

A CASE STUDY IN APPLYING QOS ADAPTATION AND MODEL-BASED DESIGN TO THE DESIGN-TIME OPTIMIZATION OF SIGNAL ANALYZER APPLICATIONS

Joseph Loyall
Jianming Ye
Richard Shapiro and
BBN Technologies
Cambridge, MA

Sandeep Neema
Nagabhushan Mahadevan
Sherif Abdelwahed and
Vanderbilt University
Nashville, TN

Michael Koets
Denise Varner
Southwest Research Institute
San Antonio, TX

ABSTRACT

The capture and classification of digital signals is an important part of military communications and of signal intelligence (SIGINT). A typical signal analyzer is built from a set of elementary signal processing operations, which often include parameters whose values can affect the quality of the signal classification. In this paper we present a case study we have conducted to evaluate the efficacy of automated parameter tuning for improving signal classification. We use a prototype signal analyzer parameter tuner, which augments a signal analyzer design with a search space controller driven by a utility function based on correct classification. It tunes parameters by automatically tweaking parameter values in a systematic way and evaluating the utility of the signal analyzer with different parameter values over a set of representative signals with known ground truth. We developed the parameter tuner using a QoS adaptive design tool we developed under the DARPA MoBIES program. We have achieved improved signal classification in three signal analyzers studied. In addition, the automated parameter tuning exercise has the side effect of providing increased understanding of how parameters contribute to signal analysis. The paper describes the signal analyzer parameter tuner, the experiments that we conducted as a part of our case study, and the empirical results of the experiments.

INTRODUCTION

Government and law enforcement agencies are frequently required to monitor, intercept, and analyze radio communication signals in order to identify important signals requiring additional processing. The volume of monitored signal traffic is often very large and the number of signals of interest within that traffic is typically small. As a result, it is necessary to develop automated *signal analyzers* that can screen the signal traffic, identify the signals of interest, and classify those signals into groups for additional processing.

The data input to a signal analyzer consists of sampled data representing communications signals that have already been recognized as distinct from background noise and have been isolated in time and frequency. The signal

analyzer analyzes these signals and classifies them into groups based on their modulation characteristics. Once a signal has been classified into such a group, additional analysis performed on that signal can be tailored to its particular modulation type.

Designing and implementing these signal analyzers is a specialized skill, typically done by hand by domain experts, and involves several significant challenges. While the characteristics of the signals of interest are well defined, different signal types are often specified at differing levels of detail. This requires the signal analyzer to be sensitive and to be able to discriminate against signals with subtle differences. It is generally not possible to characterize all of the signals that the analyzer will encounter during operation. This implies that the analyzer will be confronted with highly structured but unknown signal types during operation, and these must not significantly impact its ability to correctly classify the signals of interest. The quality of a given signal analyzer depends greatly on the quality of the signal processing algorithms, the composition of the signal traffic it encounters, and the way in which the signal analyzer is configured, including the values chosen for parameters of the signal processing algorithms.

The Model Based Integration of Embedded Systems (MoBIES) program is researching the use of Model Integrated Computing (MIC) [5] for the development of embedded systems, such as signal analyzers. As part of this, we have developed the Distributed QoS Modeling Environment (DQME) [7], a modeling language for designing applications with adaptive Quality of Service (QoS) management. Using DQME, we have developed a prototype automated parameter tuner that can optimize the configuration of a signal analyzer by systematically tuning its parameters over several runs against signal sets with known ground truth to achieve the best classification (i.e., the highest QoS).

In addition to validating the approach of DQME and modeling of QoS adaptive applications, our goals in applying DQME to the tuning of signal analyzers include the following: improving the quality of signal classification; reducing the effort involved in designing,

testing, and optimizing signal analyzers; and taking a first step toward runtime tuning of signal analyzers.

The next section describes the architecture of signal analyzer applications. Following this, we describe the prototype parameter tuner that we have developed and the MIC and adaptive QoS middleware foundations upon which it was constructed. We then describe a case study of applying the parameter tuner to three signal analyzers and the improvements in classification we observed as a result of tuning. Finally, we provide some discussion and concluding remarks.

SIGNAL ANALYZER APPLICATIONS

Signal analyzer applications in this study are built on the concept of feature extractors, as illustrated in Figure 1. A *feature extractor* is a sequence of signal processing operations that operate on an incoming signal and yield a real-valued feature. Feature extractors are designed to yield feature values that are useful for discriminating between different signal types and identifying signals of interest. A signal analyzer consists of several feature extractors, which pass their feature values to a *classifier* operation. The classifier uses the feature values to identify the class of the incoming signal. Feature extractors themselves are constructed from a library of low-level signal processing operations, referred to as *OpBlocks*, interconnected to implement the feature extraction algorithm. OpBlocks are often controlled by parameters, which influence the details of their operation. The classifier itself may also be controlled by parameters.

Traditionally, these parameterized signal analyzers are built by hand by signal processing domain experts using various OpBlocks in Matlab. One of the major challenges in building them with standard means is that the ‘optimal value’ for any tunable parameter in a given OpBlock often varies from signal analyzer to signal analyzer and from feature extractor to feature extractor. Optimal tunable parameter values in these blocks depend on several factors, including the overall algorithm used, the types of signal expected, and the distribution of these signals.

Changes in parameter values can influence the performance of the signal analyzer in a number of ways. Parameters controlling OpBlocks affect the behavior of the signal processing operations and, therefore, the details of the feature extraction algorithm. Often, the feature extractors implement formal statistical estimators, which compute an estimate of some characteristics of the signal. Changes to parameters involved in such feature extractors can cause the feature extractor to become a better or worse estimator. In general, the effect of these parameter changes on the accuracy of the system is difficult to predict and must be determined experimentally.

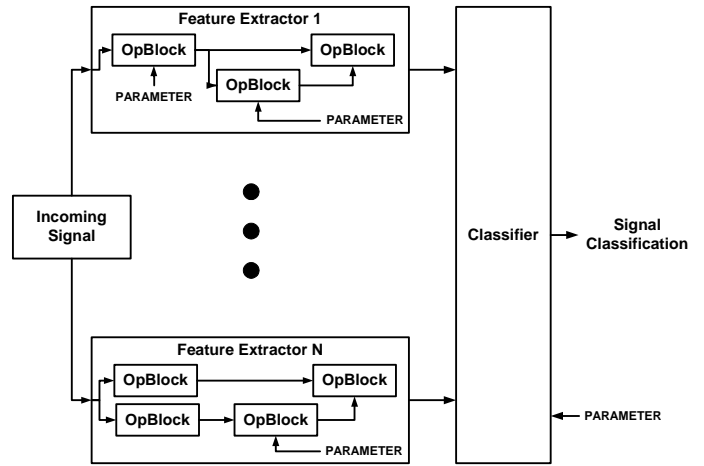


Figure 1. Architecture of a Signal Analyzer

Parameters are also used to control some of the operations of the classifier block itself. Changes to these parameters generally do not impact the computational complexity of the classification process, but can have a dramatic impact on the performance of the system. In some cases, information about the statistics of the signal types and the distributions of the features is known in advance. In these cases, it is theoretically possible to predict the impact of changes to the classifier parameters on the classification performance of the system. Often, however, this statistical information is unavailable and the impact of changes to classifier parameters on system performance must be determined experimentally.

One of the daunting tasks facing a domain designer is to pick the right parameter values so that the signal analyzer can achieve the optimal or near optimal classification. Hand picking these values is time-consuming and labor intensive.

PARAMETER TUNER DESIGN

A. The Architecture

We designed the automated parameter tuner with the DQME modeling environment [7], a domain-specific modeling environment developed in the Generic Modeling Environment (GME) [3], supporting the design and synthesis of applications that can measure, control, and adapt to QoS requirements and levels. DQME generates Matlab simulation code, C++ and CORBA application code, and QoS adaptive code in the Quality Objects (QuO) middleware framework [4, 6].

Our prototype adaptive tuner augments a signal analyzer design with a search space controller driven by a QoS utility function based on correct classification of signals. The architecture of the parameter tuner is illustrated in Figure 2. The search space controller uses a local search technique (a detailed analysis of this search technique can be

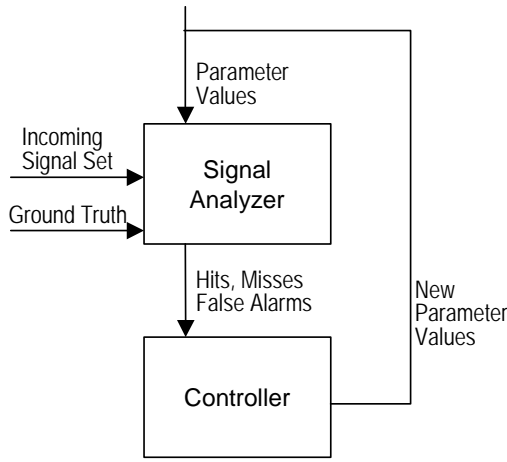


Figure 2. Architecture of the Automated Parameter Tuner

found in [2]). Local Search involves searching the parameter space in the neighborhood of the current parameter settings. Each parameter value is changed by $[\Delta, 0, -\Delta]$, where Δ is the step size of the parameter. The QoS is evaluated for all combinations of the parameter values resulting from the change in parameter values. The parameter settings with the maximum QoS value are chosen to be the new current parameter settings. This is repeated for each signal set.

The QoS utility function is calculated based on the classification outcomes and the controller determines how to tweak the parameters based on the QoS performance metric (utility) over a set of signals with known ground truth. We discuss the QoS metric for measuring the performance of the signal analyzer next.

B. QoS Performance Metric

A QoS performance metric or utility function is required to drive parameter adaptation. This metric produces a quantitative report on the quality of system operation. The performance metric permits the parameter tuner to evaluate whether changes to parameter values improved or degraded the performance of the system, and thereby guide the direction of further changes to the parameters.

We have chosen a design-time performance metric, which can be computed only when the system is operating within a controlled test. The metric is based on the average classification accuracy over a set of signals and is formulated in terms of the classification outcome. We have defined five possible classification outcomes: hit, miss, correct rejection, false alarm, and drop (which occurs when a signal is not processed due to system overload). Since the signal classification outcomes can only be determined with the known ground truth of the signal, this

design-time metric cannot be computed in an operational environment where the signal ground truth is unknown.

To compute a quantitative performance metric, we assign a cost to each possible outcome, then average the cost incurred over a set of signals processed. The cost assignments are listed in Table 1.

Table 1. Cost Factor for Classification Outcomes

Classification Outcome	Cost
Hit	0
Miss	1
Reject	0
FalseAlarm	1
Drop	0.5

Therefore, the QoS utility function can be defined as:

$$QoS = 1 - \frac{1}{N} \sum_{i=1}^N v_i$$

where i is the index of the current signal, N is the size of the window, and v_i is the cost of the i th signal processing outcome. Our adaptation (tuning) strategy is then to maximize QoS for each signal set and minimize the cost associated with each signal classification by adjusting parameter values.

SIGNAL ANALYZER CASE STUDY

To evaluate the efficacy of our parameter tuner, we applied it to the development of three signal analyzers, each with differing levels of complexity and differing characteristics. In this section, we describe the results of these experiments.

A. A Signal Analyzer for PSK Signals

The first experiment consisted of the development of a simple signal analyzer for identifying *phase shift keyed* (PSK) signals. The analyzer accepts input vectors of sampled signal data and must decide whether the input signal is a PSK signal or not. The PSK signal analyzer consists of 57 OpBlocks in four feature extractors: Number of Spectral Tones (NT), Symbol Rate (SR), Bandwidth (BW), and Number of Phase States (NP) and was constructed from readily available public domain algorithms (which are usually not optimal).

With the help of a domain engineer, we selected three parameters for tuning and initial values for those parameters (NT_Squelch = 10, NP_PeakDip = 4, NP_Squelch = 25). We attached the search space controller to the PSK signal analyzer in DQME and generated Matlab code to do the tuning. The Matlab parameter tuner produced parameter values that exhibit considerable improvement in the PSK

signal recognition. With the initial values for the above parameters, the QoS value was 0.75, meaning that the signal analyzer properly classified approximately 75% of the incoming signals in a representative sample of 188 signals. The parameter tuner converged on a set of optimal parameters after 6 iterations (using a window size of 188 signals). The resulting set of parameter values (NT_Squelch = 1, NP_PeakDip = 0.5, NP_Squelch = 11) has a QoS value of 0.971, or over 97% correct classification, nearly a 10x reduction in misclassification from 25% to 2.9%.

B. A Larger Example: Classifying PSK and FSK Signals

In a second experiment, we designed and tuned a signal analyzer that fully classifies incoming signals as PSK, FSK (Frequency Shift Keyed), or NOTA (none of the above). This signal analyzer was also constructed from public domain algorithms and consists of over 700 OpBlocks organized into the following five feature extractors: Number of spectral tones, Mean channel bandwidth, Mean channel symbol rate, Mean channel number of phase states, and Mean channel kurtosis.

With the help of a domain engineer, we identified four significant tunable parameters in the signal analyzer: NT_PeakDip (in the number of spectral tones feature extractor), and NP_Squelch, NP_NumAvg, and NP_NumBins (all in the phase states feature extractor). The domain engineer also determined possible operational ranges and initial default values for these parameters.

Starting from their initial default values, the parameter values converged in fewer than 10 iterations (10 signal sets each with 388 signals), as shown in Figure 3. In this experiment, we tuned the parameters in a Matlab simulation and a C++ implementation of the signal analyzer.

The results of this experiment along with results from the first experiment are summarized in Table 2. The QoS improved in both the Matlab simulation and the runtime. In Matlab, the initial QoS with the default parameter values was 0.6875 or about 69% correct classification. After adaptation, the final QoS improved to 0.8819 or 88% correct classification. The classification error was reduced by 62%, down from 31% to only 12%. In the C++ runtime system, QoS improved from 0.7036 to 0.8247 and classification error reduced from 30% to 18%, or a 41% reduction

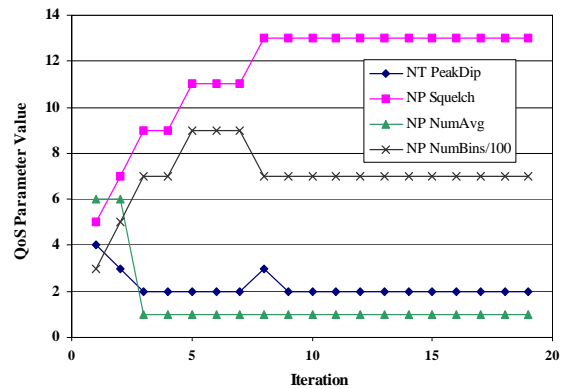


Figure 3. Parameters Change during Adaptation

in error. The differences observed between the simulation and runtime results are mainly due to the differences in the OpBlock implementations in Matlab and C++, and the tuning of one fewer parameter in the C++ (NP_NumBins was not accessible for tuning in the C++ OpBlocks). These subtle differences are sometimes hard to eliminate completely, thus tuning these parameters in the actual runtime environment is critical to optimize the performance of a deployed system.

C. A Highly Optimized Signal Analyzer

In a third experiment, Southwest Research Institute applied the parameter tuning tool to a signal analyzer that was developed using proprietary (and therefore better) algorithms and that had already been highly tuned by hand by the system designers. The purpose of this experiment was to simultaneously evaluate the time and effort involved with automatic parameter optimization as part of an overall MoBIES tool chain for signal analyzer design and to evaluate its ability to further tune a highly optimized signal analyzer.

The signal analyzer built under this experiment was designed to identify PSK, FSK, and multi-carrier PSK signals from a collection of signals containing these types as well as noise, continuous wave (CW), and analog modulated signals. The experiment consisted of several phases in which the system was modeled, implemented, tested, and refined using various MoBIES tools. Three parameters controlling the behavior of the classifier were selected for optimization. The system designers had already carefully

Table 2. Signal Analyzer Performance Improvement in Matlab and Runtime with Parameter Optimization

Platform		Initial Performance		Final Performance		Classification Error Reduction
		QoS	Classification	QoS	Classification	
PSK only	Matlab	0.7472	74.7%	0.9710	97.1%	88.5%
PSK and FSK classifier	Matlab	0.6875	68.8%	0.8819	88.2%	62.2%
	Runtime	0.7036	70.3%	0.8247	82.5%	40.9%

selected the initial values of these parameters and SwRI expected that there was only limited potential for improvement.

The evaluation of the parameter tuner began with a complete signal classification system model built using other MoBIES tools in an earlier phase. A knowledgeable domain engineer selected the significant parameters for tuning and manually optimized the signal analyzer. Parameter selection and manual tuning took about 1 and 8 hrs, respectively. This model was then imported into GME under the DQME paradigm and connected to the search space controller with three parameters selected for tuning. The combined system incorporating the signal analyzer and controller was then instantiated into a format that could be executed in Matlab, including script files to be used during the optimization process. This process took about 1.25 hrs after which the parameter tuner ran automatically until completion overnight.

The signal analyzer was evaluated using a fixed set of 5000 signals. The overall performance of the signal analyzer system before and after parameter tuning is shown in Table 3. In addition, Table 4 indicates the percentages of signals of interest properly classified and the percentages of signals not of interest properly rejected, before and after tuning.

Table 3. Optimized Signal Analyzer Performance Improvement with Parameter Optimization

Experiment 3	Initial Classification Percentage	Final Classification Percentage	Classification Error Reduction
PSK, FSK, Multi-Carrier PSK	89.12%	89.22%	1%

Table 4. Classification Breakdown Before and After Parameter Optimization

PSK, FSK, Multi-Carrier PSK	Percent Signals Properly Classified	Percent Signals Properly Rejected
Initial	89.36%	88.41%
Final	89.55%	88.41%

As expected, the initial values chosen by the system designers appear to be near optimal for this signal analyzer and signal set. Even so, the parameter tuning tool did succeed in finding values with marginally improved performance. It is obvious that manually optimizing the parameters is tedious and requires significant effort from highly

skilled domain engineers (1 hour selecting the parameters and 8 hours tuning them). Using automated tuning in place of manual tuning would dramatically reduce the effort of the domain engineer (from 9 hrs to just 2.25 hrs) with the potential tradeoff that the automatic tuning time may increase. However, the cost associated with increased system tuning time can be minimal, since the automated tuner can run non-stop and in parallel as long as the computing resources are sufficient.

DISCUSSION

As mentioned earlier, our goals in undertaking the case study described in this paper were twofold:

- Evaluate the usefulness of our DQME modeling environment for building QoS adaptive applications
- Apply QoS adaptation to the design and development of more accurate signal analyzers.

Despite the success we have reported, we are just beginning to explore the possibilities of modeling QoS adaptive applications. The application to signal analyzer tuning only exercises a part of the capabilities that we have developed in DQME. We are currently applying it to other domains to build upon the results we have had in the signal analysis domain.

With regard to the parameter tuning tool and its use in the design and development of more accurate signal analyzers, we have clearly shown success. It is difficult to measure specifically how much success since the accuracy of a signal analyzer depends on many factors. The values chosen for parameters is just one of the factors in the quality of the signal processing algorithm, but is one that is amenable to automated QoS tuning. There is an upper bound on how accurate a particular signal classification can be, based on the signal classification algorithm and the signal set, but this upper bound is impossible to accurately determine in practice, and will likely change from laboratory testing to deployment. The correct way to judge how well the parameter tuner performed is to identify how close the signal analyzer gets to the upper bound on classification after tuning, impossible to do in practice. An 88% correct classification percentage after tuning for one signal analyzer might or might not be better than a 97% percentage after tuning for another, all depend on the theoretical upper bounds in each case. Similarly, comparing the percent improvement in two signal analyzers before and after tuning can be misleading, since it can depend significantly on how carefully and how well the values for the parameters were chosen during the signal analyzer design. That is, adaptive parameter tuning should have much less effect on carefully chosen parameter values than on less carefully chosen ones.

This leads us to examine exactly how the adaptive parameter tuning tool can be used in a signal analysis development process:

- *Reduce the design effort.* As mentioned in experiment 3, if automated parameter tuning can reduce the amount of time and effort the designer spends on selecting the right values for parameters, testing the signal analyzer against signal sets, and configuring the feature extractors, it could be a significant win. Especially so, if the saved time can be spent on improving the signal analysis algorithms or better identifying the characteristics of signals in the deployment environment.
- *Validate parameter values.* Currently, the designer that spends a significant amount of time configuring the signal analyzer has no real way to validate the values chosen. The parameter tuning tool provides a “sanity check.” Even if parameter tuning results in little or no improvement in classification, as in our third experiment above, it provides validation for the chosen values and at least some indication of how close the signal analyzer is to its upper bound on classification.
- *Improve classification.* Of course, the main use indicated by our experiments is that the parameter tuning tool can improve the performance of signal analyzers, even highly optimized ones. Performance improvement is not necessarily linear, either. It might be simple for a designer to choose parameter values that get close to optimal classification, even while eking out an extra percentage or two of classification is crucial, but prohibitively expensive to do manually.

One of the side effects of the work presented in this paper is improved understanding of the factors contributing to the performance of signal analyzers. We are building upon the results we have reported to try to improve our support for signal analyzer design. First, we are working on better search algorithms than the “brute force” search we report in this paper. We have developed two new algorithms that we are presently evaluating. Our initial experiments are indicating improved speed of tuning with similar tuning results with the new algorithms.

We are also continuing to study the factors contributing to the speed and accuracy of parameter tuning in all the techniques under development. These appear to include the number of parameters chosen, the specific parameters (significant or not) chosen, and the order of tuning. This has led us to explore strategies for identifying significant parameters, analyzing their dependencies, and grouping them according to feature extractors. Simultaneously, we are actively researching runtime signal analyzer optimization using the expected cost/payoff associated with each signal classification as the runtime QoS metric

and exploring techniques to improve overall system performance and throughput through online dynamic resource management [1]. The runtime tuner would allow a signal analyzer to dynamically adapt to changing signal environments after deployment.

CONCLUSIONS

The work described in this paper has shown the successful application of QoS adaptation and modeling to the design of signal analyzer applications. It has the possibility of contributing significantly to the science and practice in two areas: QoS adaptation for embedded systems and the design of optimized signal analyzers. The continued exploration of both of these simultaneously has the potential for building upon the successful results reported in this paper and carrying over into the deployment of signal analyzer applications and tuning of them at runtime.

ACKNOWLEDGMENTS

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) under contract F33615-02-C-4037.

REFERENCES

- [1] Abdelwahed, S., Neema, S., Loyall, J., and Shapiro, R. A Hybrid Control Approach for QoS Management. The 24th IEEE International Real-Time Systems Symposium, Cancun, Mexico, 2003.
- [2] Abdelwahed, S. and Neema, S. Efficient Performance Tuning algorithms for a general class of computational systems. Technical Report, 2004.
- [3] Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., and Volgyesi, P. The Generic Modeling Environment. WISP'2001, Budapest, Hungary, May 24-25 2001.
- [4] Loyall, J., Bakken, D., Schantz, R., Zinky, J., Karr, D., Vanegas, R., and Anderson, K. QoS Aspect Languages and Their Runtime Integration. Lecture Notes in Computer Science, 1511, Springer-Verlag. Proceedings of the Fourth Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers (LCR98), May 28-30, 1998.
- [5] Sztipanovits, J. and Karsai, G., Model-integrated computing. Computer 30, 4(April, 1998), 110-111.
- [6] Zinky, J., Bakken, D., and Schantz, R. Architectural Support for Quality of Service for CORBA Objects. Theory and Practice of Object Systems 3(1). 1997.
- [7] Ye, J., Loyall, J., Shapiro, R., Neema, S., Mahadevan, N., Abdelwahed, S., Koets, M., and Denise, W. A Model-Based Approach to Designing QoS Adaptive Applications. IEEE RTSS, 2004.