Design and Realization of Universal Radar Signal Processing Software

Xiang Li\textsuperscript{1, a}, Jinsong Du\textsuperscript{1, b}, Yiyang Liu\textsuperscript{1, c} and Xin Bi\textsuperscript{1, d}

\textsuperscript{1}Key Laboratory of Liaoning Province on Radar System and Application, Shenyang Institute of Automation
Chinese Academy of Sciences, China

\textsuperscript{a}lixiang@sia.cn, \textsuperscript{b}jsdu@sia.cn, \textsuperscript{c}sialiuyiyang@sia.cn, \textsuperscript{d}bixin@sia.cn

\textbf{Keywords:} Radar Signal Processing; Graphical; Generalization; Modularization.

\textbf{Abstract.} As parallel signal processing tasks are more and more widely used, a universal, efficient and flexible radar signal processing software is urgently needed in the field of radar signal processing. Traditional development of parallel signal processing needs a large number of specialized programs, and must couple with the hardware platform. Therefore, the cost of development and promotion is quite large. The universal radar signal processing software designed in this paper is an integrated development environment to achieve fully graphical, generalized, and modularized development. The software is able to automatically generate codes. This function replaces traditional developing method that needs designers write a large number of programs. The above function significantly shortens the developing cycle and improves the developing efficiency. Besides, the software is of powerful debugging features. Furthermore, the software has excellent generalization and expansibility for hardware platform.

\textbf{Introduction}

During "Eleventh Five-Year" period, universal radar signal processing software had already been designed, and formed a preliminary version. But that version had many shortcomings, such as unreasonable structure, low performance, strong platform-dependent, weak object-oriented concept, and so on. Based on that version, the software developed in this paper designed brand-new architecture and used WPF technology. The name of this software is Radar Laboratory, referred to as RarLab.

Rarlab software is designed according to abstract factory pattern, façade pattern and other patterns in software architecture [1], which makes the architecture more reasonable. The software can extend the functional module library, and establish signal processing flowcharts and hardware topology diagrams based on graphical and modular approach. The software is able to be applied for Multi-DSP systems, power PC and other different hardware platforms, and to automatically generate the appropriate codes that are executable for corresponding hardware platforms. Furthermore, the software is of powerful debugging features. Therefore, the software really makes generalization, expansibility and high efficiency of development come true.

\textbf{Software Architecture}

Rarlab software is a powerful integrated development environment. The developing procedure is shown as below. Users drag and drop functional modules in radar functional module library to build radar signal processing flowchart, and assign each functional module in the flowchart to every hardware unit on the underlying hardware platform, which is called mapping. The software automatically generates codes appropriate to corresponding hardware platform, based on signal processing flowchart, the underlying hardware topology and the mapping program. During the process of code execution, the software provides real-time monitoring and debugging, displays results detected by radars, and keeps the track of objects for playback. The software architecture is shown in Fig. 1.
Design and Realization of Each Part in the Software

RarLab software includes the following main functions: management of module library, modeling of signal processing flow chart, modeling of hardware topology structure, mapping, and automatic generation of codes. Each function will be respectively described in this section.

Management of Module Library. In the RarLab, there are two kinds of module libraries: functional module library and hardware module library. Functional module library is used to build the signal processing flowchart, and the hardware module library is used to build the topology map. The design and realization of functional module library is mainly introduced in this paper. Because the hardware module library is similar to functional module library, it is not introduced repeatedly.

Compared with former version of radar signal processing software, RarLab carries out the expansibility of functional module library. The software provides the usual radar functional module library. Developers of signal processing system can also add self-defining modules to the library. The above functions make sure the expansibility of RarLab. When creating new functional modules, the users only need to fill in the properties of each functional module, according to the guidance of creating new modules. Functional module includes the following properties: module names, module categories, information of input ports, information of output ports, and information of parameters. The software stores information in the module library file in the form of XML. Specific XML hierarchy structure will be introduced in next section, along with the signal processing flow chart.

RarLab also adds editing modules, removing modules, displaying according to categories, and fuzzy query. That displaying according to categories is that software divides all the modules according to functions, and displays the modules in tree-like structure to make the users convenient to seek the modules. The fuzzy query is that users only need to partly input the symbols of module names to find out all the modules whose names contain the symbols input. This function increases the speed of seeking modules.

Modeling of Signal Processing FlowChart. Building signal processing flowchart is the core operation of RarLab. Users can build signal processing flowchart by graphical and modularized method to complete the design of radar signal processing system. An example of building signal processing flowchart is shown in Fig. 2.

The procedure of graphically modeling signal processing flowchart is shown as below:
First, drag and drop modules. From the functional module library of interface’s top shown in Fig. 2, users drag and drop the functional modules to the right signal processing flowchart, and release the cursor mouse, then the functional modules are changed into yellow rectangles.
Second, connect lines. Red dot on each functional module indicates output port, while the blue dot indicates input port. Connect lines between input ports and output ports of modules, then the relationship between two modules are setup.
Third, common edit commands. Each functional module and connector in flowchart supports operations such as copy, paste, cut, and delete.
Fourth, configure the views. The software provides many operations for flow chart processing, such as zooming in, zooming out, restoring to the best view and so on.
Fifth, undo and regenerate. All the above operations support undoing and regeneration operations. The functional modules referred in last section have a number of properties, and lines between modules also have many properties. These properties are used to describe information in signal processing. Most integrated development environments provide the function that editing the properties of the controls. Similarly, RarLab software had the same function. Property edit box is in the lower left of the interface shown in Fig. 2, and two columns in the edit box are respectively property names and property values.

Types of property value in RarLab are very flexible. They can be int, float, double and array. When the property value is array, the data in array is stored in txt file.

RarLab adds a function of hierarchical package. It refers to combine a number of separate modules into a bigger functional module, and the function does not change. Hierarchical package also supports nested package of multiple layers. It means the bigger functional modules made of many separate modules can also be combined into a much bigger module. And the bigger module is able to be decomposed into many smaller modules. The decomposition also supports decomposition of multiple layers.

When users develop a very large signal processing system, the number of signal processing function modules in flowchart can be very much. At this time, packaging the adjacent modules that are used to complete same function into one big module can simplify the flowchart. Dragging and dropping the big module also improves the developing efficiency.

After building one signal processing flow chart, how to save the flow chart so as to modify later? How to get the information from the flow chart to generate advanced language code?

On the basis of XML language [2], RarLab defines a language which describes specially the signal processing flow chart. We call it RML (Radar Markup Language) language. The reason of using XML language is that it has some important advantages, such as opening to public use and independence from platform. In addition, XML language is extensible. So we can define our own language according to demand. An example of RML hierarchical structure is shown in Fig. 3.

According to Fig. 3, we introduce the meaning of the primary nodes and attributes in RML briefly.

rml: the root node, showing the file type and the version number.
system: the radar signal processing system.
application: the signal processing flow chart, including some “actor” nodes, “connect” nodes and “subsystem” nodes.
actor: the function module, including some attributes as follows. “name” attribute: the module name; “chinese_name” attribute: the Chinese name of the module; “class” attribute: the class name of the module; “inport” attribute: the amount of the input ports of the module; “outport” attribute: the amount of the output ports of the module; “x” attribute: the horizontal ordinate in the flow chart; “y” attribute: the vertical ordinate in the flow chart.
editable: the editable attributes of the module, including the feedback value and the runtime.
params: the parameter list, including some “param” nodes. Each “param” node includes some attributes, such as the parameter name, the parameter value and the parameter type.
port: the input or output port. Each “port” node includes some attributes, such as the port name, the data transfer direction and the port type.

connect: the link between two function modules.

src_port: the source port.

dest_port: the destination port.

src_actor: the source function module.

dest_actor: the destination function module.

Subsystem: the hierarchical package module. Each “Subsystem” node includes some “actor” nodes, “connect” nodes and “Subsystem” nodes.

**Modeling of Hardware Topology Structure.** By contrast to the earlier version, adding this function is an important improvement. The modeling of hardware topology structure achieves the generality on the hardware platforms of the RarLab. The software user can use the modules (such as DSP and FPGA) in the hardware module library to build a hardware topology structure chart in a fully graphical and modular way. With the function module library, the hardware module library is also extensible. So the RarLab can describe the hardware topology structure of any hardware platform.

As with the modeling of signal processing flow chart, in order to save and analyze the hardware topology structure chart, on the basis of XML language, we define a language which describes specially the hardware topology structure. We call it HTL (Hardware Topology Language) language.

The basic idea and the grammar of HTL are similar to RML. And the operation of building the hardware topology structure chart is also similar to building the signal processing flow chart. So they are not introduced in detail in this paper.

**Mapping Function.** Mapping function is allocating the function modules in the signal processing flow chart to the hardware modules in the hardware topology structure chart. That means mapping decides which signal processing programs run on which hardware unit. After the allocation, the RarLab can evaluate the performance of the whole hardware and software system.

The RarLab provides two mapping modes: the graphics mode and the forms mode. The graphics mode: the software shows the signal processing flow chart and the hardware topology structure chart together. The software user drags the module in the signal processing flow chart, and drops it on the module in the hardware topology structure chart. The forms mode: the software shows the list of the modules in the signal processing flow chart. The software user selects the hardware module in each row of the list.

Multi-DSP system is in common use in the radar signal processing field [3]. For a long time, at the beginning of system design, it is very hard for developers to suggest the accurate request of the hardware and software. Performance evaluating can guide developers to grasp the demand of the hardware and software at the beginning of system design. There are three common evaluating methods: the regression method, the analytical method and the simulation method [4]. The RarLab uses the analysis method and simulation method in combination, paying more attention to the influence of the hardware for the system performance, and paying less attention to the detail which has little impact on the system.

There are three primary performance indexes, including the time delay of the system, the load balance degree, and the efficiency. Through experimental test, their error rates are respectively 1.94%, 0.23% and 1.18%.

**Automatic Code Generation.** The automatic code generation replaces the traditional artificial programming way. The process is as follows. The RarLab analyzes the signal processing flow chart, the hardware topology structure chart and the mapping solution, and get the useful information from these three files. Then according to the code template, the RarLab generates the C++ code which can be compiled and run on the relevant hardware platform. The structure diagram of automatic code generation is shown in Fig. 4.

The code generated includes five kinds of files. The .cpp/.h files are relevant with the function modules, completing radar signal processing tasks. The .dpg file includes the configuration information of the whole program. The .dpj file describes the whole program is made up with which
files, including the source files, the link files and the library files. The .mak file is used for compiling and linking. The .ldf file describes that the code and the data how to be arranged in the memory.

![Diagram]

Figure 4 The structure diagram of automatic code generation

The automatic code generation is a difficult point in design of the RarLab. The software will be used on different hardware platforms, and the code generated on each platform must be different. So the mechanism of generating code reasonable enough, so that when the software is used on different platforms, we can modify the program as little as possible.

We use the abstract factory pattern in design of the automatic code generation. This is an important improvement by contrast to the original mechanism. The abstract factory pattern provides an interface which creates a series of related or dependent objects without specific class [5]. The abstract factory pattern promotes the object-oriented programming to the interface-oriented programming [6], which fits “OCP(Open Closed Principal)” [7].

We design an abstract class used to generate code. Each subclass of this abstract class is relevant to the code template on a kind of hardware platform. Creation of special object is postponed to the subclass. The calling program of generating code only deals with the abstract class. So when changing hardware platforms, we just need to add a subclass which represents the code template on the new hardware platform. The calling program and the analysis program do not need to be modified.

**Debugging Function.** The Rarlab provides powerful debugging capabilities, including two aspects:

![2D mesh chart](image1)

Figure 5 2D mesh chart

![3D mesh chart](image2)

Figure 6 3D mesh chart

First, memory reading and writing: When the code generated running on the hardware platform, the RarLab can read and monitor the value of any variable or any memory address in the code in real time. The Rarlab can also write a value into any variable or memory address. In addition, the RarLab can also save the data read from the memory into a file on hard disk.

Second, graphing function: The RarLab can not only read data in the memory, and provide real-time graphing capabilities, including the line, 2D mesh and 3D mesh chart. The line chart is relatively simple. The 2D mesh and 3D mesh charts are often used in the field of signal processing. They represent respectively values in the matrix by the brightness and height. The examples are shown in Fig. 5 and Fig. 6.

**PPI (Plane Position Indicator).** When the code generated running on the hardware platform, the RarLab can display the flight path of planes detected by radar in real time, as shown in Fig. 7. Meanwhile, the RarLab can save the data for playback. In addition, it provides many other capabilities, including zooming in and out on the chart, setting the maximum range and the refresh interval.
Conclusions
RarLab software designed in this paper is already running on real hardware platform. The working results indicate that as integrated development environment of radar signal processing system, RarLab is able to replace traditional developing method of writing specific programs, because the software has graphical and modularized developing method. This software shortens the developing cycle and improves the developing efficiency. Besides, the software is of powerful debugging features. Furthermore, the software has excellent generalization and expansibility for hardware platform.

References
[1] L. Zhen, .NET and Design Patterns, Publishing House of Electronics Industry, Beijing, 2005.
[2] W3C. Extensible Markup Language (XML) [EB/OL]. (2007-08-28). http://www.w3.org/XML/
[3] X. Tang, J. Tang, and Y. Peng, Parallel Signal Processing Algorithm and Implementation of Digital Array Radar, Information and Electronic Engineering, 7(4) (2005) 294-299.
[4] V. S. Adve, M. K. Vernon, Parallel Program Performance Prediction Using Deterministic Task Graph Analysis, ACM Transactions on Computer Systems, 22(1) (2004) 94-136.
[5] W. Liang, F. Zheng, Y. Du, and L. Dang, Abstract Factory Pattern. NET Application of Multi-layer Distributed Programs, Computer Era, 3(2006) 27-29.
[6] Yongwu Wang, Yonggang Wang, Object-oriented Guide to Practice, Publishing House of Electronics Industry, Beijing, 2005.
[7] A. Shalloway, J. R. Trott, Design Patterns Explanation, Translated by J. Xiong, Tsinghua University Press, Beijing, 2004.