(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 10 2008 061 480 A1** 2010.04.08

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2008 061 480.7**
(22) Anmeldetag: **10.12.2008**
(43) Offenlegungstag: **08.04.2010**

(51) Int Cl.[8]: ***G06F 9/45*** (2006.01)
G06F 9/445 (2006.01)
G06F 9/44 (2006.01)

(66) Innere Priorität:
10 2008 050 568.4 06.10.2008

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

(72) Erfinder:
Hohenstein, Uwe, Dr., 85591 Vaterstetten, DE;
Jäger, Michael, Dr., 81541 München, DE

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:

Serban, C. et al.: AspectJTamer: The Controlled Weaving of Independently Developed Aspects. In: IEEE International Conference on Software-Science, Technology & Engineering, 2007, pp 57-65

Colyer, A. et al.: Using AspectJ for component integration in middleware. In: Conference on Object Oriented Programming Systems Languages and Applications, 2003. pp. 339-344

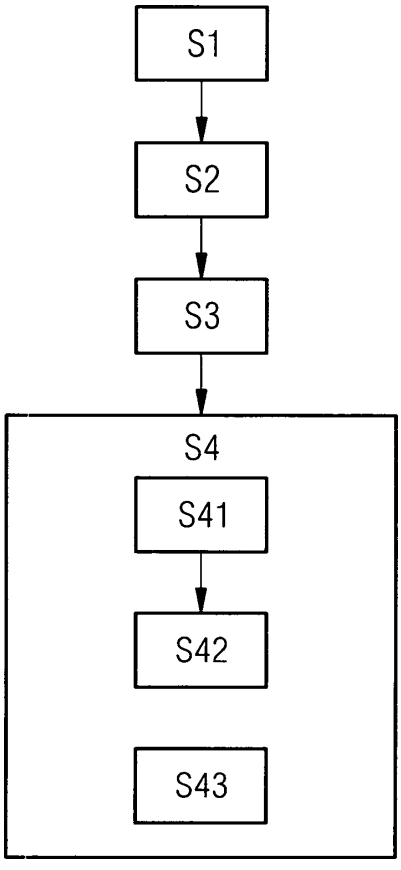
Yuan M.J. et al.: Leightweight Aspect-Oriented Programming. Dr. Dobb's Portal. 01.08.2003 <http://www.ddj.com/development-tools/184405402>

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gemäß § 44 PatG ist gestellt.

(54) Bezeichnung: **Verfahren und Vorrichtung zum Austauschen einer Komponente eines Computersystems**

(57) Zusammenfassung: Die vorliegende Erfindung betrifft das Austauschen einer Austauschkomponente (1) eines Computersystems gegen eine Ersatzkomponente (2) im Rahmen eines Migrationsprozesses in dem Computersystem, wobei die Austauschkomponente (1) in dem Computersystem durch die Ersatzkomponente (2) ersetzt wird und wobei die Ersatzkomponente (2) eine alternative Komponente zu der auszutauschenden Komponente (1) darstellt und von gleicher Art wie die auszutauschende Komponente (1) ist. Dabei weist das erfindungsgemäße Verfahren die folgenden Schritte auf: Ermitteln von Funktionalitätsunterschieden (S1) zwischen der Austauschkomponente (1) und der Ersatzkomponente (2); Implementieren von Modifikationen der Ersatzkomponente (S2) mittels Aspektprogrammierung in einem sogenannten Aspekt (4a), wobei das Implementieren auf den ermittelten Funktionalitätsunterschieden (3a) basiert; Übersetzen (S3) des Aspekts (4a) und Erzeugen einer den übersetzten Aspekt beinhaltenden Bibliothek (5a); und Anwenden (S4) des Aspekts (4a) auf die als übersetzte Bibliothek vorliegende Ersatzkomponente (2). Auf diese Weise wird ein flexibles Durchführen einer Migration in einem Computersystem erreicht.



Beschreibung

[0001] Die vorliegende Erfindung betrifft das Austauschen einer Austauschkomponente eines Computersystems gegen eine Ersatzkomponente im Rahmen eines Migrationsprozesses in dem Computersystem.

[0002] Der Begriff Migration bezeichnet den Austausch eines Bestandteils innerhalb eines Computersystems mit einem äquivalenten Ersatz bezeichnen. Der ersetzte Teil wird im Folgenden Austauschkomponente der Migration genannt. Der Teil, der an die Stelle der Austauschkomponente tritt, wird Ersatzkomponente der Migration genannt. Das Gesamtsystem, das von der Migration betroffen ist, ist das Migrationssystem.

[0003] Allgemein können sowohl Softwarekomponenten als auch Hardwarekomponenten durch die Migration durch entsprechende Softwarekomponenten und/oder Hardwarekomponenten ausgetauscht bzw. ersetzt werden. Im Nachfolgenden werden im Rahmen der Migration solche Ersatzkomponenten betrachtet, die auf Software basieren, d. h. Softwarekomponenten als solche und Hardwarekomponenten, deren Betreiben auf Software beruht. Im Nachfolgenden wird in beiden Fällen beim Durchführen der Migration die Softwareseite betrachtet und behandelt.

[0004] Dabei ist klarzustellen, dass eine Migration in erster Linie keine Neugestaltung der Architektur eines Systems oder die Erstellung eines zuvor nicht vorhandenen eigenständigen Bestandteils meint. Vielmehr sind Austausch- und Ersatzkomponente der Migration Elemente bzw. Komponenten gleicher Art. Anderenfalls müssten die Vorgänge einer Migration zusammenfassend als Neustrukturierung, Refaktorisierung, Reengineering etc. bezeichnet werden, diesen liegen aber andere Vorgänge und andere Zielsetzungen zugrunde.

[0005] Ein typischer Anwendungsfall für eine Migration ist das Ersetzen eines Bestandteils durch eine fehlerbereinigte, alternative, bessere oder neuere Version. Beispielsweise kann ein Datenbanksystem einer Mitarbeiterverwaltungssoftware durch eine aktuellere Version des Datenbanksystem ersetzt werden. In diesem Fall bezeichnet der Begriff Migration einen abgeschlossenen Vorgang, der das gegebenenfalls notwendige Sichern und Übertragen von Daten, die Konfiguration und das Testen der Migration und des Migrationssystems aufweist.

[0006] Ein anderer bekannter Fall einer Migration ist das Einspielen einer neuen Version eines Betriebssystems in ein Computersystem. Auch hier sind Austausch- und Ersatzkomponente der Migration (alte und neue Version des Betriebssystems) von gleicher Art. Dennoch ist ein gewisser Aufwand bezüglich der Planung, Konfiguration und bezüglich des Testens der Migration zu erwarten, so dass die Migration eine eigenständige und strukturierbare Aktivität darstellt.

[0007] Das Ausmaß einer Migration hängt von den Unterschieden zwischen der Austauschkomponente und der Ersatzkomponente ab. Auch wenn die Austauschkomponente und die Ersatzkomponente von gleicher Art sind, so kann die zur Verfügung gestellte Funktionalität der Austauschkomponente sich doch von der Funktionalität der Ersatzkomponente im Detail unterscheiden, so dass Anpassungen der Ersatzkomponente notwendig sind.

[0008] Das Ziel ist dabei, nach dem Durchführen der Migration dasselbe Verhalten des Migrationssystems zu gewährleisten wie vor der Migration. Anpassungen können dabei entweder bei der Ersatzkomponente vorgenommen werden, um damit die gleichen relevanten Eigenschaften des Gesamtsystems herzustellen, die beim Ausgangspunkt, d. h. vor dem Durchführen der Migration vorlagen, oder bei dem Migrationssystem vorgenommen werden, wenn die Unterschiede zwischen der Austauschkomponente und der Ersatzkomponente durch die Anpassung der Ersatzkomponente nicht kompensiert werden können. In beiden Fällen besteht die Absicht bzw. Intention, die Kompatibilität zwischen der Ersatzkomponente und dem Migrationssystem bis zu dem Grad herzustellen, der vor der Migration zwischen der Austauschkomponente und dem Migrationssystem bestand.

[0009] Heutzutage erlangen Migrationsszenarien und das Anwenden der Migration allgemein eine erhöhte Aufmerksamkeit. Dieses geschieht insbesondere aufgrund eines zunehmenden Einsatzes von Fremdsoftware wie zum Beispiel Datenbanken, objektrelationalen (O/R) Frameworks, Java Platform, Enterprise Edition (JEE)-Applikationsservern, Open Services Gateway initiative (OSGi)-Containern.

[0010] Bei Problemen, mangelnder Funktionalität oder schlechtem Preis-Leistungsverhältnis stellt eine Migration in der Regel einen Ausweg zum Bewältigen dieser Probleme dar. Die heutige Informationstechnik-(IT)-Landschaft ist zudem dadurch geprägt, dass weniger Neuentwicklungen als vielmehr Weiterentwicklungen bestehender Softwarelösungen erfolgen und ein großer Markt an Beratungsleistungen für solche Mig-

rationen existiert.

[0011] Im Hinblick auf das Durchführen einer Migration mit bekannten Verfahren oder Vorgehensweisen lassen sich je nach Ausgangslage die folgenden grundsätzlichen Fälle unterscheiden:

1. Liegt der Quellcode bzw. der Softwarecode der Ersatzkomponente nicht vor, besteht keine technische Möglichkeit, die interne Funktionalität der Ersatzkomponente zu erweitern.
2. Falls der Quellcode bzw. der Softwarecode der Ersatzkomponente und die Übersetzungsanweisungen ("build scripts") für den Quellcode bzw. den Softwarecode vorliegen, kann der Quellcode modifiziert und neu übersetzt werden. Die Modifikation und die erneute Übersetzung lassen sich relativ einfach durchführen.
3. Liegt der Quellcode bzw. der Softwarecode der Ersatzkomponente vor und fehlt es aber an Anweisungen zur Übersetzung des Quellcodes bzw. des Softwarecodes, so muss der Übersetzungsvorgang für den Quellcode bzw. den Softwarecode der Ersatzkomponente neu konfiguriert werden, wobei dies mit erhöhtem zusätzlichen Aufwand verbunden ist.

[0012] Neben den technischen Voraussetzungen sind auch weitere Aspekte zu berücksichtigen:

a) Die Modifikation und eine neue Übersetzung einer Ersatzkomponente werden durch in der Regel gegebene Lizenzbedingungen nur mit zusätzlichen Auflagen erlaubt. Ein weit verbreiteter Fall für derartige Auflagen ist die Offenlegung der Modifikationen. Dies kann in manchen Fällen nicht gewollt sein, da damit eine Offenlegung des Know-hows einhergeht. Weitere Auflagen können zum Beispiel Auswirkungen auf die Lizenzbestimmungen des gesamten Migrationssystems sein, welches unter Umständen das Geschäftsmodell verändern kann.

b) Die Modifikation und neue Übersetzung ist zwar technisch möglich, würde aber einen Verlust der Lizenzierung bedeuten, da Änderungen vom Lizenzgeber untersagt sind.

c) Die Modifikation und neue Übersetzung der Ersatzkomponente ist ohne Einschränkungen zu sehen.

[0013] Diese Auflistung macht deutlich, dass nur die Kombination der obigen Punkte 2 und c) und teilweise auch die Kombination der obigen Punkte 2 und a) aus der technisch möglichen Übersetzbarkeit und den mit der technisch möglichen Übersetzbarkeit korrespondierenden Rahmenbedingungen als lösbare und vertretbare Situationen für die Migration gelten können.

[0014] Für die anderen Fälle existiert bisher kein allgemeingültiges und systematisch einsetzbares Verfahren. Je nach Einzelfall existieren gegebenenfalls Ansätze, die eine erfolgreiche Durchführung der Migration nicht garantieren. Beispielsweise kann die Ersatzkomponente gekapselt werden, um zusätzliche Funktionalität bereitzustellen. Das funktioniert allerdings nur für extern verfügbare Schnittstellen. Für das Behandeln von Unterschieden, welche die internen Abläufe betreffen, kann auf diese Weise keine geeignete Lösung bereitgestellt werden.

[0015] Es wird deutlich, dass abgesehen von den eindeutig lösbaren Fällen, Kombinationen aus den anderen oben genannten Punkten aufgrund fehlender systematisch angelegter Lösungswege eine hohe Wahrscheinlichkeit für das Scheitern der Migration vorliegt. Dies resultiert in nicht unerheblichen Nachteilen. So wird zum Beispiel die fehlende Funktionalität der Ersatzkomponente trotz gründlicher Analyse und Planung in der Regel erst spät erkannt. Ein Misserfolg und Abbruch der Migration führt sodann zwangsläufig zu einer vorliegenden Fehlinvestition und zu einer Notwendigkeit einer erneuten Investition mit einer alternativen Ersatzkomponente.

[0016] Eine Migration ist aber trotz der der oben ausgeführten Probleme das Mittel der Wahl, wenn es um Verbesserung der Leistung, Steigerung der Effizienz, Weiterentwicklung und/oder Anpassung von rechnenden Systemen, Komponenten, Vorrichtungen, Modulen und/oder Anwendungen geht.

[0017] Eine Aufgabe der vorliegenden Erfindung besteht somit darin, ein verbessertes Verfahren zur Durchführung einer Migration einer Komponente eines rechnenden Systems, insbesondere eines Computersystems, bereitzustellen.

[0018] Diese Aufgabe wird gelöst durch ein Verfahren zum Austauschen einer Austauschkomponente eines Computersystems gegen eine Ersatzkomponente mit den Merkmalen des Anspruchs 1.

[0019] Die Unteransprüche geben weitere Ausgestaltungen der vorliegenden Erfindung an.

[0020] Wie bereits erwähnt, wird die oben genannte Aufgabe mittels eines Verfahrens zum Austauschen einer Austauschkomponente eines Computersystems gegen eine Ersatzkomponente gelöst, wobei die Austauschkomponente in dem Computersystem durch die Ersatzkomponente ersetzt wird, wobei die Ersatzkomponente

eine alternative Komponente zu der auszutauschenden Komponente darstellt und von gleicher Art wie die auszutauschende Komponente ist, und wobei das Verfahren aufweist:

– Implementieren von Modifikationen der Ersatzkomponente mittels Aspektprogrammierung in einem Aspekt basierend auf ermittelten Funktionalitätsunterschieden zwischen der Austauschkomponente und der Ersatzkomponente;
– Übersetzen des Aspekts und Erzeugen einer den übersetzten Aspekt beinhaltenden Bibliothek; und
– Anwenden des Aspekts auf die als übersetzte Bibliothek vorliegende Ersatzkomponente.

[0021] Sowohl die Austauschkomponente als auch die Ersatzkomponente stellt jeweils eine Softwarekomponente dar, die zum Einsetzen und Verwenden in einem Computersystem vorgesehen und konfiguriert ist. Dabei übernehmen die Austauschkomponente und die Ersatzkomponente verschiedene Funktionen des Computersystems. So können zum Beispiel die Austauschkomponente und die Ersatzkomponente das Betreiben, Steuern und/oder Verwalten von anderen Komponenten des Computersystems übernehmen. Die anderen Komponenten können dabei Soft- und/oder Hardwarekomponenten sein.

[0022] Ferner ist anzumerken, dass gemäß der vorliegenden Erfindung unter dem Begriff ”Bibliothek” eine Programm- oder Softwarebibliothek verstanden wird, die in der Softwareprogrammierung eine Sammlung von Programm- oder Softwarefunktionen für zusammengehörende Aufgaben wie vorstehend aufgezeigt bezeichnet.

[0023] Durch die vorliegende Erfindung wird ein effizienter, effektiver und flexibel gestaltbarer Weg der Umsetzung einer Migration ermöglicht. Insbesondere ist dabei kein explizites Anpassen des Quellcodes der Ersatzkomponente notwendig. D. h. die vorliegende Erfindung kann umgesetzt werden selbst wenn kein Quellcode vorliegt. Ferner kann durch die vorliegende Erfindung den oben erläuterten Rahmenproblemen aus dem Weg gegangen werden. Zusätzlich wird durch das Verwenden des erfindungsgemäßen Verfahrens vermieden, dass Fehler in dem ursprünglichen Code der Ersatzkomponente einschleichen und dass ein aufwändiges Testen und Validieren der Ersatzkomponente und des Migrationssystems notwendig wird. Durch das Ausführen der Migration gemäß der vorliegenden Erfindung ist die Wartbarkeit der durchgeführten Modifikationen übersichtlich und auch deutlich einfacher, effektiver und effizienter handhabbar.

[0024] Gemäß einem Ausführungsbeispiel der vorliegenden Erfindung wird der Aspekt von der Ersatzkomponente separat implementiert. Auf diese Weise wird vermieden, dass die oben genannten Probleme entstehen. Ferner werden die Modifikationen deutlich von der Ersatzkomponente getrennt, was zu einer besseren Handhabung und zu einem verbesserten Testen des Erfolgs der Migration führt.

[0025] Gemäß einem Ausführungsbeispiel der vorliegenden Erfindung weist das Anwenden des Aspekts auf die als übersetzte Bibliothek vorliegende Ersatzkomponente ein Erstellen einer neuen und erweiterten Bibliothek der Ersatzkomponente auf. Auf diese Weise bleibt das Kompilat der Ersatzkomponente unverändert, was insbesondere den Vorteil mit sich bringt, dass die oben genannten Probleme umgangen werden können.

[0026] Gemäß einem weiteren Ausführungsbeispiel der vorliegenden Erfindung weist das Verfahren ein Festlegen der Position des Aspekts in der Ersatzkomponente auf. Hiermit wird ein gezieltes Durchführen der Modifikation oder Anpassungen ermöglicht, wobei es zugleich keines Eingreifens in die Ersatzkomponente bzw. in ihr Code oder in ihre Bibliothek bedarf.

[0027] Dabei wird gemäß einem Ausführungsbeispiel der vorliegenden Erfindung das Anwenden des Aspekts auf die als übersetzte Bibliothek vorliegende Ersatzkomponente unter Verwendung des Aspekts an der festgelegten Position in der Ersatzkomponente durchgeführt.

[0028] Gemäß einem Ausführungsbeispiel der vorliegenden Erfindung wird durch die Position festgelegt, welcher Code bzw. welcher Teil des Codes der Ersatzkomponente modifiziert werden soll. Somit wird ein sehr genaues und ein sehr gezieltes Modifizieren oder Anpassen der Ersatzkomponente bzw. ihres Softwarecodes, bzw. ihrer Bibliothek erlaubt, das auf der Code-Ebene durchgeführt wird und das zugleich die Ersatzkomponente unverändert lässt.

[0029] Die hier geschilderte Erfindung nutzt den Einsatz der aspektorientierten Programmierung (AOP), um Defizite der Funktionalität einer Ersatzkomponente auszugleichen und um notwendige Kompatibilität zwischen Migrationsystem und Ersatzkomponente herzustellen. Hierzu werden die Funktionalitätserweiterungen bzw. -Modifikationen von der Ersatzkomponente separat in einem sogenannten Aspekt definiert. Bestandteil dieser Definition ist auch das Festlegen der Position in der Ersatzkomponente, an der der Aspekt angewendet werden

soll. Das grundsätzliche Schema ist hierbei, dass die Ersatzkomponente als bereits übersetzte Bibliothek (zum Beispiel im Umfeld der Programmiersprache Java als Java-Archiv ".jar") vorliegt, unverändert bleibt und mit separat vorliegenden Aspektdefinitionen modifiziert wird.

[0030] Das Verfahren zur Anwendung eines Aspekts und die Syntax der Aspektdefinition hängen von der verwendeten Implementierung ab. Zurzeit existieren für nahezu alle gängigen Programmiersprachen Umsetzungen für die Anwendung von AOP.

[0031] In der Java-Programmierung zum Beispiel kann ein Aspekt als normale Java-Klasse implementiert werden. Die Klasse beschreibt die erweiterte oder geänderte Funktionalität. Annotationen geben an, welche Codeteile der Ersatzkomponente ausgetauscht oder erweitert werden sollen. Ein spezielles Verfahren legt fest, wie eine erweiterte Ersatzkomponente erstellt werden kann.

[0032] Die vorliegende Erfindung ist dabei nicht nur auf Java-Programmierung beschränkt. Es können die Aspekte gemäß der vorliegenden Erfindung in verschiedenen Programmiersprachen implementiert werden. Ferner ist die vorliegende Erfindung nicht nur auf das Umsetzen durch eine Java-Klasse beschränkt, sondern es können verschiedene geeignete Programmierparadigmen verwendet werden.

[0033] Durch die vorliegende Erfindung wird ein Verfahren zur Behebung von Funktionalitätsunterscheiden bei der Migration unter Einsatz von AOP geschaffen. AOP wird bisher nur zur strukturierten Neuentwicklung von Anwendungen eingesetzt.

[0034] Der erfindungsgemäße Einsatz von AOP liegt in der Anwendung auf bereits existierende, von Drittherstellern gelieferte Komponenten. Dadurch wird das klassische Rollenmodell – Entwickler der Komponente definiert auch zugehörige Aspekte – aufgehoben. Die Erstellung der Komponente und die Definition der dazugehörigen Aspekte werden gemäß der vorliegenden Erfindung von verschiedenen Parteien vorgenommen.

[0035] Wie vorstehend angedeutet, weist die vorliegende Erfindung insbesondere den Vorteil auf, dass keine explizite Quellcodeanpassung notwendig ist. Gemäß der vorliegenden Erfindung wird eine externe Beschreibung der Modifikationen als ein Aspekt erstellt, der von der Ersatzkomponente separat vorliegt. Daraus ergibt sich der Vorteil, dass die Verfügbarkeit des Quellcodes der Ersatzkomponente nicht notwendig ist.

[0036] Ferner ist gemäß der vorliegenden Erfindung keine Kenntnis bzw. kein Wissen zur Konfiguration des Übersetzungsvorgangs (z. B. des „build scripts" (englisch) bzw. des Erstellungsscripts) notwendig, da die neue Übersetzung der Ersatzkomponente entfällt.

[0037] Des Weiteren bleibt der bestehende und original erstellte Quellcode bzw. der übersetzte Quellcode oder das Kompilat der Ersatzkomponente unverändert.

[0038] Im Gegensatz zu einer Kapselung der Komponente die, eine gängige Vorgehensweise des Standes der Technik darstellt, können durch die vorliegende Erfindung Änderungen der internen Funktionalität vorgenommen werden. Die bekannten Verfahren, die eine Kapselung der Komponente umsetzen, verwenden z. B. sogenannte Wrapper, die zum Übersetzen einer Schnittstelle in eine andere ausgestaltet sind, um eine Kommunikation von Klassen mit zueinander inkompatiblen Schnittstellen zu ermöglichen. Das Ändern der internen Funktionalität ist in diesen bekannten Verfahren nicht möglich, ist aber insbesondere dann von Bedeutung, wenn interne Verarbeitungsinformationen extrahiert werden müssen, die von der Ersatzkomponente nicht über die Schnittstellen angeboten werden. Die vorliegende Erfindung bietet somit ein deutlich mächtigeres Verfahren, das weit mehr Möglichkeiten der Funktionserweiterung in einer Ersatzkomponente gestattet.

[0039] Ferner erlaubt das Formulieren einer Funktionserweiterung in einem externen Aspekt ein Trennen der Modifikationen von der originalen Ausgestaltung der Ersatzkomponente. Dadurch werden Nachvollziehbarkeit und Wartbarkeit der Modifikation erhöht. Würde der Quellcode der Ersatzkomponente modifiziert werden, müsste entweder ein Verfahren zur Markierung der Modifikation oder ein Verfahren zum Vergleich mit dem Original bereitgestellt werden, um die Nachvollziehbarkeit zu gewährleisten.

[0040] Die vorliegende Erfindung bietet somit ein Entwicklungsrahmen, mit dem Funktionalitätserweiterungen in den als bestehende Bibliotheken vorliegenden Ersatzkomponenten eingebracht werden können.

[0041] Das erfindungsgemäße Verfahren erlaubt ein sicheres, effektives und umfassendes Durchführen einer Migration bzw. Austauchen einer Austauschkomponente eines Computersystems gegen eine Ersatzkompo-

nente. Dies führt zu einer sicheren und effektiven Verbesserung der Leistung, einer Effizienzsteigerung, und einer Anpassung des Computersystems, in dem die Ersatzkomponente eingesetzt wird.

[0042] Im Folgenden werden Ausführungsformen der vorliegenden Erfindung detailliert unter Bezug auf die unter beigefügten Figuren beschrieben.

[0043] Es zeigen:

[0044] Fig. 1 ein Ablaufdiagramm, das die Schritte des Verfahrens zum Austauschen einer Austauschkomponente eines Computersystems gegen eine Ersatzkomponente gemäß einem Ausführungsbeispiel der vorliegenden Erfindung darstellt; und

[0045] Fig. 2 eine schematische Darstellung einer Anordnung zum Austauschen einer Komponente eines Computersystems gemäß einem Ausführungsbeispiel der vorliegenden Erfindung.

[0046] Wie oben beschrieben, bleibt es in der Regel bei Computersystemen als Migrationssystemen nicht nur bei Adaptionen. Vielmehr ist auch die Integration der Adaption vorzunehmen. Ferner müssen auch Tests zur Verifikation der Kompatibilität zwischen dem Migrationssystem und der Ersatzkomponente durchgeführt werden. Insgesamt lässt sich eine Migration in folgende Phasen einteilen:

1. Planung, wobei die Planung die Auswahl einer Ersatzkomponente und die Ermittlung der Unterschiede zwischen Austausch- und Ersatzkomponente aufweist.
2. Fehlende Funktionalität überbrücken; wobei dies die Implementierung der notwendigen Anpassungen bei der Ersatzkomponente oder beim Migrationssystem mit umfasst.
3. Integration der Ersatzkomponente; dabei sind in der Regel andere Klassen bzw. deren Schnittstellen mit anderen Methoden zu benutzen, um die Ersatzkomponente anzubinden bzw. in das Migrationssystem einzubinden. Bei gravierenden syntaktischen Unterschieden kann das mit erhöhtem Aufwand verbunden sein.
4. Test des Migrationsergebnisses.

[0047] Die vorliegende Erfindung bezieht sich dabei insbesondere auf die Phase 2 der Migration, d. h. das Überbrücken der fehlenden Funktionalität der Ersatzkomponente und die Implementierung der notwendigen Anpassungen an der Ersatzkomponente. Der Ausgangsfall ist, dass zum Erreichen der ursprünglichen Kompatibilität Modifikationen an der Ersatzkomponente vorgenommen werden müssen. Dies ist dann notwendig, wenn die Ersatzkomponente eine Funktionalität nicht bereitstellt, die bei der Austauschkomponente vorhanden war. An dieser Stelle liegt auch der kritische Punkt der Migration. Stellt sich heraus, dass die notwendige Kompatibilität nicht in herkömmlicher Weise hergestellt werden kann, scheitert das gesamte Migrationsvorhaben. Dabei kommt erschwerend hinzu, dass derartige Defizite der Ersatzkomponente erst sehr spät erkannt werden.

[0048] Die hier vorgestellte Erfindung ermöglicht eine Modifikation der Ersatzkomponente, um die Kompatibilität mit dem Migrationssystem zu ermöglichen. Der Ansatz bietet zudem erstmalig eine Lösung für den Fall, dass eine Modifizierung am Quellcode der Ersatzkomponente aufgrund rechtlicher Rahmenbedingungen nicht möglich ist.

[0049] Fig. 1 zeigt ein Ablaufdiagramm, welches das Austauschen einer Austauschkomponente eines Computersystems gegen eine Ersatzkomponente gemäß einem Ausführungsbeispiel der vorliegenden Erfindung darstellt.

[0050] In einem Schritt S1 werden Unterschiede, insbesondere Funktionalitätsunterschiede zwischen der Austauschkomponente und der Ersatzkomponente ermittelt. Die ermittelten Funktionalitätsunterschiede werden verwendet, um im Schritt S2 Modifikationen bzw. Anpassungen der Ersatzkomponente mittels Aspektprogrammierung in einem sogenannten Aspekt zu implementieren.

[0051] Anschließend wird der implementierte Aspekt im Schritt S3 übersetzt, wobei auch eine den übersetzten Aspekt beinhaltende Bibliothek erzeugt wird. Im Schritt S4 wird dann der übersetzte Aspekt, insbesondere die den übersetzten Aspekt beinhaltende Bibliothek auf die Ersatzkomponente angewendet. Dabei liegt die Ersatzkomponente als eine übersetzte Bibliothek vor.

[0052] Das Anwenden des Aspekts auf die Ersatzkomponente im Schritt S4 kann ein Festlegen der Position des Aspekts in der Ersatzkomponente aufweisen. Dieses wird gemäß dem vorliegenden Ausführungsbeispiel im Schritt S41 durchgeführt. Wird eine solche Position des Aspekts bestimmt, so weist das Anwenden S4 des

Aspekts auf die Ersatzkomponente ein Verwenden des Aspekts an der festgelegten Position auf. Dieses wird im Schritt S42 durchgeführt. Durch die Position wird festgelegt, welcher Code der Ersatzkomponente modifiziert bzw. angepasst werden soll.

[0053] Ferner wird beim Anwenden S4 des Aspekts auf die Ersatzkomponente durchgeführt und eine neue und um den Aspekt erweiterte Bibliothek der Ersatzkomponente erstellt. Das Erstellen der erweiterten Bibliothek wird in der Fig. 1 durch den Schritt S43 repräsentiert.

[0054] Fig. 2 zeigt eine schematische Darstellung einer Anordnung zum Austauschen einer Komponente eines Computersystems gemäß einem Ausführungsbeispiel der vorliegenden Erfindung.

[0055] Es werden eine Austauschkomponente **1** und eine Ersatzkomponente **2** bereitgestellt. Hierbei soll die Austauschkomponente **1** durch die Ersatzkomponente **2** ausgetauscht werden. Eine Funktionsanalysevorrichtung **3** analysiert jeweils die Funktionalität, welche durch die Austauschkomponente **1** beziehungsweise durch die Ersatzkomponente **2** bereitgestellt wird. Der ermittelte Funktionalitätsunterschied **3a** wird an eine Anpassungsvorrichtung **4** übermittelt. In der Anpassungsvorrichtung **4** wird eine Anpassung der Funktionalität der Ersatzkomponente **2** an die Funktionalität der Austauschkomponente **1** als ein Aspekt **4a** implementiert. Der implementierte Aspekt **4a** wird an eine Übersetzungsvorrichtung **5** übermittelt. Die Übersetzungsvorrichtung **5** wird eine Aspektbibliothek **5a**, welche den Aspekt **4a** aufweist. Die Aspektbibliothek **5a** wird an eine Anwendungsvorrichtung **6** übermittelt. Die Anwendungsvorrichtung **6** wendet die Aspektbibliothek **5a** auf die Ersatzkomponente **2** an, wobei die Ersatzkomponente **2** als Bibliothek vorliegt.

[0056] Somit wurde die Funktionalität der Ersatzkomponente **2** an die Funktionalität der Austauschkomponente **1** angepasst. Folglich kann die Austauschkomponente **1** durch die Ersatzkomponente **2** ausgetauscht werden.

[0057] Im Folgenden wird die vorliegende Erfindung beispielhaft anhand eines Projekts erläutert, in dem das erfindungsgemäße Verfahren erfolgreich durchgeführt wird.

[0058] Dabei wird in einem Softwaresystem das Objekt-relationale Framework Hibernate eingesetzt, um den Zugang zu einer relationalen Datenbank zu vereinfachen. Aufgrund einer besonderen Rahmenbedingung ist in dem dargestellten Beispiel der Framework Hibernate aus dem Softwaresystem herauszulösen und durch ein alternatives Produkt zu ersetzen. Die Migration besteht darin, ”Hibernate” durch ein anderes Objekt-relationales Framework namens ”OpenJPA” zu ersetzen. Die Austauschkomponente dieser Migration ist somit Hibernate, die Ersatzkomponente OpenJPA.

[0059] Oberflächlich betrachtet handelt es sich um ein klares Migrationsvorhaben, da Austauschkomponente und Ersatzkomponente über eine vergleichbare Funktionalität verfügen. Im Prinzip sind im Migrationssystem nur unterschiedliche Schnittstellen zu benutzen, wobei dies durch eine Kapselung, wie oben erläutert, handhabbar ist.

[0060] Bei einer tiefgehenden Untersuchung treten jedoch die folgenden schwerwiegenden Probleme auf:

1. Zum Zwecke der Datenbankausfallsicherheit wird eine spezielle Failover-Datenbank-URL jdbc:solid://<host1>:<port1>,<host2>:<port2>/<usr>/<password> benutzt, die einen Primär- und einen Sekundärrechner host1 bzw. host2 festlegt. Fällt das Datenbanksystem auf dem Primärrechner aus, z. B. weil der Rechner heruntergefahren ist, so wird automatisch auf den bereitstehenden Sekundärrechner umgeschaltet, der sofort den Datenbankbetrieb übernimmt. Diese spezielle Datenbankausfallsicherheit funktionierte unter ”Hibernate”, allerdings nicht unter ”OpenJPA”. Der Grund dafür ist, dass die Failover-URL-Form von OpenJPA nicht durchgelassen wird. Die korrekte URL jdbc:solid://<host1>:<port1>,<host2>:<port2>/<usr>/<password> wird zu jdbc:solid://<host1>:<port1> abgeschnitten. Dieses falsche Behandeln der URL durch OpenJPA führt dazu, dass überhaupt kein Verbindungsaufbau zur SOLID-Datenbank mehr möglich ist. Bei Ausfall von <host1> erfolgt insbesondere kein automatischer Wechsel auf <host2> mehr, was jedoch zwingend erforderlich ist.
2. Zusätzlich zur Datenbank-URL ist eine spezielle Option solid_tf_level auf l1 zu setzen, um das Failover zu ermöglichen. OpenJPA bietet allerdings keine Möglichkeit, eine entsprechende Property (englisch) bzw. Eigenschaft zu setzen und an das Datenbanksystem weiterzureichen.

[0061] Diese Probleme sind kritisch für den Gesamterfolg der Migration. Werden diese nicht gelöst, so ist eine Migration von Hibernate auf OpenJPA, bzw. ein Ersetzen des objektrelationalen Frameworks ”Hibernate” durch ”OpenJPA” zum Scheitern verurteilt. Der bis dahin getätigte Aufwand im Projekt ist dann ebenfalls verloren.

Eine Wahl eines anderen, neuen Kandidaten als Ersatzkomponente und eine Migration mit diesem neuen Kandidaten führt zu weiteren Aufwand von mehreren Personenmonaten – ohne vorhersehbare Aussicht auf Erfolg.

[0062] Die oben diskutierten herkömmlichen Lösungsansätze, wie Kapselung oder Quellcode-Änderungen, bieten keine adäquaten Lösungen, da durch sie zum Beispiel der Quellcode nicht verfügbar und somit nicht änderbar ist.

[0063] Eine erfolgreiche Migration kann jedoch mittels der vorliegenden Erfindung, wie in diesem Ausführungsbeispiel erläutert, erreicht werden.

[0064] Die oben genannten Probleme 1 und 2 werden gemäß dem vorliegenden Ausführungsbeispiel unter Verwendung des folgenden schematisch dargestellten Aspekts gelöst:

```
@Aspect
public class ChangeURLAspect {
@Around("ececution("public static * "
+ " org..Configurations.parseProperties(String)) "
+ "&& args(str) && within(org.apache.openjpa.lib.conf..*)")
// in Configurations.parseProperties steckt der Fehler: URL
// wird auseinandergerissen: Folgende Austauschmethode
// korrigiert das:
public Object parseProperties(final String str, final Join-
Point jp) {
  Options opts = new Options();
  String properties = StringUtils.trimToNull(str);
  if (properties == null) {
     return opts;
  }
  try {
    String [] props = Strings.split(properties, ",", 0);
    int idx;
    char quote;
    String prop;
    String val;
    for (int i = 0; i < props.length; i++) {
      idx = props[i].indexOf('=');
      // Das Originalverhalten: Enthält ein Teil kein
      // "=", wird er ignoriert
      if (idx == -1) {
        prop = props[i];
        val = prop;
        // Der nachfolgende Teil korrigiert das
        // Originalverhalten
```

```
        int colon = prop.indexOf(":");
        int slash = prop.indexOf("/");
        if (colon > 0 && slash > 0) {
          // Failover-URL erkannt
          final String host = prop.substring(0, colon);
          final String ip   = prop.substring(colon + 1,
slash);
          int slash2 = prop.indexOf("/", slash + 1);
          final String usr = prop.substring(slash + 1,
slash2);
          final String pw = prop.substring(slash2);
          // Erweitere URL wieder um den ignorierten Teil:
          opts.put("URL",
                   opts.get("Url") + "," + host + ":" + ip +
"/" + usr + pw);
        }
      } else {
        prop = props[i].substring(0, idx).trim();
        val  = props[i].substring(idx + 1).trim();
      }
      ...
      opts.put(prop, val);
    }
    return opts;
  } catch (RuntimeException re) {
    throw new ParseException(re);
  }
}
@Before("execution(* solid.jdbc.SolidDriver.connect(..)) "
      + "&& within(solid.jdbc.*)")
// Dem der Methode SolidDriver.connect übergebenen
// Properties-Objekt wird noch solid_tf_level = 1 mitgegeben
public void addSolidTfLevel(final JoinPoint jp) throws Throw-
able {
  Object[] args = jp.getArgs();
  if (args != null && args.length > 1
    && args[0].getClass().equals(String.class)
    && args[0].toString().toLowerCase().contains("solid")
```

```
    //-> nur für Solid-DB
    && args[1].getClass().equals(Properties.class)) {
      // -> setze Property
      ((Properties) args[1]).setProperty("solid_tf_level",
"1");
    }
  }
}
```

[0065] Im Nachfolgenden wird eine kurze Erläuterung des oben gezeigten Software-Codes gegeben.

[0066] Bei dem Aspekt ChangeURLAspect handelt es sich um eine Java-Klasse, die durch eine @Aspect-Annotation zu einem Aspekt wird.

[0067] Die Java-Annotation @Around ersetzt den in der Ersatzkomponente existierenden Code. Die Annotation @Before führt den Code vor der Ausführung aus. Die Zeichenkette innerhalb der Annotation legt jeweils fest, welcher Code betroffen ist:

a) execution(”public static *
org..Configuraitons.parseProperties(String)):
Die Ausführung (”execution”) einer statischen Methode parseProperties einer Klasse Configurations mit einem String-Parameter und einem beliebigen Rückgabewert (*)

b) execution(* solid.jdbc.SolidDriver.connect(..)):
Die Ausführung einer connect-Methode der Klasse SolidDriver mit beliebigen Parametern und Rückgabewert

[0068] Im vorliegenden Ausführungsbeispiel wird statt der fehlerbehafteten Originalmethode parseProperties, die mit @Around annotierte Methode des Aspekts ChangeURLAspect, die das Problem behebt, ausgeführt.

[0069] Ferner wird vor der Ausführung der SolidDriver.connect-Methode in der ChangeURLAspect-Methode addSolidTfLevel die Property solid_tf_level gesetzt.

[0070] Des Weiteren sind Instruktionen zum Erstellen einer Ersatzbibliothek notwendig. In dem vorliegenden Ausführungsbeispiel werden diese beispielhaft und auszugsweise als build.xml für das Bau- bzw. Build-Tool ANT präsentiert:
Hierzu werden in der Datei build.xml die folgenden Schritte kodiert:

1. Übersetzen des obigen Aspektes

```
<!-- 1. Allgemeine Übersetzung der Aspekte -->
<target name = "compile">
  <mkdir dir = "classesaspect"/>
  <javac srcdir = "aspects" destdir = "classesaspect">
    <classpath refid = "project.classpath"/>
  </javac>
</target>

<taskdef name = "iajc"
  classname = "org.aspectj.tools.ant.taskdefs.AjcTask"
  classpath = "$(repository)/aspectj/jars/aspectjtools-
1.5.0.jar"/>
```

2. Erstellen einer Bibliothek libaspecturl.jar, die den übersetzten Aspekt enthält.

```
    <!-- 2. Bauen einer Bibliothek, die Aspekte enthält -->
    <target name = "buildaspectlib" description = "Compile as-
  pect and build library"
          depends = " compile">
      <iajc outjar = "lib-aspecturl.jar"
            XnoWeave = "true" source = "1.5">
        <classpath refid = "projectclasspath"/>
        <sourceroots location = "aspects/url"/>
      </iajc>
    </target>
```

3. Anwenden des Aspekts auf die zu ändernden Bibliotheken openjpa-0.9.7-incubating.jar und soliddriver-4.5.127.jar und Erzeugen neuer new-...-Varianten bzw. neuer Varianten der Bibliotheken der Ersatzkomponente. Im vorliegenden Beispiel sind dies new-openjpa-0.9.7.jar und new-soliddriver-4.5.127.jar.

```
  <!-3. Wende den Aspekt auf openjpa-0.9.7-incubating.jar und
      soliddriver-4.5.127.jar an -->
  <target name = "producelibs" depends = "buildaspectlib"
          description = "Enhance Java classes for patching
OpenJPA">
    <iajc outjar = "new-openjpa-0.9.7.jar">
      <classpath refid = "project.classpath"/>
      <incjars>
    <pathelement location = "openjpa-0.9.7-incubating.jar"/>
    <pathelement location = "libaspecturl.jar"/>
  </injars>
    <aspectpath>
      <pathelement location = "libaspecturl.jar"/>
    </aspectpath>
      </iajc>
      <iajc outjar = "new-soliddriver-4.5.127.jar">
      <classpath refid = "project.classpath"/>
      <injars>
        <pathelement location = "soliddriver-4.5.127.jar"/>
        <pathelement location = "libaspecturl.jar"/>
      </injars>
      <aspectpath>
        <pathelement location = "libaspecturl.jar"/>
      </aspectpath>
       </iajc>
  </target>
```

[0071] Somit wird im vorliegenden Ausführungsbeispiel das Austauschen einer Austauschkomponente ausgeführt, wobei gemäß dem vorgeschlagenen Verfahren die bei OpenJPA fehlende Unterstützung für die Datenbankausfallsicherheit und eine Verfügbarkeit von laufenden Diensten und Daten gewährleistet werden. Vor-

teilig ist hierbei, dass das vorgeschlagene Verfahren eine fehlerrobuste Migration von auszutauschenden Komponenten ermöglicht.

[0072] Insbesondere die Anwendung des aspektorientierten Programmierparadigmas, sowie die Verwendung dynamisch anbindbarer Bibliotheken ermöglichen eine flexible Migration von Komponenten.

[0073] Weiterhin ist vorteilhaft, dass der in den Bibliotheken gespeicherte Quellcode wieder verwendbar ist.

[0074] Somit betrifft die vorliegende Erfindung das Austauschen einer Austauschkomponente eines Computersystems gegen eine Ersatzkomponente im Rahmen eines Migrationsprozesses in dem Computersystem, wobei die Austauschkomponente in dem Computersystem durch die Ersatzkomponente ersetzt wird und wobei die Ersatzkomponente eine alternative Komponente zu der auszutauschenden Komponente darstellt und von gleicher Art wie die auszutauschende Komponente ist. Dabei weist das erfindungsgemäße Verfahren die folgenden Schritte auf: Ermitteln von Funktionalitätsunterschieden zwischen der Austauschkomponente und der Ersatzkomponente; Implementieren von Modifikationen der Ersatzkomponente mittels Aspektprogrammierung in einem sogenannten Aspekt, wobei das Implementieren auf den ermittelten Funktionalitätsunterschieden basiert; Übersetzen des Aspekts und Erzeugen einer den übersetzten Aspekt beinhaltenden Bibliothek; und Anwenden des Aspekts auf die als übersetzte Bibliothek vorliegende Ersatzkomponente. Auf diese Weise wird ein flexibles Durchführen einer Migration in einem Computersystem erreicht.

[0075] Obwohl die Erfindung oben unter Bezug auf die Ausführungsbeispiele gemäß der beiliegenden Zeichnungen erklärt wird, ist es ersichtlich, dass die Erfindung nicht auf diese beschränkt ist, sondern innerhalb des Bereichs der oben und in den anhängigen Ansprüchen offenbarten erfinderischen Idee modifiziert werden kann. Es versteht sich von selbst, dass es noch weitere Ausführungsbeispiele geben kann, die den Grundsatz der Erfindung darstellen und äquivalent sind, und dass somit verschiedene Modifikationen ohne Abweichen vom Umfang der Erfindung implementiert werden können. So können verschiedene Programmiersprachen, verschiedene Programmierparadigmen zum Implementieren von Aspekten gemäß der vorliegenden Erfindung verwendet werden. Ferner kann die vorliegende Erfindung in verschiedenen Anwendungsbereichen eines Computersystems eingesetzt werden, welche auf das Verwenden von Software angewiesen sind, z. B. um Steuerungs-, Betriebs-, Verwaltung- und/oder Organisationsprozesse durchzuführen.

Patentansprüche

1. Verfahren zum Austauschen einer Austauschkomponente (**1**) eines Computersystems gegen eine Ersatzkomponente (**2**), wobei die Austauschkomponente (**1**) in dem Computersystem durch die Ersatzkomponente (**2**) ersetzt wird, wobei die Ersatzkomponente (**2**) eine alternative Komponente zu der auszutauschenden Komponente (**1**) bildet und von gleicher Art wie die auszutauschende Komponente (**1**) ist, und wobei das Verfahren aufweist:
– Implementieren (S2) von Modifikationen der Ersatzkomponente (**2**) mittels Aspektprogrammierung in einem Aspekt (**4a**) basierend auf ermittelten Funktionalitätsunterschieden (**3a**) zwischen der Austauschkomponente (**1**) und der Ersatzkomponente (**2**);
– Übersetzen (S3) des Aspekts (**4a**) und Erzeugen einer den übersetzten Aspekt beinhaltenden Bibliothek (**5a**); und
– Anwenden (S4) des Aspekts (**4a**) auf die als übersetzte Bibliothek vorliegende Ersatzkomponente (**2**).

2. Verfahren nach Anspruch 1, wobei der Aspekt (**4a**) von der Ersatzkomponente (**2**) separat implementiert wird.

3. Verfahren nach Anspruch 1 oder 2, wobei das Anwenden (S4) des Aspekts (**4a**) auf die als übersetzte Bibliothek (**2**) vorliegende Ersatzkomponente (**2**) ein Erstellen (S43) einer neuen Bibliothek der Ersatzkomponente (**2**) aufweist.

4. Verfahren nach zumindest einem der vorstehenden Ansprüche, wobei das Verfahren ein Festlegen (S41) der Position des Aspekts (**4a**) in der Ersatzkomponente (**2**) aufweist.

5. Verfahren nach Anspruch 4, wobei das Anwenden (S4) des Aspekts (**4a**) auf die als übersetzte Bibliothek vorliegende Ersatzkomponente (**2**) unter Verwendung des Aspekts (**4a**) an der festgelegten Position in der Ersatzkomponente (**2**) durchgeführt (S42) wird.

6. Verfahren nach Anspruch 4 oder 5, wobei durch die Position festlegt wird, welcher Code der Ersatzkom-

ponente (**2**) modifiziert werden soll.

7. Anordnung zum Austauschen einer Austauschkomponente (**1**) eines Computersystems gegen eine Ersatzkomponente (**2**), wobei die Austauschkomponente (**1**) in dem Computersystem durch die Ersatzkomponente (**2**) ersetzt wird, wobei die Ersatzkomponente (**2**) eine alternative Komponente zu der auszutauschenden Komponente (**1**) bildet und von gleicher Art wie die auszutauschende Komponente (**1**) ist, und wobei die Anordnung aufweist:
– eine Funktionsanalysevorrichtung (**3**) zum Ermitteln von Funktionalitätsunterschieden (**3a**) zwischen der Austauschkomponente (**1**) und der Ersatzkomponente (**2**);
– eine Anpassungsvorrichtung (**4**) zum Implementieren von Modifikationen der Ersatzkomponente (**2**) mittels Aspektprogrammierung in einem Aspekt (**4a**), wobei das Implementieren auf den ermittelten Funktionalitätsunterschieden (**3a**) basiert;
– eine Übersetzungsvorrichtung (**5**) zum Übersetzen des Aspekts (**4a**) und Erzeugen einer den übersetzten Aspekt beinhaltenden Bibliothek (**5a**); und
– eine Anwendungsvorrichtung (**6**) zum Anwenden des Aspekts (**4a**) auf die als übersetzte Bibliothek (**5a**) vorliegende Ersatzkomponente

8. Computerprogrammprodukt, welches ein Verfahren nach einem der Ansprüche 1 bis 6 auf einer Rechneranlage durchführt.

Es folgt ein Blatt Zeichnungen

Anhängende Zeichnungen

FIG 1

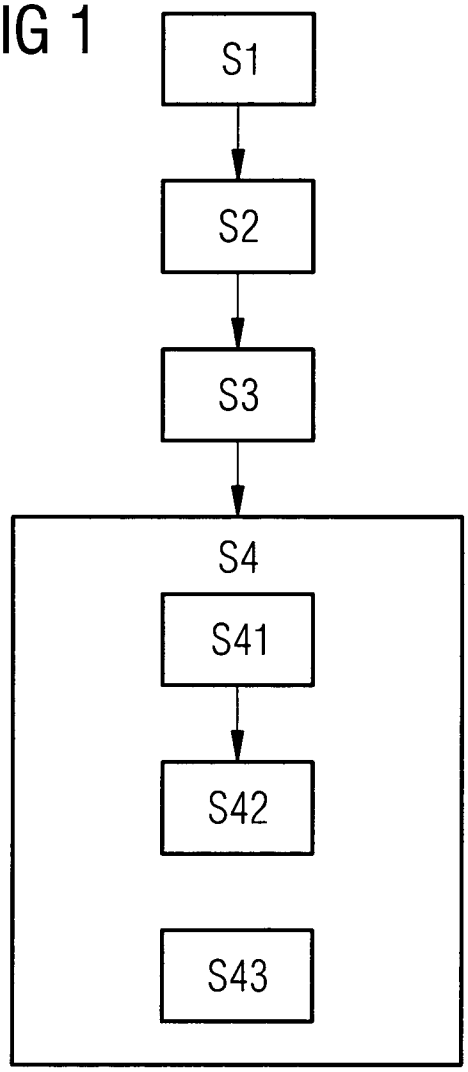


FIG 2