Research on Data Model Application of Localized Display Software

Xiaoran Zhu, Yong Zhou, Fei Zhao, Juan Zhang and Yujie Yao
China Satellite Maritime Tracking and Control Department, Jiangyin 214431, China

Abstract. This paper combines the domestic display software research and development work, the display process and main technologies are described, the model design and verification under kirin operating system was carried out, the standardization of data representation model and the generalized control display data interface are studied, and the data driven mechanism and control model has carried on the preliminary study and practical application, the rationality of the test of verify the feasibility of technology and framework. The application results show that the design of the new software is feasible and efficient, which reduces the difficulty of control development, improves the flexibility and scope of application of the control, greatly reduces the workload of software maintenance, and plays a good role in the follow-up research and development of the software.

1. Introduction
The Kirin operating system is an operating system with independent intellectual property rights in China and is a system platform for next-generation task software. In the existing display software can only run under the Microsoft platform, there are shortcomings such as unfriendly interface and complicated scene configuration.

Display software is an important tool to assist customers and units in making decisions and must be able to meet strong real-time requirements. At the same time, high-intensity, high-density, and difficult task situations require display software to be easy to configure and easy to maintain. The display software must be able to accept a variety of data sources, including real-time data of network multicast, key data obtained by compensation, and scene recovery data stored locally. The display software must also be able to output different types of data in a variety of display formats to LED displays, large screen projections, display microcomputers and other devices.

2. Display software main processes and technologies

2.1. Display process
In the task flow, after the task display software system receives the network data packet, the data processing software obtains the task parameters in real time and sends the task parameters to the data display software. The data display software projects the parameters into the display in different forms through classification, screening and typesetting. The microcomputer switches the display interface in real time by the operator according to the decision requirements. Based on the data-driven display refresh mechanism, the page refresh is paused when the network is idle, and the data is updated.
immediately when the data arrives, thereby effectively improving data real-time and reducing system load.

![Display system flow](image)

Figure 1. Display system flow

2.2. Main technique
Combining localization transformation and display software R&D requirements, the research and verification of display framework technology under Kirin operating system is carried out, including data-based unified technology, generalized and easy-to-customize display control technology, data real-time driving technology and dynamic hierarchical scene construction technology. Etc., provide reference for the localization of the task display software.

2.2.1. Data end unified technology. The problem to be solved by the data-side unification technology is to unify various data-side data (network real-time data, real-time solution data, theoretical data, historical data, compensation key data, etc.) into a consistent data model, so that the front end is displayed. You can use a unified interface to access data without paying attention to the characteristics of the data side and the differences in data. Firstly, we need to study the representation model of the displayable task data, and then design a flexible and efficient data source module, which can access multiple data terminals according to the task requirements, and provide displayable data for the research of the key technology of the project--real-time data driver module.

2.2.2. Generalized and easy to customize display control technology. Different types of data need to adopt different forms of expression (text, picture, instrument dial, trajectory, table, etc.), different devices (LED screen, large screen projection, and display microcomputer) have different requirements for display, and new tasks are new display requirements will be presented. This requires that the display control must be easy to develop, easy to customize, easy to integrate, and put a high standard for the generalization of the control. At the same time, different types of controls have different requirements for data access. For example, the instrument dial type control only needs the latest real-time data, and the track type control needs historical accumulated data. The control data interface must be able to adapt to all these display requirements. Therefore, based on Qt Quick's generalized control specification, it is easy to develop code to implement new controls using standard specifications, and can be customized and deployed with unified tools. At the same time, the standard data interface of the control is studied to receive real-time driven task data. Different display requirements are refreshed.

2.2.3. Data real-time drive technology. Data real-time drive technology is the key technology of software. According to the characteristics of high-real-time performance of various types of task data, the software researches the Qt-based data real-time driving technology under the Kirin operating system, and drives the data source data to drive the display control refresh in real time, improving the real-time
data and reducing the refresh load. The software design implements the dynamic binding mechanism of the data source and the display control, and can dynamically bind the new display control and the data source according to the requirements when the task is running, or cancel the binding of the display control and the data source.

2.2.4. *Hierarchical scene construction technology.* Combining and viewing real-time data in mission’s helps customers make better decisions. The displayable area of the display device is limited, and this problem can be solved by layering the display controls. Software research shows the hierarchical technology of controls that can be used to build hierarchical scenarios.

3. **Data model design ideas**

Summarizing the successful experience and shortcomings of the display software before, through combing and analyzing the task display data and display requirements, it is proposed to separate the data representation from the data display. The advantage of separation is that you can use a single control to display a variety of data according to the characteristics of the platform task, or you can display a kind of data in different forms through a variety of controls.

![Data representation is separated from data display](image)

*Figure 2. Data representation is separated from data display*

The specific implementation method is to design a standardized data representation model first, then design a generalized control display data interface, and finally bind the two.

3.1. **Standardized data representation model**

The task raw data format (PDXP) is complex and unintuitive. It must pass the dimension conversion, data accumulation, three-question two, virtual channel, decoding and other mechanisms to obtain intuitive and understandable task result data. This work is done by the data processing software in the display software system, and the data display software receives the result data sent by the data processing software to drive the display refresh, and the two represent the result data through a simple and unified data model, using a standard protocol. To encode and decode task parameters.

In a standardized data representation model, different solution devices (data processing software itself can be used as a special type of data source) are different data sources. The protocol used by each data source and the class of task data that can be sent are determined. The protocol used by each data source is unique and the task data sent is multi-category. By extracting and summarizing the protocol header field and the task data field, a CRP (Compact Result Protocol) is obtained.

The definition of the CRP protocol is shown in Figure 3:
The fields of the protocol packet represent: time stamp (describe data resolution time), data source code, result identifier (describes source packet status), protocol field, data item code, and data field mask (determined by mask) Subsequent available data fields), data field values and source fields (source packets). The data source encoding uniqueness determines the source unit device of the original data, and the data item encoding uniqueness determines the data category. The protocol field contains the header information of the protocol used by the device to send data. The possible protocol fields are protocol version, task code MID, target code BID, etc.; the data field is an understandable task parameter solved by the data processing software, and possible data. Fields have time, longitude, and so on. Each field has a certain encoding type (single-byte signed integer, string, date and time, source code, etc.) and units (time, length, speed, etc.).

The advantage of the CRP protocol is that the protocol design has nothing to do with specific data. By defining a CRP protocol text, it can represent the data set of a task. The CRP protocol text is an XML file that stipulates this protocol. It stipulates which data sources are available for this task, which data will be sent separately, and the format, unit, and theoretical value of the data.

If there is no standardized data representation model, it is extremely difficult for the display control to obtain specific data. It must be able to understand all the data information of this task and correctly decode and calculate it. The control code implementation and specific data binding will lead to software maintenance and the disaster of testing.

After using the standardized data representation model, the display control can separate the design implementation from the specific category data, and only need to consider how to refresh the display through some kind of data (analog number, floating point number, date time, string, etc.). Maximize the scope of display controls. After implementing the data decoding engine, by specifying the data source encoding SCode and the data item encoding DCode, the display terminal can pay attention to any type of data packet that satisfies the requirements, and perform a consistent response when the corresponding data packet arrives.

### 3.2. Generalized control display data interface

The display requirements of different tasks are different. The display control forms are colorful, and the data to be displayed also has different characteristics. In order to be able to refresh the display, the display software must temporarily store the task data. For example, the latitude and longitude data can display the digital value in the table control as needed, just overwrite the old value with the old value; and it can display the historical track in the track control, which requires the software to receive all the received The data is stored. This puts different requirements on the temporary storage and use of different task data. If all data is stored regardless of category and timeliness, the data storage capacity of long-term tasks will be huge, and the system load will increase with the task time and eventually reach the level of unloading, which is unacceptable.
The existing display software solves this problem by storing data in a centralized manner and separately processing different types of data according to data characteristics. The first problem is the complex temporary data model, which needs to customize the specific storage model (tablespace, matrix space or mapping space, etc.) according to different types of task display requirements and characteristics of each type of data (timeliness, data dimensions, etc.). And the data access interface (take the latest data, take historical data, take all the data, etc.), the workload of software maintenance is greatly increased; the second problem is that if there is a new display requirement, the data storage model must be modified. What followed was a significant increase in software maintenance test workload. Therefore, the solution adopted is to transfer the data storage requirements to the display control, and display the data interface through the generalized control to drive the control to update the data. The display requirements of the display controls are determined so that the requirements for data-assisted storage are also determined. If the control display requirement is to display the latest data value, only one variable is needed for auxiliary storage inside the control implementation; if the control display requirement is to display all data values, a list space is opened inside the control for auxiliary storage; if the control display requirement is based on a specific The code displays different data, and the mapping space is opened inside the control for auxiliary storage. The control only pays attention to whether the input data is of the correct type, does not need to understand the data category, characteristics and data meaning, reduces the development difficulty of the control, and improves the flexibility and application scope of the control.

3.3. Data filtering and binding mechanism

After defining a standardized data representation model and a generic control display data interface, a mechanism is needed to connect the two to accurately transfer data values from the data source to the target control.

The task of data binding is to connect the specific category data of the data source with the specific data unit of the target control, and the data unit receives the category data for data update response. Data binding also defines the connection between data fields and data elements.

A filter condition is a combination of logical relationships by a set of basic filter conditions. Each basic filter defines a data field, comparison relationship, and baseline value. The data field is determined by >, ≥, =, ≠, <, ≤. The combination of basic filter conditions is ∧ (and relationship), ∨ (or relationship), ! (Non-relationship), ⊕ (exclusive or relationship), etc. It should be noted that one data unit can be associated with multiple data bindings at the same time. Any data binding can drive the data unit to update the response.

4. Probe into Data Driven and Control Model

4.1. Data-driven real-time display refresh mechanism

The main content of the display software is to receive multicast real-time data from the network, and theoretical data such as theoretical trajectory and theoretical time need to be imported at the beginning of the task. Once the software is restarted, critical data must be obtained from the appropriate route for data compensation. It is also necessary to be able to load historical data during task playback.

The software has designed a data source engine that can flexibly connect multiple data ends to the system. The design of the engine is shown in Figure 4:
The data source engine defines a data converter for various types of data ends. The input of the data converter is the data of the data end (network data packet/theoretical data file, etc.), and the output is a CRP data stream (a data packet encoded by the CRP protocol). The engine uses a unified buffer to buffer the CRP data stream output by each converter. The parser takes the CRP data stream from the buffer, parses and generates a CRP object (i.e., \{data field name: data field value\} mapping table), and passes it to the data. The distribution center, which performs processing (local storage, driver control updates, etc.) according to the registered data processor.

The data source engine is very flexible: if a new data terminal is connected to the system, simply define a data converter for the data side. The data on the new data side will be converted into CRP data stream and processed by the system. If there is new data application requirement, only the new data processor needs to be defined for the data distribution center, and all data on the data end will flow through the data distribution center to reach the data. The processor performs the appropriate processing.

4.2. Hierarchical model of controls
The generalized control display data interface solves the problem of data source for the control, and the hierarchical model solves the problem of unified customization and assembly for the control.

For the purpose of joint viewing and real-time data comparison, multiple related control displays should be organically combined and placed in the main control container. In order to adapt to the display requirements of different resolutions of multi-output devices, the display area of the attached control is associated with the main control. The transparency of the attached controls on the main control is
different, and the depth is also before and after, avoiding the side effects of shadowing. You can dynamically adjust the visibility to shield the display of an attached control.

![Hierarchical model of QML controls](image)

**Figure 5.** Hierarchical model of QML controls

By defining the hierarchical model of the control, the entire display interface can be regarded as a control tree, the main interface is the root of the tree, the lower layer is the display page group and the status bar control, and the lower layer of the page group control is each display page. Inside the display page are various display controls, and the display control can also have an auxiliary display control inside.

With a generic control display data interface and attribute group interface, we can customize any control with a unified customizer. With the hierarchical tree control model, we can use the XML file to describe the display scene, which is convenient for displaying the data packet generation and setting the scene.

5. Conclusion

This paper combines the research and development of localized display software, and carries out the research of data-driven generalized display framework technology under the Kirin operating system. It mainly studies the data-side unified technology and data real-time driving technology, and conducts preliminary research on hierarchical scene construction technology. It verifies the feasibility of the technology and the rationality of the framework, and has important reference value for the subsequent engineering development of display software.

References

[1] An Xiaohui. Qt on Android core programming. Electronic Industry Press 2015.01: 33-36.
[2] Stephen Prata. C++ Primer Plus (Fifth Edition) Chinese version. People's Posts and Telecommunications Press. 2005.09.: 192-193
[3] Mark Summerfield. Qt Advanced Programming. Electronics Industry Press. 2011.04: 76-79.
[4] Jasmin Blanchette. C++ GUI Qt 4 Programming. Electronics Industry Press. 2013.05: 102-108.
[5] Yin Renkun. Data Structure. Tsinghua University Press. 2007.06: 79-80.
[6] Wang Weibo et al. Qt 5.9 C++ Development Guide. Mechanical Industry Press. 2007.05: 43-60.