

# A CONFIGURABLE SIGNAL ANALYZER FOR EMBEDDED SYSTEMS

Kenneth Holladay, Michael A. Koets, Amanda Burmeister, Robert Dollarhide, Denise Varner  
Southwest Research Institute®  
San Antonio, TX

## ABSTRACT

*Rapid adaptability is a critical requirement for modern communication intelligence systems. This paper explores the concept of combining model based design with a runtime framework to realize an easily reprogrammable signal analyzer for embedded COMINT systems. Signal processing experts design algorithms using modeling tools tailored to the domain. The tools produce output that can be loaded directly to the embedded system. Platform independence is achieved by using XML and CORBA as principal elements of the design.\**

## INTRODUCTION

Government and law enforcement agencies employ communications intelligence (COMINT) systems for surveillance of both criminal and military targets worldwide. These systems typically operate in a non-cooperative environment where the COMINT system is not the intended receiver. This task has become more complicated as modern cooperative systems employ adaptive procedures that negotiate waveforms and frequencies to improve communications. A surveillance system must detect target activity in both time and frequency, often in a crowded signal environment. In addition, both the signals of interest and the hardware platform capabilities vary from mission to mission.

Overcoming these obstacles requires not only sophisticated signal processing techniques, but also the ability to rapidly reconfigure the system, often with entirely new algorithms. The deployed software must be scaleable from small portable systems handling a few signals to large distributed systems capable of receiving and identifying hundreds of signals simultaneously.

The key to achieving this flexibility is to perform as much of the digital signal processing (DSP) as possible in software, a critical concept in current Software Defined Radio (SDR) work [1], [2]. Unfortunately, a software implementation is not a panacea. In complex COMINT systems, even small software changes can entail significant

design, development, and testing efforts. The problem is exacerbated by the need for domain experts in DSP algorithm design, data communication, and embedded software, each using different tools on different platforms. Current practice rarely achieves a smooth transition from design to deployment.

This paper explores one solution to the problem using a model-based object-oriented architecture for signal analyzers in an embedded environment. The goal of this work is to deploy and manage multiple signal analyzer (SA) objects distributed on multiple processors and be able to reprogram those objects with new algorithms on the fly.

## RELATED RESEARCH

Academic, commercial, and government organizations are actively involved in SDR research. Standardization efforts include the Software Communications Architecture (SCA) developed by the Joint Tactical Radio System (JTRS) Joint Program Office (JPO) [1] [3], the Object Management Group (OMG), Software Radio Domain Special Interest Group (DSIG) [4], and the Software Defined Radio Forum [5]. Most of this work is based on a cooperative communication environment where the desired waveforms are known. These models do not directly address the insertion points for signal analysis algorithms that are necessary in a blind system to classify a signal.

The execution approach presented in this paper is similar to adaptive software techniques [7], but with the emphasis on rapid reconfiguration based on new algorithm development.

## SIGNAL ANALYZER ARCHITECTURE

Figure 1 illustrates part of a surveillance SDR and the relative location in the data flow of the signal analysis (SA) algorithms under consideration in this paper. Platform independence in the target system is realized by employing CORBA interfaces to define the interaction between the principal objects. For this type of application, the delays introduced by CORBA have been shown to be acceptable [6].

---

\* This work was sponsored by the Defense Advanced Research Projects Agency (DARPA), and the Southwest Research Institute Advisory Committee for Research (ACR).

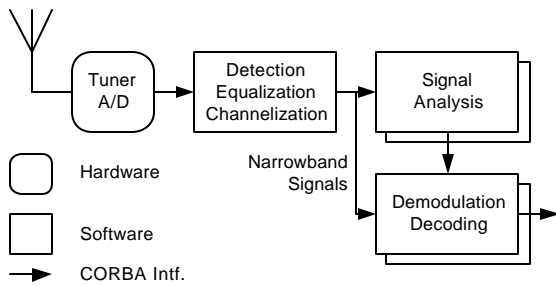


Figure 1. Conceptual COMINT SDR Data Flow

Employing a model-based approach to the design of the SA component allows a signal processing domain expert to build and validate algorithms with minimal concern for the final software implementation. The models are comprised of well-defined DSP blocks connected by links. For example, Figure 2 shows a simple algorithm for estimating the symbol rate of a Phase Shift Key (PSK) signal. This would be a small part of the full SA component.

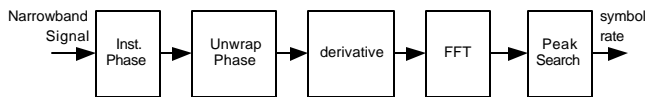


Figure 2. Sample SP Algorithm: PSK Symbol Rate Estimation

The model-based approach to system development can also enable the rapid prototyping of a signal analysis system. A system designed by a signal processing engineer in the modeling environment can be instantiated into a number of operational environments. In addition to the deployable, real-time system described in detail in this paper, the modeling environment can also be configured to build an executable version of the system for a desktop evaluation environment. All that is required to target the modeled system to an operational environment is the creation of the basic DSP blocks for that environment and a means of translating the interrelationships between DSP blocks in the model into an executable format. If the operation of the DSP blocks is consistent between operational environments, then the operation of the complete system will be consistent as well.

The COMINT signal analysis system is modeled as a collection of feature extractors and a signal classifier. The feature extractors represent mathematical operations that are applied to raw signal data to generate a small number of characteristic features of the signal. The classifier accepts a set of feature values and attempts to determine the type of the incoming signal from among a set of possible signal types.

Feature extractors are constructed from the DSP blocks and links that compose the vocabulary of the modeling system. Figure 2 is an example of a simple feature

extractor which reduces an incoming signal to a scalar symbol rate estimate. The features computed by the system are chosen for their ability to discriminate among signals of different types. In many cases, the features will take on physically meaningful values (such as symbol rate) for signals of some, but not all types. However, the feature extractors are constructed so that the mathematical manipulations leading to the feature value will run to completion on any input signal data.

The ordered set of all feature values computed on an input signal can be treated as a vector in a multi-dimensional feature space. Classification is accomplished by defining regions in the feature space corresponding to signals of interest and determining in which, if any, of these regions the feature vector lies. The only requirement on the definition of regions in the feature space is that they be disjoint. The region corresponding to a particular class of signal need not be continuous and the set of regions corresponding to valid signal classes may not occupy the entire space. In fact, rarely will every signal encountered by the classification system belong to a defined signal of interest. In these cases a large region of the feature space will correspond to signals that are not of interest and will be mapped to a default class. No further processing of signals mapped to this class is required, allowing the system to allocate resources to new incoming signals or other tasks.

In many cases it is possible to reach a classification decision without computing the entire set of features. This is accomplished by projecting the signal class regions onto the subspace of the feature space which corresponds to those feature values that have been computed. If it can be unambiguously determined that the incompletely populated feature vector lies in one of the defined signal class regions, or that it lies in none of the defined regions, then the classification process can terminate without the computation of additional features. It is highly desirable to only compute those features necessary to reach the decision since the feature extraction process is often highly computationally intensive.

The conceptual feature extraction and classification architecture is illustrated in Figure 3. A set of feature extractors is defined, each of which provides a feature value to the classifier. The classifier can be executed whenever a new feature value is available. When possible, a classification decision is made and the system is free to process another signal. If a classification decision cannot be made based on the available features, the classifier requests the next feature. In the current system, a simpler execution flow is achieved by computing all feature values and executing the classifier only once.

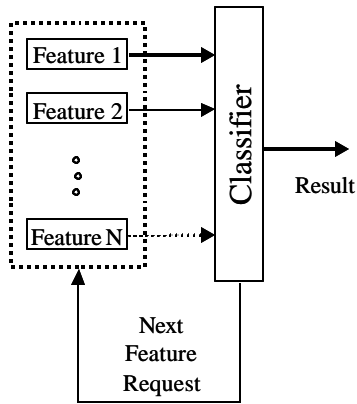


Figure 3. Signal Analysis Classification Architecture

While signal processing for cooperative communication is generally time-continuous (i.e., operating on the data stream as it is collected), many SA algorithms also contain time-discrete functions that periodically operate on large blocks of data. Most commercially available modeling tools have limited support for systems that require a mixture of these two time domains [8]. There are tools under development that are addressing this issue, including the Generic Modeling Environment (GME) [9], MATLAB and Simulink [10], and Ptolemy II [11].

One desirable output format for a description of a system generated by a modeling tool is eXtensible Markup Language (XML). This provides a platform independent transport mechanism based on an established standard. An XML Schema defines the modeling rules to ensure unambiguous semantic content. Concept testing does not require the modeling tool since simple XML models can be constructed manually.

The XML models must be realized as code in the target system. There are several viable methods to achieve this, ranging from code generators to interpreters. The approach pursued for this work combines the efficiency of compiled code with the flexibility of an interpreter while minimizing the prototype development cost. The run-time framework, along with the developmental interface used to exercise and evaluate the system, is shown in Figure 4

The Front End Simulator (FES) is intended to exercise the system in a development environment. It uses a library of test signal files to simulate the system from the tuner through the narrowband channelization. The simulator includes an interface to define various test parameters including the number and temporal distribution of signals to produce, as well as selection criteria for the signals (for example, offset frequencies, modulation types, modulation indices, symbol rates, symbol states, pulse shaping, rolloff, and signal to noise ratios).

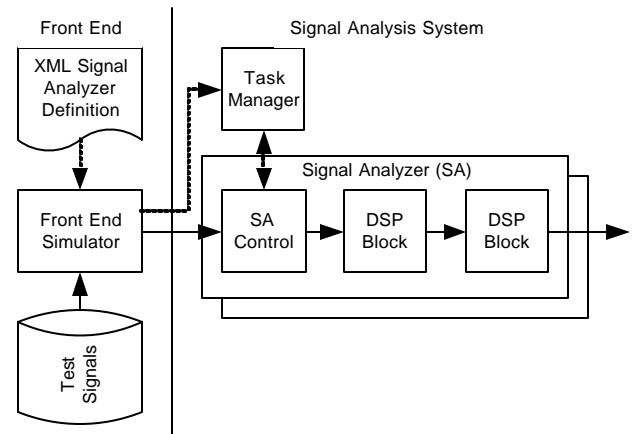


Figure 4. Test Arrangement for Configurable Signal Analyzer

The Task Manager has two principal responsibilities. During initialization, it instantiates the signal analyzers and passes the XML SA definition to the SA Control block. During operation, it is responsible for coordinating the assignment of narrow band signals to SAs as the signals become available at the front end.

The Signal Analyzer component is the housing for the analysis algorithm. When it receives the XML SA definition from the Task Manager, it uses a class factory to construct the appropriate child DSP blocks while operating on them only through an abstract DSP block interface. Figure 5 is a UML class diagram representing the XML format used to describe the SA. The single SA Control block handles the CORBA interface with the rest of the system. During execution, the SA processes the incoming signals by executing the DSP blocks according to the execution strategy specified in the XML definition and outputs signal analysis results.

This architecture provides ready scalability through the addition of processors to run additional signal analyzer objects. The task manager design requires minimal interaction to connect signal data streams to signal analysis objects while providing an access point for supervising and load balancing resources.

## IMPLEMENTATION

Because signal recognition encompasses a vast problem domain, this study focused on a tractable set of well-defined algorithms with increasing levels of complexity to demonstrate the viability of the new architecture. Although these algorithms are not particularly complex or powerful, they make use of numerous fundamental DSP operations.

The CORBA infrastructure was constructed with the ACE\_TAO ORB [12]. The Vanderbilt University Institute

for Software Integrated Systems Universal Data Model (UDM) tool [13] provided the XML services. Class wrappers around legacy C code provided some of the DSP block content for the prototype. Additional DSP blocks were constructed based on the existing DSP library SigLib from Numerix [14]. Such libraries as this or the Integrated Performance Primitives (IPP) library from Intel [15] hold the potential for the rapid construction of much richer sets of DSP blocks.

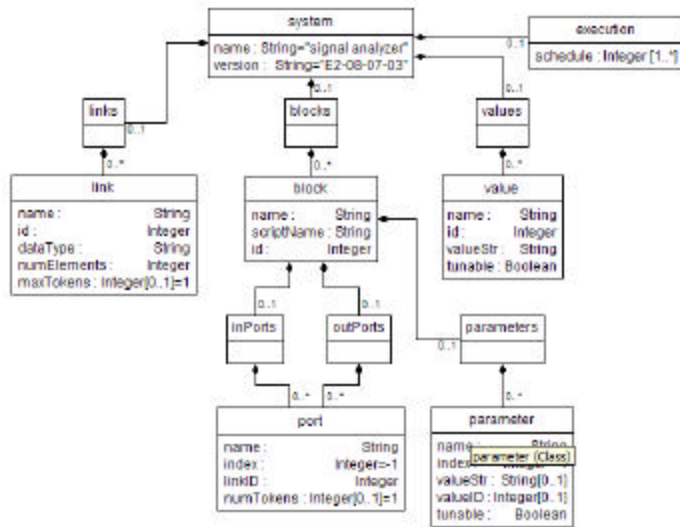


Figure 5. UML Class Diagram Representing the XML Format for the Signal Analyzer Definition

The XML description of a signal analysis system consists of three basic elements and is illustrated in the XML code snippet of Figure 6. The first element is a list of the DSP blocks which make up the system. Each block definition includes information about the inputs and output ports, and parameters of the block. Inputs and outputs are associated with links and the data on these ports will change at run time. Parameters are established when the system is instantiated and are used to refine the operation of the block.

The second element of the system description is the execution sequence which determines the order in which the DSP blocks are executed. The execution sequence is established in advance of system instantiation.

The final element of the system description is the list of system links. Link definitions include the type and size of their data.

Figure 7 is a sequence diagram that illustrates the basic system communication enabling data flow to an SA. When a signal is detected, the signal source (in this case the front end simulator) notifies the Task Manager of the arrival of the new signal. The Task Manager then selects

an available Signal Analyzer and assigns the new signal to it.

```
<block id="7" name="OBDifference">
<inPorts>
<port index="-1" linkID="2" name="inPort"
numTokens="1"/>
</inPorts>
<outPorts>
<port index="-1" linkID="6" name="inPort" numTokens="1"/>
</outPorts>
<parameters>
<parameter index="0" name="Stride" valueStr="1"/>
<parameter index="1" name="absFlag" valueStr="1"/>
</parameters>
</block>

<execution
schedule="11;16;18;20;63;64;65;69;87;88;89;102;2;1;0;17;32;34
;33;42;72;73;71;93;95;94;99;100;98;104;4;5;13;8;10;19;9;24;27
;14;30;6;26;28;23;3;7;21;15;29;22;12;25;31;45;53;46;62;66;54;
55;56;49;50;44;35;47;68;61;58;59;36;60;57;48;43;51;40;37;41;3
9;38;67;52;70;76;86;77;90;85;84;83;82;81;75;80;74;91;79;78;92
;97;96;101;103"/>

<link dataType="DT_DoubleComplex" id="121" maxTokens="1"
name="outPort" numElements="8192"/>
```

Figure 6. XML Snippet of a SA Definition

The SA communicates directly with the signal source to initiate data flow. This sequence decouples signal sources from signal analyzers while maintaining the more efficient direct interface for data flow. The signal source pushes data to the SA until either the signal goes down or the SA completes its analysis. The SA disconnects from the signal source, then returns its result. XML is also used to express the results of the analysis using a schema defined in previous work [16].

## INITIAL RESULTS

The described system was implemented in ANSI C++ using Microsoft Visual Studio 6.0 and GCC 3.2. With the UML design, software construction proceeded smoothly. Both the ACE-TAO CORBA ORB and the UDM performed well.

The test environment included both Windows 2000 and Redhat Linux 8.0 operating systems on Intel x86 processors. In addition to the benefits of flexibility and rapid reconfiguration, this architecture simplified multiplatform testing requirements.

Three signal analysis systems have been constructed for this runtime framework to date. The first experimental system consisted of a PSK baud rate estimator. The structure of this system is that shown in Figure 2. The second system performed the much more complex signal recognition function of identifying signals with PSK modulation from a diverse set of test signals. The third system was designed for a similar function, but had the objective of classifying incoming signals as PSK,

frequency shift keyed (FSK) or neither. This final system consisted of more than 750 DSP blocks and more than 1400 links, demonstrating the scalability of the framework to fairly complex systems.

All three signal analysis systems were constructed for execution within the reconfigurable signal analyzer. Each system was extensively exercised with a large body of simulated tests signals delivered by the FES. Performance of the system, in terms of the number of correct classifications, was found to be consistent with the performance of systems built by hand for a desktop simulation environment.

The first two analysis systems were constructed by the labor intensive process of hand coding the XML definition of the SA. At the time the third system was implemented, the platform had reached a level of capability suitable for use with the GME [9] model-based development tool. Use of the model-based tool substantially decreased the effort required to define this complex signal analysis system.

### CONCLUSIONS

The prototype system accomplished the goal of rapid reconfiguration. Different SA algorithms were quickly and easily brought on-line in the target environment. New algorithm configurations were constrained only by the limited block set implemented in the prototype. This also demonstrated the viability of separating the algorithm design from run-time considerations.

Future work will include execution optimization, memory optimization, CORBA delay timing, and quality of service management for the Task Manager.

### REFERENCES

- [1] Joint Tactical Radio System (JTRS) Joint Program Office (JPO), "Software communications architecture specification," v2.2, <http://jtrs.army.mil/>, 17 Nov 2001.
- [2] B. Tarver, E. Christensen, A. Miller, "Software defined radios (SDR) platform and application programming interfaces (API)," Military Communications Conference (MILCOM 2001), pp. 153-157, 2001.
- [3] M. Robert, J. Reed, "Software design issues in networks with software defined radio nodes," IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2001.
- [4] Object Management Group (OMG), Software Radio Domain Special Interest Group (DSIG), <http://swradio.omg.org/>, Mar. 2003.
- [5] Software Defined Radio Forum, [www.sdrforum.org](http://www.sdrforum.org), Mar. 2003.
- [6] J. Bertrand, J.W. Cruz, B. Majkrzak, T. Rossano, "CORBA delays in a software-defined radio," IEEE Communications Magazine, vol. 40, Issue 2, pp. 152-155, Feb 2002.
- [7] J. Sztipanovits, G. Karsai, T. Bapty, "Self-adaptive software for signal processing," Communications of the ACM, vol. 41, no. 5, May 1998.
- [8] M. Kühn, C. Reichmann, I. Prötel, K.D. Müller-Glaser, "From object oriented modeling to code generation for rapid prototyping of embedded electronic systems," Proceedings of the 13<sup>th</sup> IEEE International workshop on Rapid System Prototyping, pp. 108-114, 2002.
- [9] A. Ledeczi, et. al., "The generic modeling environment," [www.isis.vanderbilt.edu/Projects/gme/](http://www.isis.vanderbilt.edu/Projects/gme/), 2001.
- [10] Mathworks, [www.mathworks.com/products/](http://www.mathworks.com/products/), 2003.
- [11] UC Berkeley, Ptolemy project, <http://ptolemy.eecs.berkeley.edu>, 2003.
- [12] The ACE ORB, <http://www.theaceorb.com/>, 2003
- [13] Vanderbilt University Institute for Software Integrated Systems, <http://www.isis.vanderbilt.edu/projects/mobies/default.html>, 2003.
- [14] Numerix-DSP, [www.numerix-dsp.com/](http://www.numerix-dsp.com/), 2003.
- [15] Intel, Integrated Performance Primitives, [www.intel.com/software/products/](http://www.intel.com/software/products/), 2003
- [16] K. Holladay, K. Robbins, D. Varner, "An XML schema for digital communication signal research," XML Conference 2002, Baltimore, MD, December 2002.

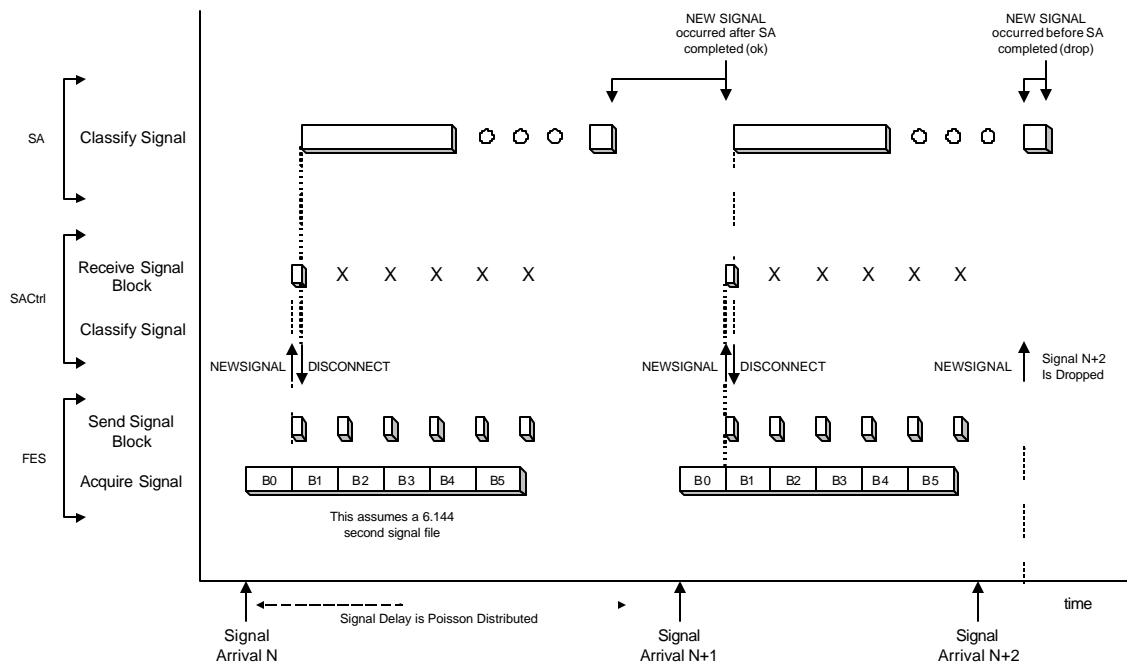


Figure 7. Illustration of Timing for Runtime Framework