

# Application of UI Design Tool “NINA” to In-Car Infotainment Systems

Authors: Akira Toyo-oka\* and Hiroki Konaka\*

## 1. Introduction

The cost of UI (User Interface) software development continues to rise since embedded devices become more functional and UI designs become richer. We have been developing a UI design tool “NINA (Navigator for Interface of Application)” to improve the efficiency of UI development for such embedded devices, to overcome the problem of cost. This report discusses the application of NINA to the UI development for in-car infotainment systems and the results of evaluating its effectiveness.

## 2. UI Design Tool “NINA”

“NINA” is a UI design tool for embedded devices. Since embedded devices have smaller screens than those of PCs, they are generally operated by switching scenes in accordance with restricted operating procedures. NINA is based on the SCO (State Chart Object) concept which is suitable for UI modeling.

An SCO is a UI component which consists of more than one state, each corresponding to a scene, and transitions between states. In each SCO scene, in addition to basic UI components such as buttons and labels, other SCOs can be laid out as UI components. With the hierarchical nature of SCOs, all of the custom UI components and the UI of the applications can be designed in the same framework and used in combination. In addition, the reusability and maintainability of UI design are improved.

Figure 1 shows an example of hierarchical SCOs. SCO1 is placed as a custom UI component in the music playing scene of SCO2. If SCO1 is modified, the music playing scene of SCO2 which uses SCO1 reflects that particular modification.

## 3. Structure of NINA

NINA consists of several functional modules used on PC and a software module for the target device. Figure 2 shows the structure of NINA and the flow of UI development.

The chart editor is a tool to design the transitions between the UI scenes into a state diagram. The procedures in response to key events during device operation and other events generated in the device can be defined as event handlers.

The layout editor is a tool to prepare the layout of UI components in the scene for each SCO state. The scene layout is designed using basic UI components such as labels, buttons, images, and container panels, and other SCOs in combination.

The simulator is a tool to check the behavior of the UI, which has been designed with the chart editor and the layout editor, on a PC. The inspection is faster and more efficient on a PC than on the target device.

The code generator generates the source code in C++ language for the target device from the UI design data. The code thus generated neither has platform-dependency nor requires modification by the

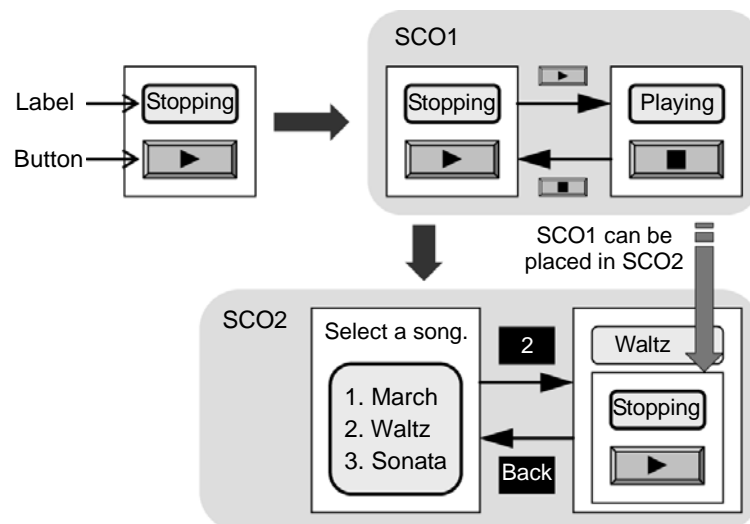


Fig. 1 Hierarchical use of SCOs

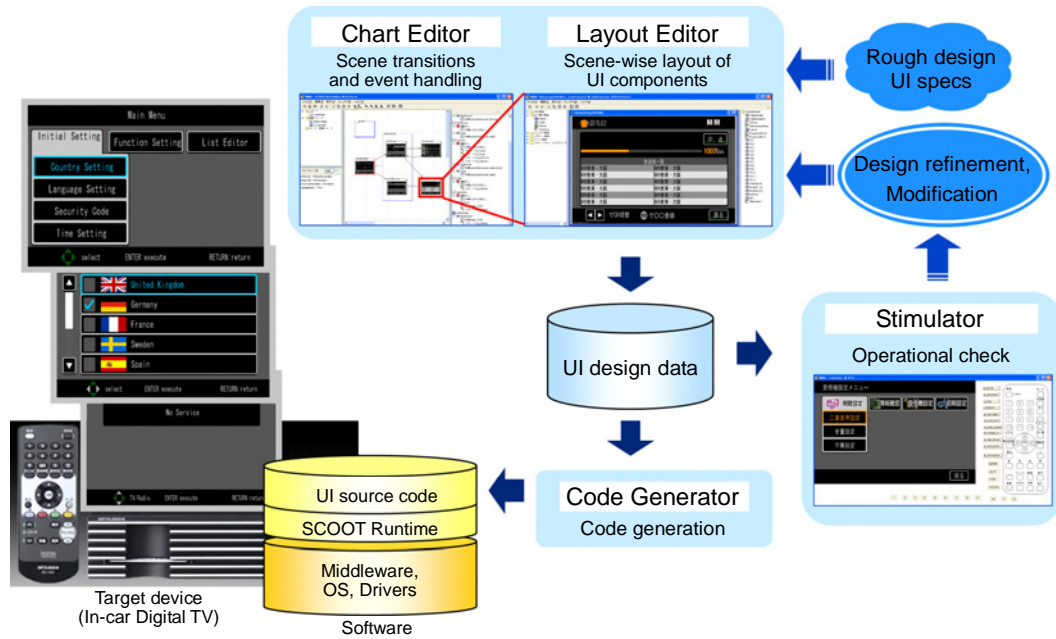







Fig. 2 The structure of NINA and the flow of UI development

Table 1 Modification tasks of UI design data

No.	Task	Scene image	Description of task
1	Change of hierarchical structure of the menu		<p>The hierarchical structure of the menu, in which a sub-menu item is selected from the tab after selecting the category by the tab, is changed.</p> <ul style="list-style-type: none"> <li>- Development of menu based on old specification has been completed. The hierarchical structure of the menu in accordance with the new specification is changed.</li> <li>- The number of categories (4) remains unchanged. Each category name is changed.</li> <li>- The number of sub-menu items in each category is changed.</li> <li>- The total number of sub-menu items remains the same (items are exchanged among the categories).</li> </ul>
2	Design of a new setting menu		<p>A new setting menu which has the function of selecting one option out of two to four items is designed.</p> <ul style="list-style-type: none"> <li>- A similar setting menu has already been developed. Development of custom UI components such as a button with a check box has been completed.</li> </ul>
3	Change of appearance of custom UI components	<p>Before change</p>  <p>After change</p> 	<p>The time display method and the layout of the component to display program broadcasting time are changed.</p> <ul style="list-style-type: none"> <li>- Time display is changed from AM/PM display to 24-hour type display.</li> <li>- Date display is added.</li> <li>- The layout (display position and background) is changed.</li> </ul>
4	Change of appearance and operating procedure of the TV program guide		<p>A new appearance design is prepared for the TV program guide. In addition, the operating procedure is also changed to add new functionality.</p> <ul style="list-style-type: none"> <li>- Date display and a selection function are added.</li> <li>- A function that lists the currently broadcast programs on all channels is added.</li> </ul>

programmer.

The SCOOT (SCO Oriented Technology) Runtime is a software module written in C++ language for running the source code generated by the code generator on the target device.

#### 4. Application to In-car Infotainment Systems

We used NINA to develop the UI for in-car infotainment systems, which are digital TV prototypes for European markets in compliance with DVB-T (Digital Video Broadcasting-Terrestrial) and domestic digital TV

products in compliance with ISDB-T (Integrated Services Digital Broadcasting-Terrestrial). The domestic digital TV products are available in the market as TU-100D (launched in December 2005) and TU-200D (launched in June 2007).

The conventional UI development was based on hand-coding by programmers using a stock UI toolkit. The problems of the conventional method and the effectiveness of NINA-based development are discussed below.

- (1) In the conventional method, generation and layout of UI components, event handling procedures, and conversion of display data are combined in the same program. On the other hand, in NINA, those functions are divided by the tools, thus making the UI design more readable. Besides, the points of modification and the extent of the impact are made clear when changing the UI specifications.
- (2) In the conventional method, operational checks of the UI were conducted on the target device, incurring overhead such as recompilation and downloading. With NINA, operational checks can be performed using the simulator, thus eliminating such overhead.
- (3) In the conventional method, routine processes commonly used in UI program were hand-coded. With NINA, routine processes are already provided by the SCOOT Runtime, thus reducing development cost and preventing bugs.

### 5. Evaluation

We evaluated the conventional method and NINA by conducting the tasks shown in Table 1 to assess the development cost.

Table 2 compares the development cost between the conventional method and NINA. One person was in charge of the development of the conventional method

**Table 2 Comparison of UI modification costs**

No.	Task	Conventional method	NINA	Man-hour rate
1	Change of hierarchical structure of the menu	16h	4h	25%
2	Preparation of a new setting menu	16h	8h	50%
3	Change of appearance of custom UI components	6h	1h	17%
4	Change of appearance and operating procedure of the TV program guide	120h	48h	40%

and NINA respectively, and each person was skilled in the respective method. The development cost is shown in man-hours. The man-hour rate shows the value for NINA against 100 for the conventional method. For any type of task, development efficiency with NINA is higher than with the conventional method. The conventional method required more time than NINA for resetting UI component coordinates when changing the scene layout, modification of transition processing when changing scene transition, and functional checks on the target device. With NINA, each task can be completed while checking the results of modifications by using the layout editor, chart editor and simulator, resulting in a clear difference over the conventional method.

### 6. Conclusion

We used the UI design tool "NINA" to develop UI for in-car infotainment systems. We confirmed that the cost of developing UI software for specification changes can be reduced to around 17% to 50%. We will continue to develop the technology described in this report for application to other derivational products in the future.