BUNDESPATENTGERICHT

7 W (pat) 76/11

_____

(Aktenzeichen)

Verkündet am
15. März 2013
…

B E S C H L U S S

In der Beschwerdesache

betreffend die Patentanmeldung 10 2004 012 315.2-53

…

hat der 7. Senat (Technischer Beschwerdesenat) des Bundespatentgerichts auf die mündliche Verhandlung vom 15. März 2013 durch den Richter Dipl.-Phys. Dipl.-Wirt.-Phys. Maile als Vorsitzenden sowie die Richter Schwarz, Dipl.-Phys. Dr. Schwengelbeck und die Richterin Dipl.-Phys. Dr. Otten-Dünnweber

beschlossen:

Die Beschwerde wird zurückgewiesen.

G r ü n d e

I.

Die Prüfungsstelle für Klasse G 06 F des Deutschen Patent- und Markenamts hat mit Beschluss vom 25. Juni 2009 die Patentanmeldung 10 2004 012 315.2-53 mit der Bezeichnung

Verfahren zur automatischen Anpassung von Software

zurückgewiesen, weil der Gegenstand des damals geltenden Anspruchs 1, eingegangen am 05. März 2008, im Hinblick auf die im Prüfungsverfahren ermittelte Druckschrift

D 2: HANSELMANN, H.; KIFFMEIER, U.; KOSTER, L.; MEYER, M.; RUKGAUER, A.: Production quality code generation from Simulink block diagrams, Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on

nicht neu sei.

Gegen diesen Beschluss richtet sich die Beschwerde der Anmelderin.

Im Prüfungsverfahren waren bereits die folgenden Druckschriften genannt:

D1: US 6 654 954 B1

D3: US 6 609 248 B1

Mit Fax vom 12. März 2013 hat der Senat des weiteren zur Vorbereitung auf die mündliche Verhandlung auf die von ihm ermittelte Druckschrift

D4: DE 202 11 265 U1

hingewiesen.

In der mündlichen Verhandlung begehrt die Anmelderin die Erteilung des Patents mit einer geänderten Anspruchsfassung laut Hauptantrag sowie einer geänderten Anspruchsfassung laut Hilfsantrag. Sie macht hierzu geltend, dass die geänderten Anspruchsfassungen jeweils zulässig, neu und erfinderisch seien.

Der seitens des Senats mit einer Gliederung versehene Patentanspruch 1 nach Hauptantrag lautet:

M1) Verfahren zur automatischen Anpassung einer plattformunabhängig programmierten Steuersoftwarekomponente an eine Steuergeräteplattform mit netzwerkverbundenen Steuergeräten, um ein Ablaufen der Steuersoftwarekomponente in der Steuergeräteplattform zu ermöglichen,

M2) wobei die Steuersoftwarekomponente plattformunabhängig unter Verwendung von Middleware-Programmschritten durch einen Programmierer programmiert wird,

dadurch gekennzeichnet, dass

M3) im Anschluss dieser Programmierung oder während dieser Pro-

grammierung computerimplementiert durch eine Software

M4) plattformunabhängige Middleware-Befehle der Steuersoftwarekomponente für das Versenden einer Nachricht über das Netzwerk im Programmcode ersetzt werden durch plattformabhängige Befehle,

M5) wobei die Software Zugriff hat auf Konfigurationsinformationen, in denen die konkrete Hardwarearchitektur abgelegt ist und denen entnommen wird, wie ein plattformunabhängiger Middleware-Befehl gegen einen plattformabhängigen zu ersetzen ist,

M6) wobei während der Entwicklungszeit der plattformunabhängigen Software-Komponente deren Anpassung und die Erstellung einer plattformabhängigen Software-Komponente erfolgt

M7) und eine Middleware zur Laufzeit vollständig oder zumindest teilweise überflüssig wird.

Der seitens des Senats mit einer Gliederung versehene Patentanspruch 1 nach Hilfsantrag lautet:

M1) Verfahren zur automatischen Anpassung einer plattformunabhängig programmierten Steuersoftwarekomponente an eine Steuergeräteplattform mit netzwerkverbundenen Steuergeräten, um ein Ablaufen der Steuersoftwarekomponente in der Steuergeräteplattform zu ermöglichen,

M2) wobei die Steuersoftwarekomponente plattformunabhängig unter Verwendung von Middleware-Programmschritten durch einen Programmierer programmiert wird,

dadurch gekennzeichnet, dass

M2a) zum Zeitpunkt der Quell-Codeerstellung von Funktionscode für eine Software-Komponente A der Programmierer einen plattform-

unabhängigen Middleware-Befehl für das Senden einer Nachricht von der Software-Komponente A an eine Software-Komponente B, die sich auf einem anderen über das Netzwerk verbundenen Steuergerät befindet, ohne Bezug auf die Art und Weise der physikalischen Versendeart erstellt und

M3n) computerimplementiert durch eine Software der Quell-Code auf diesen plattformunabhängigen Programmschritt hin untersucht wird und

M4n) aufgrund der statischen Situation, dass die Softwarekomponenten zur Laufzeit immer an der gleichen Stelle sind, die Versendeart für die gesamte Zeit festlegt und den Befehl für das Versenden der Nachricht via Netzwerk direkt in den Quellcode der Software-Komponente schreibt,

M5) wobei die Software Zugriff hat auf Konfigurationsinformationen, in denen die konkrete Hardwarearchitektur abgelegt ist und denen entnommen wird, wie ein plattformunabhängiger Middleware-Befehl gegen einen plattformabhängigen zu ersetzen ist,

M6) wobei während der Entwicklungszeit der plattformunabhängigen Software-Komponente deren Anpassung und die Erstellung einer plattformabhängigen Software-Komponente erfolgt

M7n) und die bislang in der Middleware befindliche Funktion, welche überprüft, wohin eine Nachricht gesendet werden soll und dafür den entsprechenden Befehl heraussucht und ausführt, überflüssig wird.

Wegen den nach Hauptantrag geltenden abhängigen Ansprüchen 2 bis 5 und den nach Hilfsantrag geltenden abhängigen Ansprüchen 2 bis 4 wird auf den Aktenin-halt verwiesen.

Die Anmelderin stellt den Antrag,

den Beschluss der Prüfungsstelle für Klasse G 06 F des Deutschen Patent- und Markenamtes vom 25. Juni 2009 aufzuheben und auf die Anmeldung ein Patent mit den folgenden Unterlagen zu erteilen:

- Patentansprüche 1 bis 5 laut dem in der mündlichen Verhandlung vom 15. März 2013 überreichten neuen Hauptantrag

  hilfsweise

  Patentansprüche 1 bis 4 laut dem in der mündlichen Verhandlung vom 15. März 2013 überreichten Hilfsantrag

- ggfs. anzupassende Beschreibung und Zeichnungen (Fig. 1 bis 4a) laut Offenlegungsschrift.

Wegen der weiteren Einzelheiten wird auf den Akteninhalt verwiesen.

II.

Die zulässige Beschwerde hat in der Sache keinen Erfolg. Denn nach dem Ergebnis der mündlichen Verhandlung beruhen die Gegenstände der jeweiligen Ansprüche 1 nach Hauptantrag sowie nach Hilfsantrag nicht auf einer erfinderischen Tätigkeit des Fachmanns. Dieser ist vorliegend als Diplom-Informatiker mit mehrjähriger Erfahrung auf dem Gebiet der Software-Portierung und der Anpassung von Software an verschiedene Hardwaresysteme zu definieren. Die Fragen der Zulässigkeit der geltenden Ansprüche nach Haupt- und Hilfsantrag sowie der Neuheit

der Anspruchsgegenstände können somit dahinstehen (vgl. BGH GRUR 1991, 120, 121 li. Sp. Abs. 3 - „Elastische Bandage“).

1) Die Patentanmeldung betrifft ein Verfahren zur automatischen Anpassung einer plattformunabhängig programmierten Steuersoftwarekomponente an eine Steuergeräteplattform (vgl. Offenlegungsschrift, Abs. [0001]). Gemäß Abs. [0006] der Offenlegungsschrift ist es aus dem Stand der Technik bekannt, die Anpassung einer Softwarekomponente an die Hardware, also die Steuergeräteplattform, nicht in der Softwarekomponente selbst durchzuführen, sondern außerhalb in einer zusätzlichen Softwareschicht, der sogenannten Middleware. Die Middleware ermittelt dabei zur Laufzeit anhand der in ihr festgelegten Konfiguration den Aufenthaltsort der Software-Komponenten und führt die plattformabhängigen Funktionen aus (vgl. Fig. 1a und Abs. [0017] der Offenlegungsschrift). Bei der Verwendung einer Middleware sind die zusätzlich benötigten Kommunikationsschichten, der zusätzliche Speicherplatz, die zusätzliche Prozessorleistung und die zusätzliche Software von Nachteil (vgl. Abs. [0020] der Offenlegungsschrift).

   Der Patentanmeldung liegt dabei die Aufgabe zugrunde, die Middleware-typischen Vorteile wie wiederverwertbare Software-Komponenten und Abstrahierung von der Hardwareplattform zu nutzen, dabei aber die typischen Nachteile zu vermeiden (vgl. Abs. [0021] der Offenlegungsschrift).

   Als Middleware ist gemäß der Beschreibung Software zu verstehen, die als „Vermittler“ zwischen der wiederverwendbaren Software-Komponente und den weiteren Software- und Hardwareschichten einer Steuergeräteplattform fungiert und damit die gerätespezifischen Hardware- wie auch Software-Schnittstellen beispielsweise zum Betriebssystem oder zu Kommunikationsschichten abstrahiert (vgl. Abs. [0005] - [0007] und [0009] der Offenlegungsschrift). Als weitere Middleware-typische Dienste werden etwa die Unterstützung von Kommunikationsschnittstellen in einem Netzwerk (Proxy-

Unterstützung) oder Verzeichnisdienste (Naming Service) genannt (vgl. Abs. [0010] der Offenlegungsschrift).

Gemäß den Ausführungen der Beschreibung ist ein wesentlicher Vorteil des beanspruchten Verfahrens, dass schon während der Entwicklungszeit einer plattformunabhängigen Software-Komponente die Anpassung und Erstellung einer plattformabhängigen Software-Komponente erfolgt, wodurch die bekannte Middleware zur Laufzeit vollständig oder zumindest teilweise überflüssig wird (vgl. Abs. [0028] der Offenlegungsschrift, siehe Merkmal M7). Zum Entwicklungszeitpunkt werden dem Programmierer die middleware-typischen abstrakten Schnittstellen-Funktionalitäten angeboten, so dass eine wiederverwendbare Anwendung programmiert werden kann, die über Rechner- bzw. Systemplattformen hinweg zum Einsatz kommen kann (vgl. die Abs. [0029] und [0030] der Offenlegungsschrift). Zur Laufzeit hingegen – wie für ein Ausführungsbeispiel erläutert wird – existiert gemäß der Erfindung keine Middleware mehr und in den Softwarekomponenten werden die plattformabhängigen Funktionen direkt aufgerufen (vgl. Abs. [0071] der Offenlegungsschrift). Demgemäß ist der Begriff eines Middleware-Programmschritts bzw. eines Middleware-Befehls im Lichte der Beschreibung als ein Software-Programmschritt bzw. ein Software-Befehl zu verstehen, der von der gerätespezifischen Hard- oder Softwareschnittstelle abstrahiert ist oder der einen abstrakten Funktionsaufruf darstellt (vgl. Abs. [0033] der Offenlegungsschrift). Unter diese allgemeine Definition fallen auch plattformunabhängig formulierte Programmschritte, die beispielsweise die Kommunikation der Schnittstellen zwischen verschiedenen Programmkomponenten abstrakt definieren.

2) Zum Hauptantrag

Das Verfahren des Patentanspruchs 1 gemäß Hauptantrag beruht nicht auf einer erfinderischen Tätigkeit gegenüber dem Stand der Technik gemäß Druckschrift **D3** i.V.m. dem fachmännischen Handeln.

Die Druckschrift D3 offenbart ein Verfahren zur automatischen Anpassung (D3: Anspruch 1: *computerized method;* Sp. 3, Z. 32: *automatically adjusting)* einer plattformunabhängig programmierten Softwarekomponente (D3: vgl. Fig. 1, Anspruch 1 und Sp. 3, Z. 19 - 25: *An output translator provides for cross module representations of components within a heterogeneous program by enabling a code block in a component to be translated from an platform-neutral intermediate representation of the program into a set of platform-specific instructions that are directed to a different architecture than that for which the code block was originally written.*). Die in Merkmal M1 geforderte „Steuergeräteplattform mit netzwerkverbundenen Steuergeräten“ wird in der Beschreibung der Anmeldung so erläutert, dass es sich um eine aus Hardware-Komponenten bestehende Plattform handelt, deren verschiedene, nicht näher spezifizierte Steuergeräte in einem vernetzten System miteinander kommunizieren (vgl. Abs. [0002], [0049] und [0063] der Offenlegungsschrift). Eine solche Steuergeräteplattform stellt das in der D3 beschriebene Computer-System dar, das mehrere Laufwerke mit Schnittstellen, Prozessoren sowie Netzwerk-Schnittstellen aufweist (vgl. Fig. 1, *interfaces 32, 33, 34; 46; 53* sowie Sp. 4, Z. 59 *multiprocessor systems,* Z. 61 *network PC's,* ferner Sp. 5, Z. 66 - Sp. 6, Z. 2: *The remote computer may be another computer, a server, a router, a network PC, a client, a peer device or other common network node*). Das in der D3 erläuterte Verfahren kann in verschiedenen Konfigurationen von Computersystemen zum Einsatz kommen, neben Multiprozessorsystemen auch für Netzwerke von mehreren Computern (vgl. Sp. 4, Z. 62: *practiced in distributed computing environments,* Z. 63 - 65: *where tasks are performed by remote processing*

devices that are linked through a communications network). Software, die in einem solchen vernetzten System laufen soll, muss zwingend auch Komponenten enthalten, die für eine geordnete Ansteuerung der verschiedenen Netzwerkkomponenten sorgen – in der D3 ist in diesem Zusammenhang die Zwischenrepräsentation IR genannt, welche eine Steuersoftwarekomponente (vgl. D3, Sp. 7, Z. 60: *code and data*) umfasst - die an eine Steuergeräteplattform mit netzwerkverbundenen Steuergeräten angepasst ist, um ein Ablaufen der Steuersoftwarekomponente in der Steuergeräteplattform zu ermöglichen (vgl. Sp. 14, Z. 49 - 50: *an intermediate representation that is a hierarchy of platform-neutral elements that correspond to instructions* / **Merkmal M1**).

In der D3 nimmt ein Programmierer an einer Zwischenrepräsentation IR Änderungen vor (siehe die Fig. 2A und Sp. 7, Z. 59 - 62: *Once the intermediate representation is complete, the user is allowed to manipulate the code and data (illustrated by the IR transformation module 230) through an application program interface (API) 250.*). Die Steuersoftwarekomponente wird dabei unter Verwendung von plattformunabhängigen Programmschritten durch einen Programmierer programmiert (vgl. Sp. 7, Z. 67 - Sp. 8, Z. 1: *The API 250 also permits the user direct access 232 to the IR to navigate through the IR and to make changes*.). Die plattformunabhängigen Programmschritte sind somit abstrahiert von den konkreten gerätespezifischen Hard- oder Softwareschnittstellen und stellen damit Middleware-Programmschritte im Sinne der vorliegenden Anmeldung dar (siehe Abschnitt 1).

Ob – wie in der mündlichen Verhandlung thematisiert – die die plattformunabhängigen Programmschritte enthaltende Softwarekomponente originär durch den Programmierer selbst oder wie in der D3 (siehe Abstract) ggf. erst durch eine Transformation aus einem heterogenen Programm erzeugt wird, ist nicht Bestandteil des vorliegenden Anspruchs und daher für die

Klärung der Frage der Patentfähigkeit des beanspruchten Verfahrens unerheblich. Die D3 offenbart damit das **Merkmal M2**.

An der durch den Programmierer bearbeiteten Softwarekomponente (*transformed IR*) werden durch einen Übersetzer 240, also computerimplementiert durch eine Software, Ersetzungen plattformunabhängiger Befehle vorgenommen (D3: vgl. Sp. 8, Z. 22: *The transformed IR is now input into the output translator*; Anspruch 1: *a computerized method for translating*), wobei der Übersetzer zweiphasig arbeitet (vgl. Sp. 8, Z. 23 - 24: *The output translator 240 operates on the IR in two phases*; vgl. auch Fig. 2A und 2D). Damit ist der D3 zu entnehmen, dass die Ersetzung computerimplementiert im Anschluss oder während der Programmierung durch eine Software erfolgt (**Merkmal M3 in einer beanspruchten Alternative**).

Im Merkmal M4 sind die vorstehend genannten zu ersetzenden plattformunabhängig programmierten Befehle als „Middleware-Befehle der Steuersoftwarekomponente für das Versenden einer Nachricht über das Netzwerk“ konkretisiert. Das Versenden einer Nachricht bezieht sich in der vorliegenden Anmeldung auf Kommunikationsbefehle zwischen verschiedenen Software-Komponenten (vgl. Abs. [0014] - [0017] und Abs. [0037] der Offenlegungsschrift). Auch die in der D3 bearbeiteten Softwarekomponenten finden in vernetzten Computersystemen mit verschiedenen netzwerkverbundenen Steuergeräten Anwendung (siehe obige Ausführungen zu Merkmal M1). Die Steuersoftware läuft dabei nicht allein auf einem einzelnen Steuergerät, sondern die verschiedenen Softwarekomponenten sind auf mehreren, durch ein Netzwerk verbundenen, Steuergeräten implementiert (D3: Sp. 5, Z. 61 - 63: *The computer may operate in a networked environment using logical connections to one or more remote computers*, sowie Sp. 4, Z. 66 - 67: *program modules may be located in both local and remote memory storage devices*). Dies impliziert, dass die Steuersoftwarekomponenten miteinander kommunizieren, was regelmäßig über das Versenden

von Nachrichten im Sinne der Anmeldung erfolgt. Ein Programmierer, der eine Softwarekomponente für eine Plattform mit netzwerkverbundenen Steuergeräten – wie in der D3 erläutert (siehe Ausführungen zu Merkmal M2) – plattformunabhängig programmiert, muss daher auch auf plattformunabhängige Befehle zurückgreifen, die das Versenden einer Nachricht über das Netzwerk – im Sinne der Anmeldung – betreffen. In der D3 wird die plattformunabhängige Zwischenrepräsentation *IR* blockweise einem zweiphasigen Übersetzungsprozess (*output translator 240*) unterworfen: In einer Link-Phase (*linker 241*, vgl. Fig. 2D und die Schritte *401 - 404* in Fig. 4A, vgl. Sp. 10, Z. 5 - 8) werden die logischen Verbindungen in absolute Adressen umgesetzt (vgl. Sp. 8, Z. 24 - 25), sowie Modifikationen wie das Hinzufügen von erforderlichem Prolog- oder Epilog-Code und weitere Optimierungen am Programmcode vorgenommen (vgl. Sp. 8, Z. 37 - 42: *The linker 241 is also responsible for adding whenever prologue and epilogue code necessary to "glue" together contiguous blocks that will be assembled into different platform-dependent instructions. As part of the address resolution, the linker 241 also can perform limited code modification or optimization.*). An die Link-Phase schließt sich eine Schreib-Phase an (*writer 242*, vgl. Fig. 2D und die Schritte *405* und *406* in Fig. 4A), in der jeder plattformunabhängige Befehl der Zwischenrepräsentation in die plattformabhängige Entsprechung überführt wird (vgl. Sp. 8, Z. 56 - 58 und Z. 61 - 64: *The writer 242 assembles each IR instruction into its platform-dependent counterpart based on the architecture specified in the code block. If the complex instruction is designated to be translated into a different architecture, the writer 242 creates the appropriate set of platform-specific instructions...*). Die D3 offenbart damit das Ersetzen von plattformunabhängigen Middleware-Befehlen im Programmcode von Steuersoftwarekomponenten für das Versenden einer Nachricht über das Netzwerk durch plattformabhängige Befehle (**Merkmal M4**).

Bei der Ersetzung der Befehle hat die Software Zugriff auf Konfigurationsinformationen (D3: vgl. Sp. 13, Z. 61 - 62: *the linker/writer method 400 uses an translation data structure)*, in denen die konkrete Hardwarearchitektur abgelegt ist (vgl. Sp. 13, Z. 66 - Sp. 14, Z. 3: *There is a translation data structure for each architecture supported by the system and each translation data structure contains entries for only those platform-specific instructions that cannot be represented by the basic IR instruction form.*). Diesen Konfigurationsinformationen kann entnommen werden, wie die plattformunabhängigen Befehle durch plattformabhängige zu ersetzen sind (vgl. Sp. 15, Z. 16 - 19: *the translation from the IR instructions into the platform-specific instructions can be accomplished through the use of look-up tables, hashing function, or database records*. / **Merkmal M5**$_{\textbf{teilweise / ohne Detailangaben}}$).

Die D3 macht jedoch keine konkreten Angaben, welche Befehlsersetzungen in den Konfigurationsinformationen enthalten sind (vgl. Sp. 14, Z. 4 - 5: *The translation data structures are created outside the system*). Der Fachmann hat daher Veranlassung bei der Realisierung des in der D3 beschriebenen Verfahrens die Konfigurationsinformationen an den jeweiligen Anwendungsfall anzupassen. Bei einer Steuersoftware, die in solch komplexen Computersystemen wie in der D3 beschrieben (vgl. Fig. 1) implementiert werden soll, muss der Programmierer, dem uneingeschränkt eine plattformunabhängige Programmierung erlaubt sein soll (D3: vgl. Sp. 9, Z. 62 - 63: *any desired transformations 230 are performed on the IR hierarchy 220 by a user*), auch Programmschritte vorsehen, die die Kommunikation und den Datenaustausch zwischen den verteilten Softwarekomponenten betreffen. Der Fachmann wird daher in naheliegender Weise bei der konkreten Ausgestaltung des aus der D3 bekannten Verfahrens in den Konfigurationsinformationen auch die Kommunikation betreffende, plattformunabhängige Middleware-Befehle – im Sinne der An-

meldung – mit ihren entsprechenden Ersetzungsvorschriften hinterlegen (**Merkmal M5$_{\text{Rest / Detailangaben}}$**).

Auch das Merkmal M6 ist der Druckschrift D3 zu entnehmen, denn bei dem in der D3 offenbarten Verfahren nimmt ein Programmierer an der Zwischenrepräsentation Änderungen vor (siehe die Ausführungen zu Merkmal M2), so dass in jedem Fall während der Entwicklungszeit der plattformunabhängigen Software-Komponente, d.h. vor deren Kompilierung, deren Anpassung erfolgt. Bereits in der ersten Phase des zweistufigen Übersetzungsprozesses (siehe Ausführungen zu Merkmal M4), der Link-Phase, die jedenfalls vor der Kompilierung durchlaufen wird, werden Ersetzungen vorgenommen und an die Hardwareplattform angepasste Komponenten erzeugt (vgl. Fig. 4A, Schritt *401 block substitution, 403 assign addresses*). Damit erfolgt auch eine Erstellung einer plattformabhängigen Software-Komponente während der Entwicklungszeit (**Merkmal M6)**.

Bei der Abarbeitung des in der D3 beschriebenen Verfahrens wird für alle plattformunabhängigen Befehle, für die in den Konfigurationsinformationen ein zugehöriger plattformabhängiger Befehl hinterlegt ist, eine Ersetzung vorgenommen, so dass für diese Befehle zur Laufzeit bereits ein ausführbarer, an die Architektur angepasster Befehl vorliegt (vgl. Fig. 2A, *executable EXE‘ 203*); für all diese Befehle ist daher zur Laufzeit keine Middleware vonnöten, so dass diese zur Laufzeit zumindest teilweise überflüssig wird (**Merkmal M7 in einer beanspruchten Alternative)**.

Den Ausführungen der Anmelderin in der mündlichen Verhandlung, in der D3 sei keine Middleware beschrieben bzw. nahegelegt, weswegen in der Steuersoftwarekomponente auch keine plattformunabhängigen Middleware-Befehle für das Versenden von Nachrichten zu ersetzen seien, kann seitens des Senats nicht gefolgt werden. Weder in Merkmal M2 noch in den Merkmalen M4 oder M5 oder in der Beschreibung ist angegeben, dass der Pro-

grammierer auf Programmschritte eines speziellen Middlewareprodukts zurückgreift; unter die Begriffe Middleware-Programmschritt (Merkmal M2) bzw. Middleware-Befehl (Merkmale M4, M5) sind damit für den Fachmann sämtliche abstrahierte Programmschritte bzw. Befehle subsumiert, die z.B. die Kommunikation oder den Datenaustausch zwischen Applikationen betreffen und vom Programmierer plattformunabhängig formuliert werden (vgl. hierzu auch die Ausführungen in Abschnitt 1)).

Der Fachmann gelangt somit, ausgehend von der technischen Lehre der Druckschrift D3 zur automatischen Anpassung einer Steuersoftwarekomponente an eine Steuergeräteplattform, und unter Bereitstellung von Konfigurationsinformationen, die er gemäß seines Fachwissens für den jeweiligen Anwendungsfall entsprechend anpasst, in nahe liegender Weise zum Verfahren des geltenden Patentanspruchs 1 zumindest in einer Alternative nach Hauptantrag, ohne erfinderisch tätig werden zu müssen.

Der Anspruch 1 nach Hauptantrag ist daher nicht patentfähig.

3) Zum Hilfsantrag

Auch das Verfahren des Patentanspruchs 1 gemäß Hilfsantrag beruht nicht auf einer erfinderischen Tätigkeit, da es sich für den Fachmann in naheliegender Weise aus dem Stand der Technik gemäß Druckschrift **D3** ergibt.

3.1) Im Patentanspruch 1 des Hilfsantrags sind die Merkmale des Oberbegriffs (Merkmale M1 und M2) sowie die Merkmale M5 und M6 wortgleich mit den entsprechenden Merkmalen des Patentanspruchs 1 nach Hauptantrag, so dass hinsichtlich dieser Merkmale auf die Ausführungen unter Abschnitt 2) verwiesen wird. Mit dem als erstes Merkmal im kennzeichnenden Teil eingefügten Merkmal M2a sowie den gegenüber dem Hauptantrag geänderten

Merkmalen M3n und M4n ist statt einer Ersetzung im Programmcode eine Untersuchung des Quell-Codes gefordert.

3.2) Auch das zusätzlich zum Hauptantrag aufgenommene Merkmal M2a, wonach zum Zeitpunkt der Quell-Codeerstellung von Funktionscode für eine Software-Komponente A der Programmierer einen plattformunabhängigen Middleware-Befehl für das Senden einer Nachricht von der Software-Komponente A an eine Software-Komponente B, die sich auf einem anderen über das Netzwerk verbundenen Steuergerät befindet, ohne Bezug auf die Art und Weise der physikalischen Versendeart erstellt, kann keine erfinderische Tätigkeit begründen.

Bei dem in der D3 offenbarten Verfahren arbeitet der Programmierer an der Zwischenrepräsentation, die plattformunabhängige Befehle enthält. Die Zwischenrepräsentation wird zunächst durch einen der eigentlichen Programmierung vorgeschalteten Schritt einer Eingangs-Übersetzung aus einer ausführbaren Programmkomponente erzeugt (vgl. Fig. 2A, Block *210 input translation (reader)*). Bei diesem Prozess werden keinerlei Optimierungen vorgenommen, die Zwischenrepräsentation soll sich möglichst nah an ein Programm anlehnen, wie es ein Programmierer ursprünglich schrieb (vgl. Sp. 7, Z. 36 - 37: *it is desirable that the intermediate representation be as close to what the programmer originally wrote as possible*), was im Idealfall bedeutet, dass die Zwischenrepräsentation Quellcode darstellt. Der Programmierer kann im Schritt 230 (*transformation of IR*) an diesem Quellcode beliebige Änderungen vornehmen und plattformunabhängige Befehle erstellen. Da es in der D3 um Software für ein System aus mehreren vernetzten Steuergeräten geht, wird der Fachmann – wie schon zum Anspruch 1 nach Hauptantrag ausgeführt – bei den plattformunabhängigen Befehlen auch Middleware-Befehle für das Versenden einer Nachricht von einer Software-Komponente (A) an eine andere Software-Komponente (B), die sich auf einem anderen über das Netzwerk verbundenen Steuergerät

befindet, über das Netzwerk vorsehen. Allein schon der Begriff „plattformunabhängig“ beinhaltet dabei, dass diese Befehle vom Programmierer ohne Bezug auf die Art und Weise der physikalischen Versendeart erstellt werden (D3: vgl. Sp. 8, Z. 49 - 51: *the linker 241 has more knowledge of the placement of instructions than did the programmer*). Damit ist auch das **Merkmal M2a** in der D3 offenbart.

An der transformierten, also durch den Programmierer bearbeiteten Zwischenrepräsentation werden in der D3 durch den Übersetzer 240 computerimplementiert Ersetzungen vorgenommen. In die Link-Phase geht als Eingangsgröße direkt die transformierte Zwischenrepräsentation ein (D3: vgl. Fig. 2A und 2D: *linker 241* als erste Phase des *output translator 240* und Sp. 9, Z. 62 - 63: *After any desired transformations 230 are performed on the IR hierarchy 220 by a user, the […] linker/writer method 400 takes each component at a time and translates the IR instructions in the component into platform-specific instructions*), was bedeutet, dass mittels *linker 241* der Quell-Code durch eine Software auf diesen plattformunabhängigen Programmschritt hin untersucht wird (**Merkmal M3n**).

In Merkmal M4n ist ausgeführt, dass „aufgrund der statischen Situation, dass die Softwarekomponenten zur Laufzeit immer an der gleichen Stelle sind, die Versendeart für die gesamte Zeit“ festgelegt wird. Ob eine solche statische Situation vorliegt, wird von der speziellen Ausbildung der Steuergeräteplattform im Einzelfall abhängen, die Information muss dem Verfahren aber in den Konfigurationsinformationen zur Hardwarearchitektur (siehe die Ausführungen zum Merkmal M5 nach Hauptantrag) zur Verfügung gestellt werden. Bei der Hinterlegung der Konfigurationsinformationen handelt es sich um eine fachübliche Tätigkeit, der Fachmann muss die Architekturinformationen dem jeweiligen Einzelfall anpassen. Die Software soll ferner „den Befehl für das Versenden der Nachricht via Netzwerk direkt in den Quellcode der Softwarekomponente“ schreiben. In der D3 werden bereits in

der Link-Phase Änderungen und Ergänzungen gemäß der späteren Hardware-Architektur vorgenommen (siehe auch die Ausführungen zu Merkmal M4 des Hauptantrags). Erläutert werden die in der Link-Phase vorgenommenen Übersetzungen am Beispiel von allgemeinen Programmcode-Blöcken mit Befehlen, denen für die Ausführung des Programms in der speziellen Hardwarearchitektur absolute Adressen zugewiesen werden (vgl. Sp. 8, Z. 24 - 26: *a linker phase 241 that resolves the logical connections into absolute addresses in an address space für a modified version of the executable*). Sofern der Programmierer in die Zwischenrepräsentation auch Befehle, die das Versenden von Nachrichten via Netzwerk betreffen, geschrieben hat, werden diese bei der blockweisen Bearbeitung in der Link-Phase ebenfalls bearbeitet (vgl. Sp. 8, Z. 41 - 42: *the linker 241 also can perform limited code modification or optimization*). Dies impliziert, dass die Software direkt in den Quell-Code der Software-Komponente schreibt, da in der D3 erst in der anschließenden Schreib-Phase der modifizierte ausführbare Befehlssatz erzeugt wird (vgl. Sp. 8, Z. 26 - 27: *a writer phase 242 that assembles the IR into the modified version of the executable (EXE') /* **Merkmal M4n**).

Die Anmelderin hat in Bezug auf das vorstehend abgehandelte Merkmal M4n ausgeführt, bei dem aus der D3 bekannten Verfahren würden durch die Software keine Ersetzungen oder Änderungen am Quellcode vorgenommen werden, da bei dem dort beschriebenen iterativen Ablauf (vgl. Sp. 9, Z. 2 - 7, Pfeil *260* in Fig. 2A) immer wieder auf den ausführbaren Programmcode (*EXE 201, EXE' 203* in Fig. 2A) zugegriffen werde. Es ist zwar zutreffend, dass bei dem durch den Pfeil 260 in Fig. 2A angezeigten Ablauf bei jeder Iteration der gesamte Prozess durchlaufen wird, nämlich vom ausführbaren Code EXE 201 über die Eingangs-Übersetzung (*reader 210*) in die Zwischenrepräsentation – die der Programmierer bearbeitet – weiter über die Ausgangs-Übersetzung 240 in den modifizierten ausführbaren Code EXE' 203, welcher wiederum der nächsten Eingangs-Übersetzung zu

Grunde gelegt wird. Dieser Ablauf impliziert aber – entgegen der Argumentation der Anmelderin – nicht, dass während der zweiphasig implementierten Ausgangs-Übersetzung nicht auch der Quell-Code untersucht und modifiziert wird. Vielmehr nimmt die Software in der Link-Phase, wie erläutert, Änderungen am Quell-Code vor. Zudem wird in der D3 explizit ein alternativer Iterationsprozess beschrieben, der durchlaufen werden kann, bevor der ausführbare Code EXE 203 durch die Schreib-Phase 242 erstellt ist (vgl. Sp. 9, Z. 13 - 20: *In an alternate exemplary embodiment of the translation and transformation system 200 not illustrated, the IR containing the absolute addresses assigned by the linker 241 is used as input into the IR creation process 212 for further iteration through the system 200. […] much of the work performed by the creation process 212 […] can be skipped when iterating the modified IR through the system 200.*). Bei diesem alternativen Iterationsprozess wird nur die Zwischenrepräsentation, die als Quell-Code vorliegt, iterativ durch den Programmierer und die die Link-Phase durchführende Software modifiziert, was dem Programmierer die Arbeit erleichtern kann (vgl. Sp. 9, Z. 20 - 23: *This embodiment allows the user to transform a heterogeneous program in stages rather than having to make all the changes in a single pass through the system 200.*). Erst im Anschluss an diese iterativ durchlaufenen Programmier- und Link-Phasen wird die Software in der Schreib-Phase aus der Zwischenrepräsentation den ausführbaren Code erzeugen.

Auch das gegenüber dem Hauptantrag, Merkmal M7, geänderte Merkmal M7n kann eine erfinderische Tätigkeit nicht begründen. Denn durch das in der D3 beschriebene Verfahren werden die plattformunabhängigen Programmschritte durch plattformabhängige Programmschritte ersetzt (D3: vgl. Anspruch 1: *emitting a platform-specific executable instruction for each instruction represented in the intermediate representation*). Diese Ersetzung erfolgt im Quell-Code wie auch im ausführbaren Programmcode, in jedem Fall aber noch vor der Laufzeit, so dass eine Middleware zur Laufzeit für

alle ersetzten Befehle überflüssig ist und auf eine in der Middleware befindliche Funktion zur Überprüfung, wohin eine Nachricht gesendet werden soll und die dafür den entsprechenden Befehl heraussucht und ausführt, verzichtet werden kann (**Merkmal M7n**).

Damit gelangt der Fachmann in naheliegender Weise zum Verfahren des Anspruchs 1 nach Hilfsantrag, indem er das aus der Druckschrift D3 bekannte Verfahren zur automatischen Anpassung einer Steuersoftwarekomponente an eine Steuergeräteplattform gemäß seines Fachwissens an den jeweiligen Anwendungsfall entsprechend anpasst, ohne erfinderisch tätig werden zu müssen.

Der Anspruch 1 nach Hilfsantrag ist daher nicht patentfähig.

4) Mit den jeweils nicht patentfähigen Ansprüchen 1 nach Haupt- und Hilfsantrag sind auch die auf diese Ansprüche direkt oder indirekt rückbezogenen Unteransprüche nicht schutzfähig, da auf diese Ansprüche kein eigenständiges Patentbegehren gerichtet war (vgl. BGH, GRUR 2007, 862 Leitsatz – „Informationsübermittlungsverfahren II“).

5) Nachdem die jeweiligen Anspruchssätze nach Hauptantrag bzw. nach Hilfsantrag nicht patentfähig sind, war die Beschwerde zurückzuweisen.

Maile Schwarz Dr. Schwengelbeck Dr. Otten-Dünnweber

Hu