Internal distribution code:

(A) [-] Publication in OJ
(B) [-] To Chairmen and Members
(C) [-] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 3 May 2019

Case Number:	T 0900/14 - 3.5.04
Application Number:	06847852.8
Publication Number:	1964403
IPC:	H04N7/16
Language of the proceedings:	EN

Title of invention:
Programmable multimedia controller with programmable services

Applicant:
Savant Systems LLC

Headword:

Relevant legal provisions:
EPC 1973 Art. 56

Keyword:
Inventive step - (no)

Decisions cited:
T 2270/10

Catchword:

Case Number: T 0900/14 - 3.5.04

D E C I S I O N
# of Technical Board of Appeal 3.5.04
of 3 May 2019

Appellant: (Applicant)	Savant Systems LLC 32 Wianno Avenue Osterville, MA 02655 (US)
Representative:	Rupprecht, Kay Meissner Bolte Patentanwälte Rechtsanwälte Partnerschaft mbB Widenmayerstraße 47 80538 München (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 24 October 2013 refusing European patent application No. 06847852.8 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	C. Kunzelmann
Members:	R. Gerdes
	T. Karamanli

Summary of Facts and Submissions

I. The appeal is against the decision to refuse European patent application No. 06 847 852.8, published as international application WO 2007/075816 A2.

II. The examining division refused the patent application on the grounds that the subject-matter of the independent claims of the then main and auxiliary requests lacked inventive step in view of document:

D1: US 2004/0267914 A1

In its decision, the examining division also referred to document:

D2: EP 1 355 485 A1

III. The applicant/appellant filed notice of appeal against this decision. With its statement of grounds of appeal, it submitted claims of new main and auxiliary requests.

IV. The board issued a summons to oral proceedings together with a communication according to Article 15 (1) RPBA. In the communication, the board gave its preliminary opinion that the claims of the appellant's main and auxiliary requests did not involve an inventive step (Article 56 EPC 1973).

V. The appellant replied with letters dated 2 April 2019 and 30 April 2019. With the letter of 30 April 2019, the appellant submitted amended claims of a main request, replacing the claims of all previous requests.

VI. Oral proceedings were held before the board on 3 May 2019. During the oral proceedings, the appellant filed amended claims according to a new main and sole request and stated that the new main request replaced the previously filed main request. The appellant's final requests were that the decision under appeal be set aside and that a European patent be granted on the basis of the claims of the main and sole request filed at the oral proceedings of 3 May 2019.

VII. Claim 1 of the main request reads as follows:

"An integrated multimedia, entertainment, communications and control system having an environment for creating programmable services to be performed by said system comprising:

at least one component profile which includes information regarding a physical characteristic of a predetermined component which may be interfaced with said system, one or more commands that said component recognizes, and one or more functions that said component is capable of performing;

at least one zone configuration which includes information regarding an identification of a plurality of components that are present in one or more predetermined zones and one or more types of controllers available to the user in said predetermined zones;

a connection configuration which includes information regarding how components are interconnected;

at least one service rule which includes information regarding what component functions are needed to implement a predetermined programmable service and how a particular group of components providing said functions will interact to provide said predetermined service;

a configuration compiler, that is configured to receive as an input said at least one component profile, zone configuration, connection configuration and service rule, and which is configured to generate as an output a service implementation that describes in machine-readable markup language how to utilize a particular group of components to provide said predetermined service;

a workflow generator that is configured to generate, responsive to said service implementation, a graphical workflow that graphically represents said service implementation, and an executable program, said graphical workflow viewable by a user in an application and including a plurality of panels that represent discrete actions that are executed consecutively to provide said predetermined service, said executable program being executable to provide said predetermined service; and

a workflow browser/search engine/customization application configured to display, inspect and modify the graphical workflow, wherein, when a user edits said graphical workflow, said executable program is automatically changed in accordance with the user's edits."

VIII. In the decision under appeal (see point B.1.4), the examining division acknowledged that D1 did not disclose the feature of "at least one service rule which includes information regarding what component functions are needed to implement a predetermined programmable service and how a particular group of components providing said functions will interact to provide said predetermined service". In addition, claim 1 of the then main request differed from D1 in that the programming environment was a graphical programming environment, whereas D1 disclosed textual development tools.

The technical effect of the first distinguishing feature was that tasks could be performed by more than one device in the network. The corresponding technical problem was to develop applications that required the capabilities provided by more than one device in the network. Referring to D2, the examining division argued that the requirement of interoperability of different devices in universal plug and play (UPnP) systems was well known. Therefore, it would have been obvious to use services provided by more than one device to implement an application scenario.

Regarding the second distinguishing feature, the examining division argued that D1 disclosed textual development tools. It was also well known at the priority date of the application to use graphical development tools as an alternative to classical programming languages. Hence, the claimed subject-matter lacked an inventive step.

IX. The appellant's arguments, as far as relevant to the present decision, may be summarised as follows:

Apart from some differences in the configuration data that were input to the system of claim 1, D1 did not disclose the workflow generator and the "workflow browser/search engine/customization application" (henceforth only referred to as "workflow browser") of the present claimed invention. There was no graphical representation of actions occurring in a corresponding executable program, much less a graphical workflow with panels that represented discrete actions executed consecutively.

By modifying an executable program in response to edits to graphical workflows, the inventive technique allowed a user with little or no programming expertise to customise a service as desired since they could modify an executable program by changing a graphical workflow, rather than by writing lines of program code. Another effect was the immediate and direct modification of an executable program after receiving the user's input. To modify the executable, the programming did not have to restart at the beginning (input services in D1), but could be carried out using the workflow browser. Hence, an essential difference with respect to D1 was the short feedback loop using the workflow browser. Based on this, the service controller could execute the corresponding program without any delay allowing a quicker response to changes made by a user.

According to the invention, the graphical workflow and the executable program were linked such that editing the graphical workflow automatically caused a modification in the executable program. There was no link between a graphical workflow and an executable program in D1. In D1, no edits to a graphical workflow were received. Nor were any modifications made automatically in response to such edits to a graphical workflow.

The invention addressed the technical problem of configuring a group of disparate components to work together to provide a service, while allowing a user without detailed knowledge of software code to understand, and potentially modify, how this was performed.

Rather than having to understand software code, a user could simply look at a graphical workflow having panels representing discrete actions that were executed consecutively to understand what actions were occurring in the corresponding software code and potentially modify them. D1 taught away from the inventive technique. D1 envisioned operation according to "plug and play" principles, with the workings of underlying source code largely hidden from the user (see statement of grounds, points 4 to 16, and the letter dated 2 April 2019, point 3).

Reasons for the Decision

1. The appeal is admissible.

2. It is common ground that D1 may be considered the closest prior art for the subject-matter of claim 1.

2.1 D1 discloses a system for generating UPnP control points, which serve to control network devices to perform specified actions ("services"). The control points discover devices connected to the network and retrieve XML documents that describe the services provided by the devices as well as associated information (see paragraphs [0002], [0010], [0015] to [0017]).

To generate the control point software, a control point generator (engine) contains input services that receive control point target information such as device and/or service descriptions and/or platform specific information. The control point target information may be received via an interface such as a keyboard or a network from a programmer. Code services in the generator engine produce code for the desired control point from the control point target information. In a further step, compile services are used to compile the generated control point code (see paragraphs [0018] to [0032]).

2.2 Hence, D1 discloses an environment for creating programmable services having component profiles and service rules within the meaning of claim 1 (device and service descriptions). The code service of D1 corresponds to the configuration compiler of claim 1 except that the code services "generate custom source code and/or interfaces tailored to the enhanced control point" (see paragraph [0025]), whereas the configuration compiler of claim 1 generates "as an output a service implementation that describes in machine-readable markup language how to utilize a particular group of components to provide said predetermined service". Finally, the compile service produces an executable program similarly to the workflow generator of claim 1.

D1 also implicitly requires zone and connection configurations like in claim 1 to be able to establish the control point software.

2.3 Hence, D1 fails to disclose the following features of claim 1:

(a) at least one service rule which includes information regarding ... how a particular group of components providing said functions will interact to provide said predetermined service

(b) a configuration compiler, ... which is configured to generate as an output a service implementation that describes in machine-readable markup language how to utilise a particular group of components to provide said predetermined service

(c) a workflow generator that is configured to generate, responsive to said service implementation, a graphical workflow that graphically represents said service implementation, ... said graphical workflow viewable by a user in an application and including a plurality of panels that represent discrete actions that are executed consecutively to provide said predetermined service

(d) a workflow browser/search engine/customization application configured to display, inspect and modify the graphical workflow, wherein, when a user edits said graphical workflow, said executable program is automatically changed in accordance with the user's edits

2.4 Feature (a) relates to the input data of the claimed system, whereas feature (b) refers to an internal data representation in the system. Features (c) and (d) relate essentially to the user interface for the design process. Hence, the board considers features (a) and (b) to provide effects that are unrelated to each other and can be dealt with separately and independently of those associated with features (c) and (d).

2.5 Regarding feature (a), the board agrees with the decision under appeal that the corresponding technical effect is that tasks can be performed by more than one device in the network.

Distinguishing feature (b) refers to an alternative representation of the custom source code disclosed in D1 (see paragraphs [0028] and [0029]).

2.6 The board agrees with the decision under appeal that the technical problem solved by feature (a) may be considered as developing applications that require the capabilities provided by more than one device in the network (see point VIII above). Feature (b) solves the technical problem of providing an alternative representation of the custom source code.

2.7 The appellant argued that features (c) and (d) specified a graphical programming environment which allowed the immediate and direct modification of an executable program after receiving the user's input. To modify the executable, the programming did not have to restart at the beginning (input services in D1), but could be carried out using the workflow browser. Hence, an essential difference with respect to D1 was the short feedback loop in response to program modifications using the workflow browser.

2.8 Claim 1 does not contain features supporting this technical effect. Claim 1 specifies that the workflow generator generates the graphical workflow and an executable program which is automatically changed in accordance with the user's edits of the graphical workflow. Neither of these features nor the corresponding passages of the description (see page 16, lines 5 to 18) provide details regarding the generation of the executable in response to user edits. Therefore, changes to the graphical workflow could be converted into changes to the service definitions (see figure 4: 364), which would then be processed by the workflow generator to generate the modified executable. Such processing would be very similar to the initial generation of the graphical workflow and the corresponding executable and would not lead to a short feedback loop. The fact that figure 4, which is a block diagram showing the major software components involved in creating services in an illustrative embodiment, does not show an arrow pointing from the workflow browser (358) to the services definition (364), does not imply that claim 1 excludes such a generation of the modified executable. Hence, the effect of a quicker response to changes made by a user due to a short feedback loop in claim 1 is not credible.

2.9 The appellant also argued that the graphical programming environment specified by features (c) and (d) allowed a user with little or no programming expertise to customise a service as desired since they could modify an executable program by changing a graphical workflow rather than by writing lines of program code. Consequently, the claimed invention addressed the technical problem "of configuring a group of disparate components to work together to provide a service, while allowing a user without detailed knowledge of software code to understand, and potentially modify, how this is performed" (see point IX above).

2.10 It has been held in decision T 2270/10 that the question of whether a program is easy to read and maintain is largely a subjective one. However, the deciding board in T 2270/10 did not exclude that concrete details of a graphical user interface could simplify the use of a computer as a technical device and/or simplify the use of the computer when applied to solving a technical problem. It could thus contribute to solving a technical problem (see Reasons, points 8.2 and 9).

Claim 1 contains the features that the graphical workflow includes a plurality of panels representing discrete actions and that a user edits this graphical workflow. These features might be considered as concrete details of the graphical user interface. However, it remains questionable whether providing panels representing discrete actions and a graphical editor necessarily results in a better understanding of the software code and a simplified use of the computer.

2.11 There is no need to further consider this question, because, even if the board agrees with the appellant's formulation of the technical problem resulting from features (c) and (d), this question does not affect the outcome of the decision. Hence, in the following the board adopts the appellant's formulation of the technical problem corresponding to features (c) and (d) (see point 2.9 above).

2.12 Regarding feature (a), the board agrees with the decision under appeal that there was an apparent demand for services that needed to be performed by more than one device in the network. Hence, it would have been obvious to specify rules determining how a particular group of components interact to provide the service (see point VIII). With respect to feature (b), the alternative representation of the source code in markup language was well known to the skilled person, and it would have been obvious to use that representation.

2.13 Regarding features (c) and (d), the board also agrees with the decision under appeal that graphical development tools were well known in the art and commonly used as an alternative to classical programming languages (see point VIII above). Displaying a program in step-by-step execution was common practice in integrated development environments and debugging tools, which were often provided with graphical user interfaces employing panels representing discrete actions. Similarly, graphical editors (workflow browser/search engine/customisation application) were well known in the technical field of software development. Hence, depending on the circumstances, the skilled person would have considered to use such graphical development tools in the system of D1.

2.14 As a result, the subject-matter of claim 1 would have been obvious to the person skilled in the art in view of D1 and the common general knowledge and thus lacks inventive step (Article 56 EPC 1973).

Conclusion

3. It follows from the above that the appellant's request is not allowable and that therefore the appeal is to be dismissed.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

Beschwerdekammern des Europäischen Patentamts
Boards of Appeal of the European Patent Office
Chambres de recours de l'Office européen des brevets

K. Boelicke C. Kunzelmann

Decision electronically authenticated