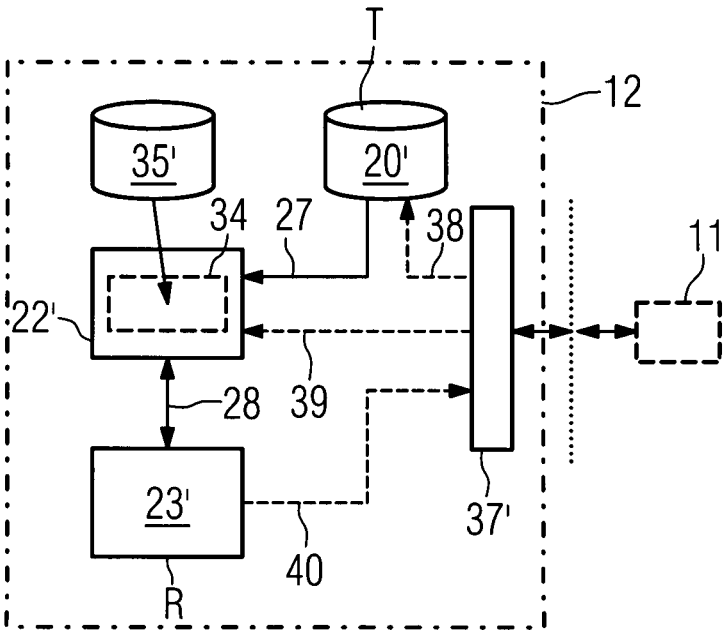

T
12
35'
20'
34
27
38
11
22'
28
39
23'
37'
40
R

FIG 4

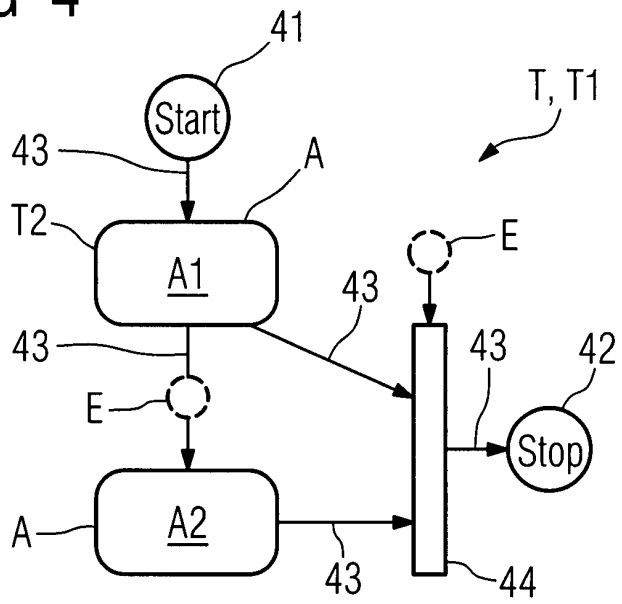

41
Start
T, T1
43
A
T2
A1
E
43
43
43
42
E
Stop
A
A2
43
44

FIG 5

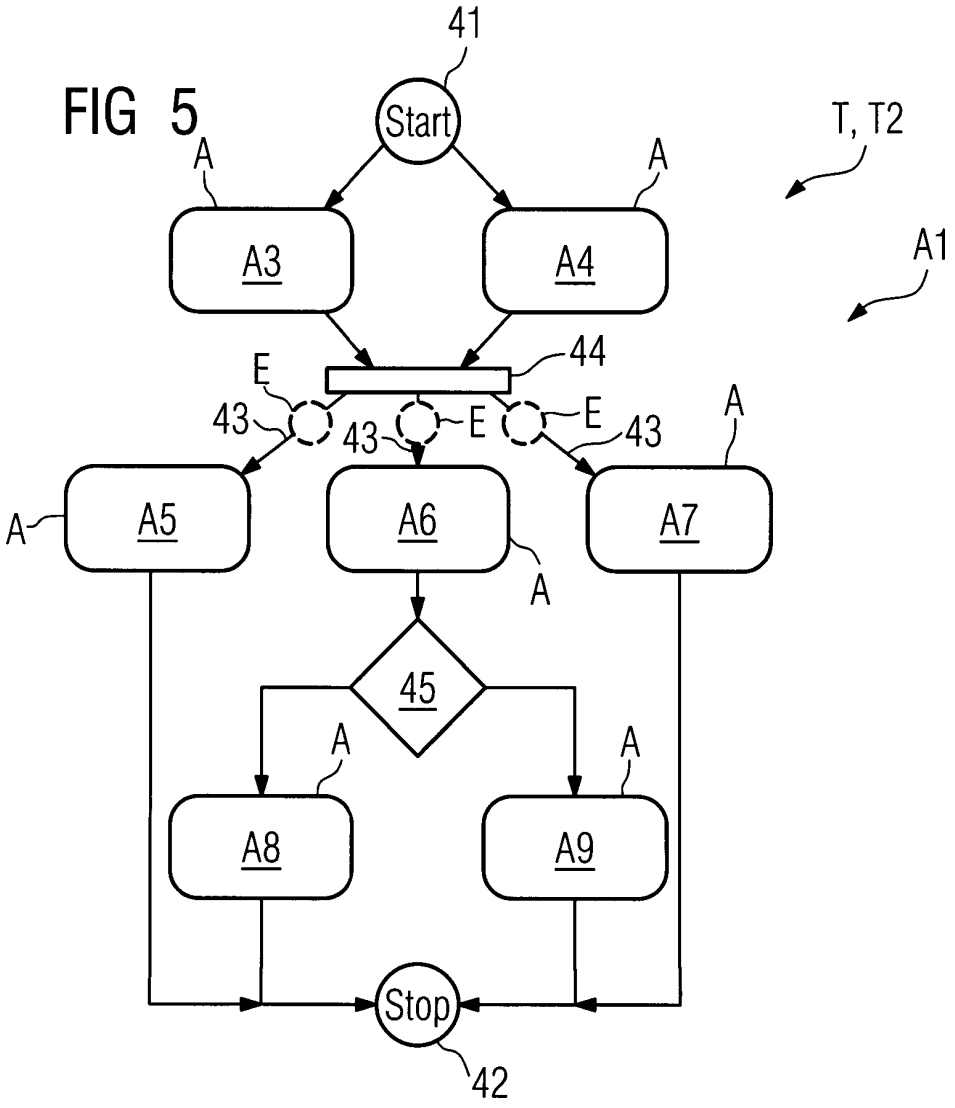

41
Start
T, T2
A
A
A1
A3
A4
44
E
43
43
E
E
43
A
A
A5
A6
A7
A
45
A
A
A8
A9
Stop
42

FIG 6

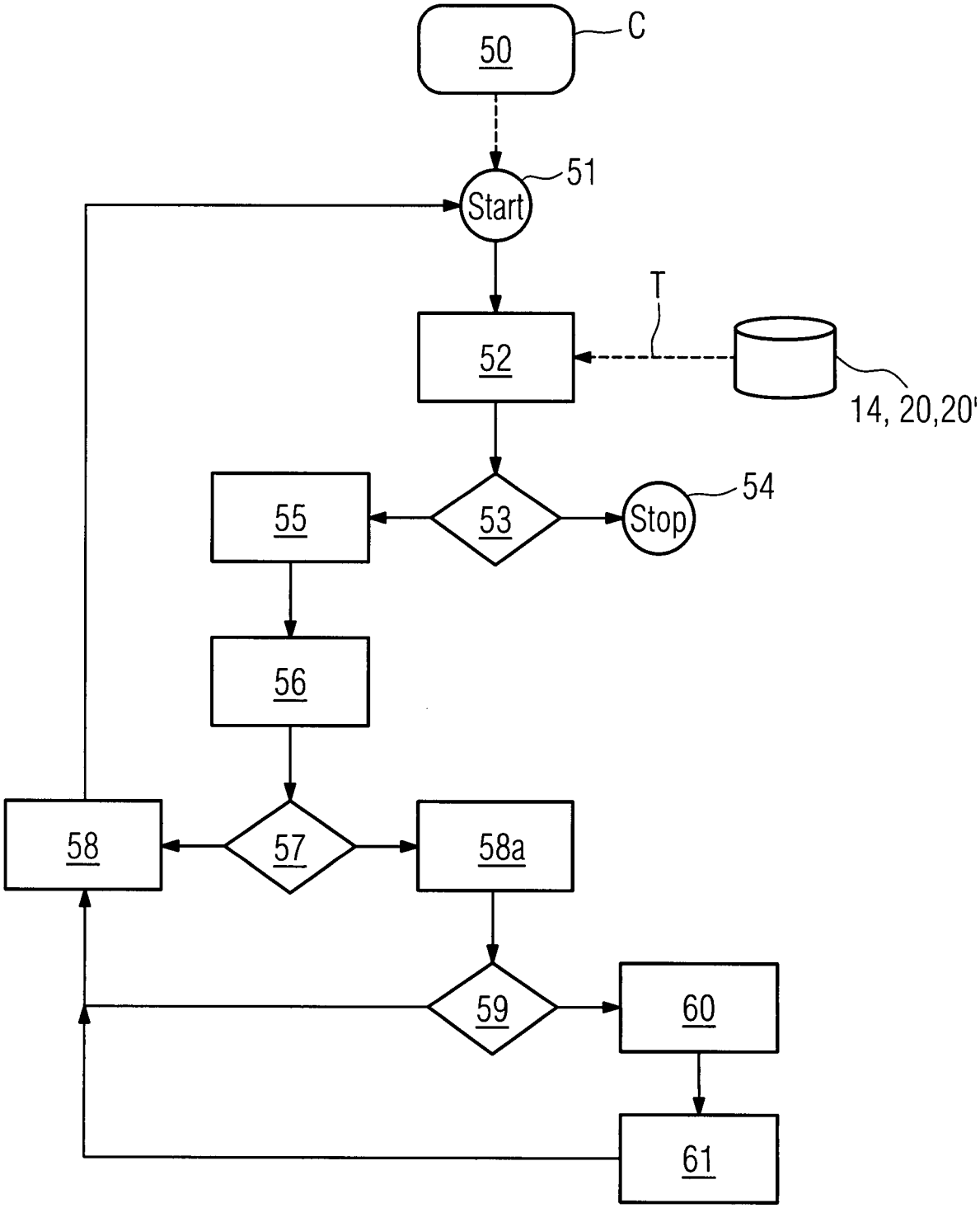


FIG 7