

The opinion in support of the decision being entered today was not written for publication and is not binding precedent of the Board.

Paper No. 33

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE BOARD OF PATENT APPEALS  
AND INTERFERENCES

---

*Ex parte* JOHN S. ANDERSON and CHARLES D. BLEWETT

---

Appeal No. 1999-1604  
Application 08/741,419

---

ON BRIEF

---

Before HAIRSTON, JERRY SMITH and FLEMING, **Administrative Patent Judges**.

FLEMING, *Administrative Patent Judge*.

**DECISION ON APPEAL**

This is a decision on appeal from the final rejection of claims 1-25, all the claims pending in the present application.

The invention relates generally to a compiled object

contained in memory useable in a computer system that is implementing a windowing system. Specifically, the compiled objects are widgets (figure 2, item 201) representing a window displayed in a windowing system. The widget comprises means which when executed in the computer system at runtime set a resource of the widget to invocations of arbitrary functions of a set (specification, page 5, lines 9-30). The widget also comprises means which when executed in the computer in response to a change in the widget, employ the invocation to execute a function of the set of arbitrary functions specified in the invocation (specification, page 5, line 25 through page 6, line 12).

Independent claims 1 and 25 are reproduced as follows:

1. A widget contained in memory means useable in a computer system that is implementing a windowing system, the widget representing a window displayed in the windowing system and the widget comprising:

means which when executed in the computer system at runtime set a resource of the widget to one or more invocations of arbitrary functions of a set thereof; and

means which when executed in the computer system in response to a change in the widget employ the invocation to execute a function of the set of arbitrary functions specified in the invocation.

25. A drawing widget contained in memory means useable



Appeal No. 1999-1604  
Application 08/741,419

Answer<sup>2</sup>, for the respective details thereof.

**OPINION**

**A. Rejection of claims 1-24 under 35 U.S.C. § 103 as being unpatentable over Tonouchi when taken with Swanson**

We will not sustain the rejection of claims 1-24 under 35 U.S.C. § 103 as unpatentable over Tonouchi when taken with Swanson.

The Examiner has failed to set forth a *prima facie* case. It is the burden of the Examiner to establish why one having ordinary skill in the art would have been led to the claimed invention by the express teachings or suggestions found in the prior art, or by implications contained in such teachings or suggestions. *In re Sernaker*, 702 F.2d 989, 995, 217 USPQ 1, 6 (Fed. Cir. 1983).

Appellants assert<sup>3</sup> that neither Tonouchi nor Swanson

---

<sup>2</sup> Mailed November 23, 1998.

<sup>3</sup> Brief, pages 6, 10 and 11.

teaches or suggest a widget comprising means which, at runtime, set a resource of that widget to one or more invocations of arbitrary functions. Turning to Tonouchi, Appellants point out that this reference discloses<sup>4</sup> a system named Oak which supports construction of a widget by direct manipulation of graphics. The widget is programmed by Oak and thus does not modify itself. Appellants note that in Tonouchi<sup>5</sup> predefined widgets using a graphics editor are used to create the widget instead of writing program code.

In regard to the Examiner's contention that Tonouchi teaches<sup>6</sup> a means which sets the resources of the widget to one or more invocations of arbitrary functions, Appellants assert<sup>7</sup> that this is incorrect as this section of Tonouchi discusses the BVI model. This model, Appellants argue, provides for the behavior of the object which specifies the action of the widget, and this behavior is set at compile time, and

---

<sup>4</sup> Abstract, page 95, column 1, lines 10-14.

<sup>5</sup> Page 95, column 2, lines 26-36.

<sup>6</sup> Page 96, column 1, paragraph 2.

<sup>7</sup> Brief, pages 6-7.

therefore the function is not arbitrary.

Furthermore, Appellants argue<sup>8</sup> that the widgets of Tonouchi do not exist until the code generator creates them, and therefore the means disclosed in Tonouchi for setting resources form part of the code generator and not the widget as recited in claim 1. Appellants point to Tonouchi's disclosure<sup>9</sup> that the graphics editor of the Oak system permits designers to compose visual objects from subobjects through direct manipulation. The code generator then translates library information generated by the graphics editor into Interview widgets<sup>10</sup>. Therefore, Appellants conclude that the means for setting resources are part of the code generator and not the widget.

Appellants further argue<sup>11</sup> that Tonouchi does not teach or suggest modifying a widget that has already been compiled. Appellants assert that as widgets coded by the Oak system

---

<sup>8</sup> Brief, page 7.

<sup>9</sup> Page 97, column 1, lines 9-13 and 27-29.

<sup>10</sup> Page 98, column 1, lines 12-14.

<sup>11</sup> Brief, pages 7-8.

require compiling before execution, and because they have not been compiled they cannot be executed, and because they cannot be executed they cannot set their own resources. Thus, Appellants argue that Tonouchi does not teach the claimed widget comprising a means which when executed in a computer system at runtime set a resource.

As regards Swanson, Appellants admit<sup>12</sup> that Swanson teaches modifying the resources of a widget at runtime. However, Appellants assert that Swanson does not remedy the deficiencies of Tonouchi by teaching a widget that sets its own resources, and that the resources of the widget are set to one or more invocations of arbitrary functions.

Appellants assert that Swanson teaches a graphical resource editor that is useful for editing and changing the resource values of a completely separate application that is running concurrently with the graphical resource editor. Referring to figure 17 of Swanson, Appellants point out that the graphical resource editor is identified as a custom

---

<sup>12</sup> Brief, page 8.

application, while the application being edited is a client application. The graphical resource editor changes the resource values of widgets in the custom application.

In addition, Appellants admit<sup>13</sup> that Swanson teaches<sup>14</sup> applying resource edits to the client application while the client application is running, and therefore the widget of the client application is set while the widget is running. However, Appellants argue, that as shown in figure 17 of Swanson, the on-the-fly customization of the client application relies solely

upon X toolkit 2030, the Xt intrinsics Library 2040 and the Xlib 2050, none of which forms a part of a widget being modified in the client application.

Appellants further point out that in Swanson<sup>15</sup> the client application uses an Xlib function to get the custom data

---

<sup>13</sup> Brief, page 9.

<sup>14</sup> Column 16, line 48 through column 17, line 4.

<sup>15</sup> Column 21, line 55 through column 22, line 2.

property and then determines widgets in the X windows motif to match the resource. After a matching resource in a widget in the client application is found, the resource value is set in the matching widget using Xt Intrinsics function.

Therefore Swanson relies on intrinsic functions and Xlib functions to set a resource in the widget, and intrinsics and Xlib functions are not part of the widget. This, Appellants argue<sup>16</sup>, is contrary to the claimed requirement that the widget comprise a means that sets a resource to one or more invocations.

Appellants further argue<sup>17</sup> that the widgets of Swanson do not set a resource to an invocation to a set of functions as recited in the claims (Appellants' emphasis). Appellants point to the definition of invocations<sup>18</sup> being data items which identify functions and the arguments for the identified functions. In Swanson, Appellants assert, the function of the widgets for window 300 are set when the widget is

---

<sup>16</sup> Brief, page 10.

<sup>17</sup> Brief, page 10.

<sup>18</sup> Specification, page 8, lines 5-8.

compiled<sup>19</sup>. Thus, the widget does not set a resource of the widget to an invocation of a function as claimed.

The Examiner asserts<sup>20</sup> that Tonouchi teaches all the elements of claim 1, except that it does not specifically teach setting a resource of the widget at runtime (Examiner's emphasis). The Examiner then applies<sup>21</sup> Swanson<sup>22</sup> as "teaching a 'graphical resource editor for selectively modifying graphical resources in software applications' wherein 'customization of software application may be performed statically' or 'dynamically by applying resource edits on-the-fly to an application running concurrently with the graphical resource editor'."

The Examiner then finds that it would have been obvious to one of ordinary skill in the art at the time of the invention

---

<sup>19</sup> Column 14, line 34 through column 15, line 13.

<sup>20</sup> Answer, page 5.

<sup>21</sup> Answer, page 5.

<sup>22</sup> Abstract.

to include the graphical resource editor of Swanson in the invention of Tonouchi because it allows a user to dynamically set the resource of a widget at runtime.

In response to Appellants' argument that Swanson does not teach a widget that sets its own resources, the Examiner cites Swanson's statement<sup>23</sup> that the "graphical resource editor provides users with the capability to access application resources and selectively modify the resources to change one or more attributes of an application's graphical user interface." Thus, the Examiner asserts that Swanson teaches the use of a graphical resource editor, which represents the widget, to modify the resources of the widget, wherein the widget represents a window, being the application's graphical interface.

In response to Appellants' argument that Swanson does not teach a widget that sets its own resources to one or more invocations of arbitrary functions, the Examiner cites<sup>24</sup>

---

<sup>23</sup> Column 6, lines 54-57.

<sup>24</sup> Answer, page 9.

Swanson's teachings<sup>25</sup> stating "that 'widgets utilize application resources to specify window characteristics' and that an 'application resource can be any user-customizable parameter that affects an application's behavior or appearance'." In addition, the Examiner cites Appellants' definition<sup>26</sup> that an invocation is a "data item which represents a function and actual arguments for the function." Thus, the Examiner finds that Swanson teaches changing invocations to functions by dynamically modifying resource values.

Finally, regarding Tonouchi, the Examiner admits<sup>27</sup> that Tonouchi does not teach setting a resource at runtime, but asserts that in Tonouchi<sup>28</sup> the widget set its own resource to an invocation of a function where the callback function to be invoked is entered.

After careful consideration of the arguments presented,

---

<sup>25</sup> Column 6, lines 39-45.

<sup>26</sup> Specification, page 8, lines 5-8.

<sup>27</sup> Answer, page 9.

<sup>28</sup> Page 97, column 2.

we find that Tonouchi does not teach a means which sets the resources of the widget to one or more invocations of arbitrary functions. The BVI model of Tonouchi provides<sup>29</sup> for the behavior of the object which specifies the action of the widget, and this behavior is set at compile time, and therefore the function is not arbitrary.

Furthermore, we find that as the widgets of Tonouchi do not exist until the code generator creates them<sup>30</sup>, the means disclosed in Tonouchi for setting resources form part of the code generator and not the widget. As the graphics editor of the Oak system permits designers to compose visual objects from subobjects through direct manipulation, and the code generator then translates library information generated by the graphics editor into Interview widgets as disclosed by Tonouchi<sup>31</sup>, therefore the means for setting resources are part of the code generator and not the widget.

The Examiner's argument that Tonouchi teaches the widget

---

<sup>29</sup> Page 96, column 1.

<sup>30</sup> Page 98, column 1, lines 12-14.

<sup>31</sup> Page 98, column 1, lines 12-14.

as setting its own resource to an invocation to a function where the call back function to be invoked is entered, is not cogent. Tonouchi provides<sup>32</sup> an editor which requires a designer to enter the name of the callback function for a second entity, and the editor is used to program the widget of the second entity. Thus, the graphics editor and code generator program the code for the widget, and the widget does not comprise means for setting a resource of that same widget as claimed.

In addition, we find that Swanson does not teach a widget that sets its own resources. We agree with Appellants that Swanson teaches a graphical resource editor that is useful for editing and changing the resource values of a completely separate application that is running concurrently with the graphical resource editor. In Swanson<sup>33</sup> the graphical resource editor is custom application 2010, and the application being edited is client application 2020. Thus, this graphical resource editor changes the resource values of

---

<sup>32</sup> Page 97.

<sup>33</sup> Figure 17.

widgets in the custom application. In addition, we find that Swanson teaches that the on-the-fly customization of the client application relies solely upon the X toolkit, the Xt Intrinsic Library and

the Xlib, none of which forms a part of a widget being modified in the client application<sup>34</sup>.

Furthermore, Swanson teaches<sup>35</sup> a client application using an Xlib function to get the custom data property 3000 and then determining widgets in the X windows motif 2030 to match the resource. After a matching resource in a widget in the client application is found<sup>36</sup>, the resource value is set in the matching widget using Xt Intrinsic function<sup>37</sup>. Therefore Swanson relies on intrinsic functions and Xlib functions to set a resource in the widget, and intrinsics and Xlib functions are not part of the widget. Thus, we find that

---

<sup>34</sup> Column 20, lines 10-23; figure 17.

<sup>35</sup> Column 21, line 55 through column 22, line 2.

<sup>36</sup> Column 23, lines 61-62.

<sup>37</sup> Column 24, lines 32-34.

Appeal No. 1999-1604  
Application 08/741,419

Swanson does not teach the claimed limitation that the widget comprises a means that sets a resource to one or more invocations.

The Examiner's argument that Swanson teaches a graphical resource editor which represents a widget, to modify the resources of the widget is not cogent. The graphical resource

editor modifies the resources of a separate application as set forth above. The Xlib and Intrinsic functions are independent of any widget to set resources in a widget. Thus, Swanson does not teach or suggest the graphical resource editor setting resources of its own widgets to invocations as claimed.

The Examiner has failed to provide any teaching or suggestion from the prior art to provide a widget setting its own resource at runtime to one or more invocations of arbitrary

functions. Therefore, the rejection of claims 1-24 under 35 U.S.C. § 103 is reversed.

**B. Rejection of claim 25 under 35 U.S.C. § 102(e) as**

**anticipated by Swanson**

We will not sustain the rejection of claim 25 under 35 U.S.C. § 102(e) as anticipated by Swanson.

It is axiomatic that anticipation of a claim under 35 U.S.C. § 102 can be found only if the prior art reference discloses every element of the claim. See *In re King*, 801 F.2d 1324, 1326, 231 USPQ 136, 138 (Fed. Cir. 1986) and *Lindemann Maschinenfabrik GMBH v. American Hoist & Derrick Co.*, 730 F.2d 1452, 1458, 221 USPQ 481, 485 (Fed. Cir. 1984). "Anticipation is established only when a single prior art reference discloses, expressly or under principles of inherency, each and every element of a claimed invention." *RCA Corp. v. Applied Digital Data Systems, Inc.*, 730 F.2d 1440, 1444, 221 USPQ 385, 388 (Fed. Cir. 1984), cert. dismissed, 468 U.S. 1228 (1984), citing *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 772, 218 USPQ 781, 789 (Fed. Cir. 1983).

Appellants submit<sup>38</sup> that Swanson does not teach either a drawing widget that includes a set values method, or a set

---

<sup>38</sup> Brief, page 13.

values method that sets a resource of the widget to one or more invocations of functions. Appellants' point to Swanson's teaching<sup>39</sup> of a set values function that is an intrinsic function 2040 and not part of a widget 2030, and that there is no mention in Swanson of setting a resource to one or more invocations of functions.

The Examiner points first to Swanson's teaching<sup>40</sup> of setting a resource and editing resource values, then to Swanson's teaching<sup>41</sup> of the Xlib 2050, and then to Swanson's teaching<sup>42</sup> that widgets utilizing the resources are modified.

We find that these citations from Swanson do not disclose

---

<sup>39</sup> Column 24, lines 32-44.

<sup>40</sup> Column 21, lines 41-54.

<sup>41</sup> Column 20, line 23; figure 17.

<sup>42</sup> Column 21, line 55 through column 22, line 2.

the claimed<sup>43</sup> drawing widget which includes a set values method that sets a resource of the widget to one or more invocations of functions. As we have found above, as shown in figure 17 of Swanson, the on-the-fly customization of the client application relies solely upon X toolkit 2030, the Xt intrinsics Library 2040 and the Xlib 2050, none of which forms a part of a widget being modified in the client application<sup>44</sup>. The set values function is intrinsics function 2040 and not part of widget 2030<sup>45</sup>.

Therefore, the rejection of claim 25 under 35 U.S.C. § 102(e) as anticipated by Swanson is reversed.

We have not sustained any of the rejections of claims 1-25. Accordingly, the Examiner's decision is reversed.

**REVERSED**

---

<sup>43</sup> Claim 25, lines 5-7.

<sup>44</sup> Column 20, lines 10-23.

<sup>45</sup> Column 24, lines 32-44.

Appeal No. 1999-1604  
Application 08/741,419

	KENNETH W. HAIRSTON	)	
	Administrative Patent Judge	)	
		)	
		)	
		)	BOARD OF
PATENT		)	
	JERRY SMITH	)	APPEALS AND
	Administrative Patent Judge	)	
INTERFERENCES		)	
		)	
		)	
	MICHAEL R. FLEMING	)	
	Administrative Patent Judge	)	

MRF:pgg

Appeal No. 1999-1604  
Application 08/741,419

Docket Admin. Rm 3C-572  
Lucent Technologies, Inc.  
600 Mountain Ave.  
P.O. Box 636  
Murray Hill, NJ 07974-0636