A Model-Based Approach to Designing QoS Adaptive Applications

Jianming Ye
phone: 617-873-3486
e-mail: jye@bbn.com

Joseph P. Loyall
BBN Technologies

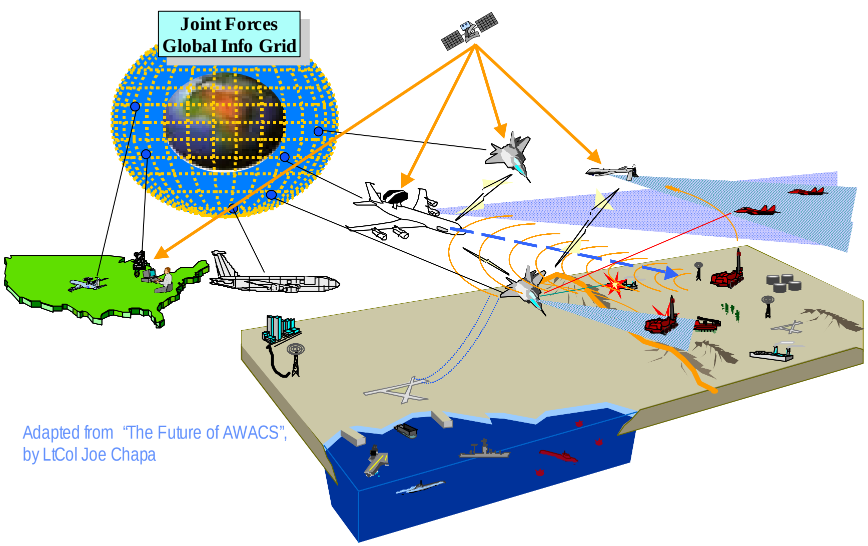
Sandeep Neema, Sherif Abdelwahed, Nagabhushan Mahadevan
Vanderbilt University

Motivation and Background

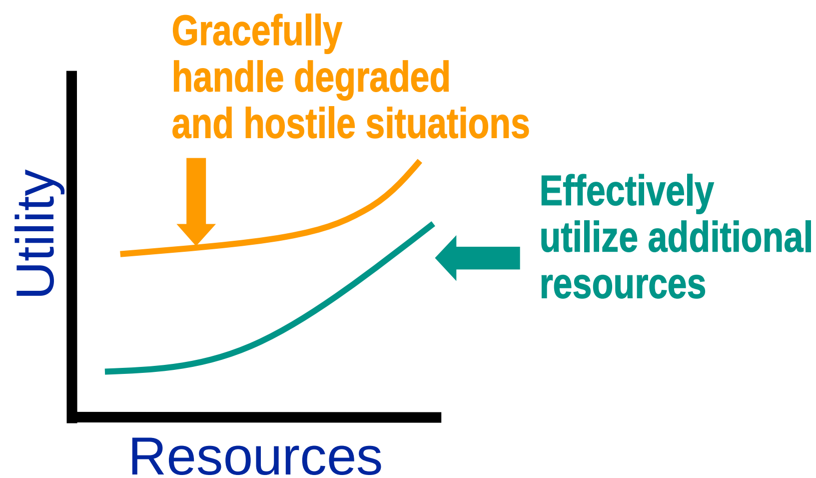
- Applications are distributed and network centric
- Stringent QoS requirements, including predictable and efficient data transfer and control
- Resources are constrained and shared
- Operate in dynamic environments

An emerging trend of customized critical distributed real-time embedded systems that are required to operate in highly dynamic networked environments

The need to handle dynamic conditions effectively with predictable quality of service



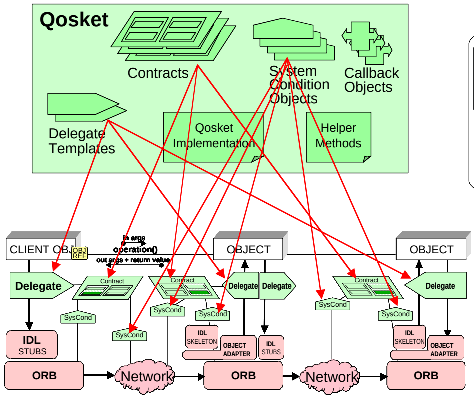
Adapted from "The Future of AWACS", by LtCol Joe Chapa

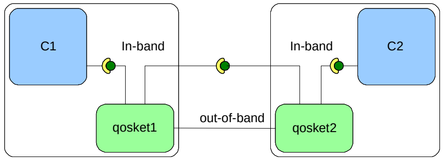


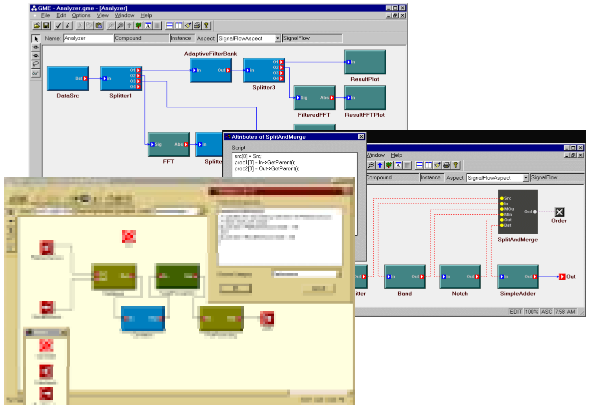
Our Goals

1. Identify essential elements of QoS adaptation that need to be captured in the modeling tool
2. Insert notations of these elements in design tools to enable quality-of-service modeling
3. Simplify the development of end-to-end and system-wide QoS control and adaptation for DRE systems
4. Improve the formalization, usability, and robustness of QoS adaptive concepts
5. Provide automated code synthesis and model refinement capabilities
6. Develop tools for predictable, controlled behavior and graceful degradation and recovery

Technology Bases – QuO Adaptive Middleware and GME Metamodeling Environment





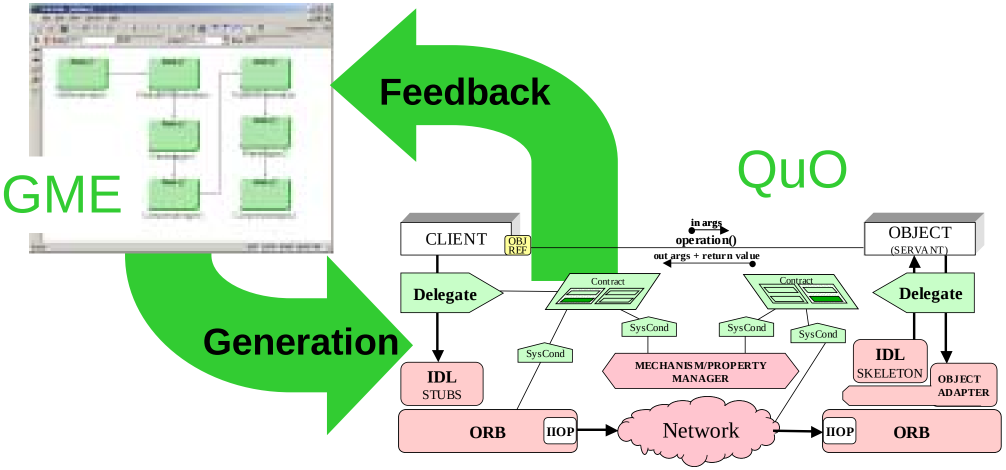


- QuO is a middleware framework for developing QoS adaptive distributed systems
  - Works with CORBA, Java RMI
  - Provides measurement, control, and contract-driven adaptation
  - Provides a QoS behavior encapsulation model, called *qoskets*
  - Available open source at http:://quo.bbn.com

- GME is a framework for creating domain-specific design environments
  - Supports abstract modeling concepts generic enough for a wide range of domains
  - Uses metamodeling to define the family of domain models that can be created
  - Supports tool integration and extensibility
  - Available at http://www.isis.vanderbilt.edu/Projects/gme/default.html

Technical Approach

- Develop a modeling language supporting the high-level design of adaptation strategies in GME
- Map high-level specifications to run-time interfaces and constructs, such as QuO
- Feedback loop for incorporating runtime information into refinement of design-time model
- Simulation and analysis of adaptation strategies
- Demonstrate and evaluate in different DRE applications

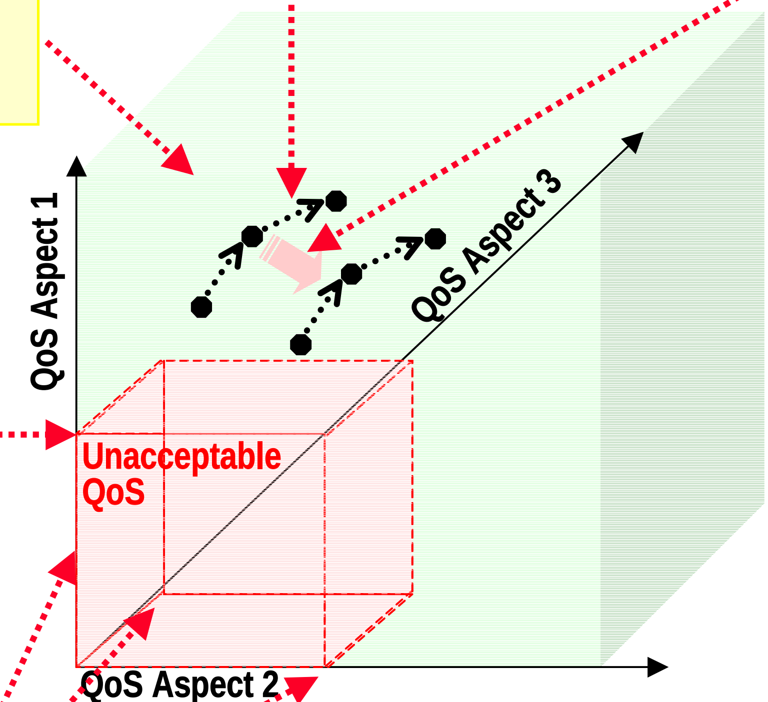


Dissecting the Constituent Elements of QoS Adaptation

The domain and mission define an *upper bound* on the QoS, e.g.,
- full motion video
- highest resolution a sensor can provide

QoS *knobs* determine what can be changed

QoS dimensions are not independent, knobs frequently affect more than one, e.g.,
- compressing data decreases network bandwidth but increases CPU usage
- increasing security commonly affects performance



and a *lower bound*, e.g.,
- minimum useful resolution
- lowest useful rate

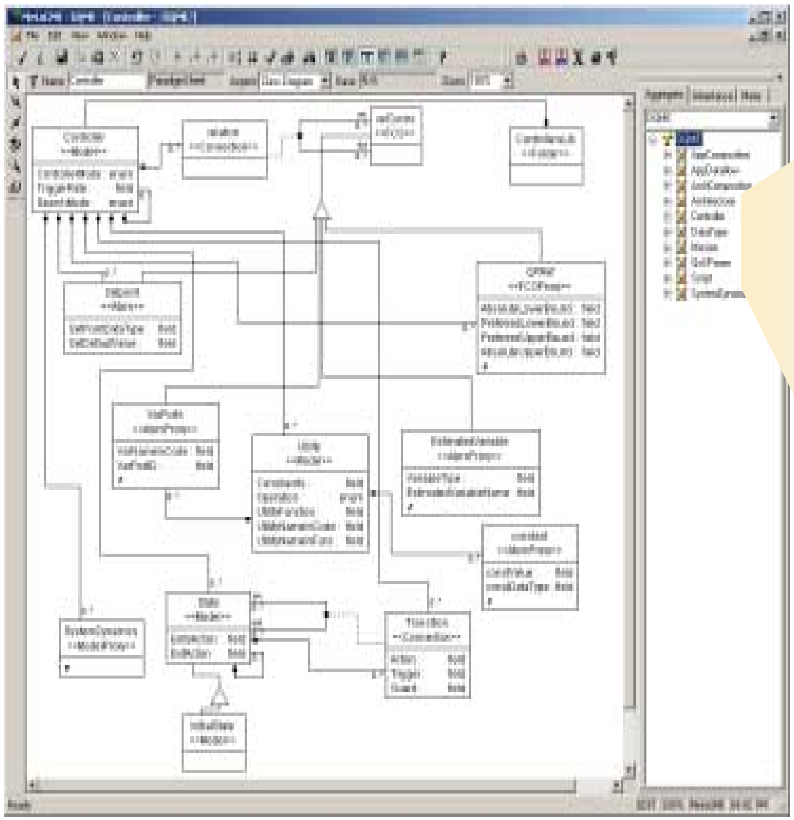
QoS aspects define a multi-dimensional adaptation space

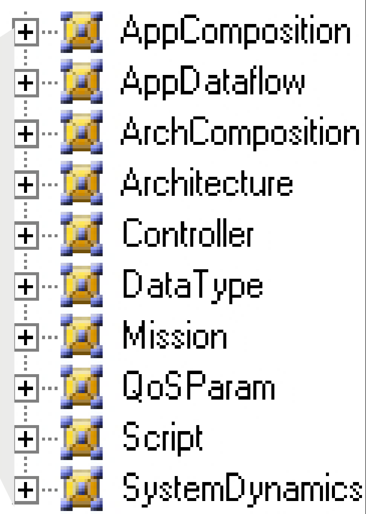
- The goal is to support the design of QoS adaptation separate from the functionality and at a higher level than programming
  - Simplify the design of QoS adaptation
  - Enable a larger class of practitioners
- Model and synthesize adaptive behaviors that
  - Ensure predictable, controlled behavior
  - Satisfy mission requirements (i.e., use requirements to choose suitable tradeoffs)
  - Graceful degradation and recovery

Essential Elements in QoS Adaptation

- **Mission requirements -** What does the system need to achieve? This is used to capture the high-level mission requirements of the system
- **Observable parameters -** Where is the system within the QoS space? Before any adaptation action can be taken, the system has to know its current state with regard to the QoS of interest
- **Controllable parameters and adaptation behaviors -** How can the system move through the QoS space? These are the knobs available to the application for QoS control and adaptation
- **System dynamics -** Where can the system move within the QoS space? Based on the current state of the system (observable parameters) and the set of knobs available (controllable parameters and adaptation behaviors), there could be several options for adaptation
- **Adaptation strategy -** Where should the system go within the QoS space? Here we actually choose an adaptation strategy, which specifies the adaptations employed and the tradeoffs made in response to dynamic system conditions in order to maintain an acceptable mission posture

Distributed QoS Modeling Environment (DQME)





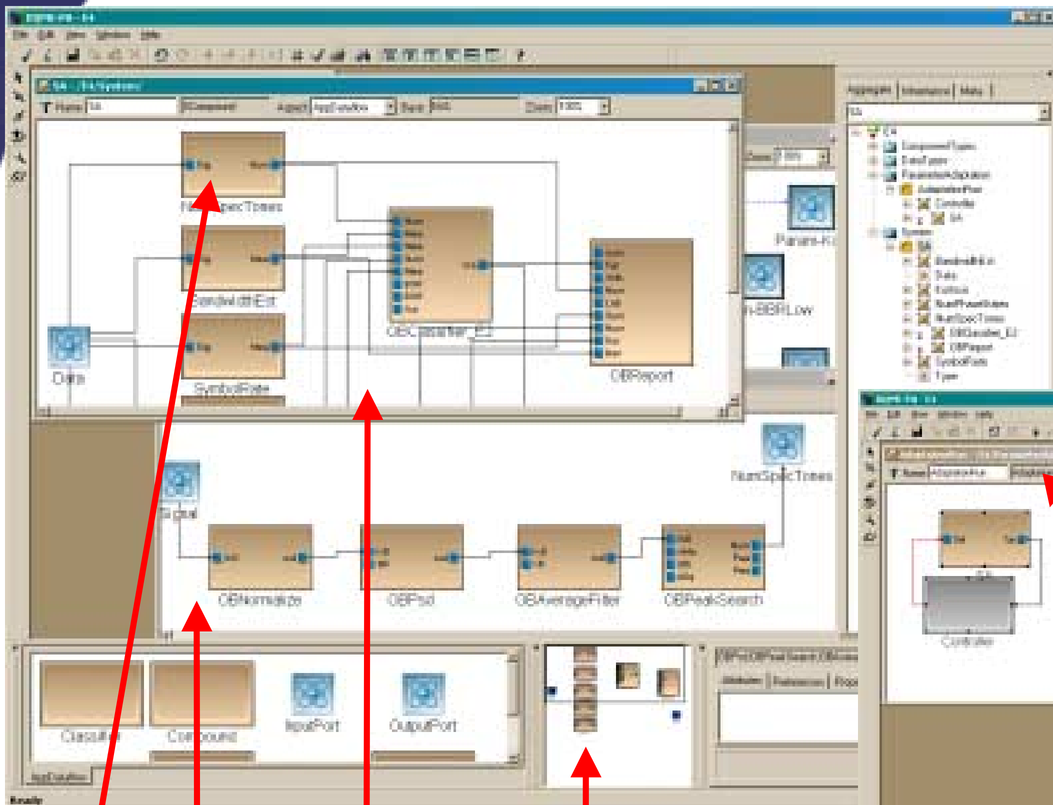
DQME Metamodel

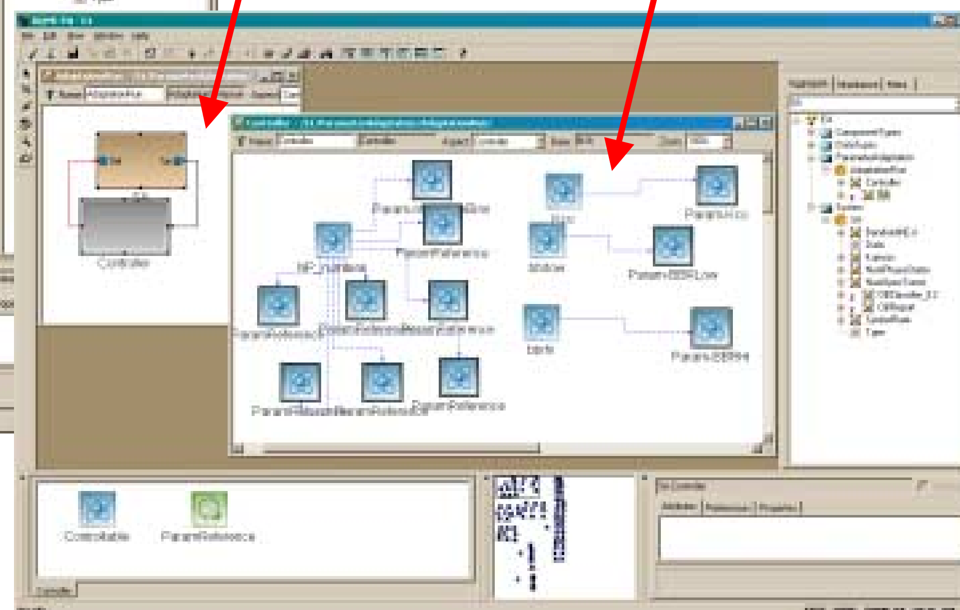
- Prototype DQME in the GME modeling environment captures all QoS essential elements identified:
  - Mission -> Mission requirements
  - QoSParam -> Observable and Controllable QoS parameters
  - SystemDynamics -> System dynamics
  - Controller->Adaptation strategies
- Works with a modeling language supporting Application Structure (SRML)
- Code synthesis through model interpreters

Application of DQME to Optimization of Signal Analyzers

- Our focus has been on the design stage optimization (vs. deployment stage) of signal analyzers, using adaptation and control to measure, predict, and tune parameter values for signals with known ground truth
  - Experiment 1 – Optimized parameters for PSK signal recognition (PSK, non-PSK)
  - Experiment 2 – Optimized parameters for PSK and FSK signal recognition (PSK, FSK, other)
  - Experiment 3 – Build an optimized signal analyzer for PSK, FSK, and multi-carrier PSK
- Signal Analyzer Architecture
  - Build on feature extractors - yield feature values that are useful for discriminating between different signal types and identifying signals of interest
  - OpBlocks - low-level signal processing operations in feature extractors
  - A Classifier - uses the feature values to identify the class of the incoming signal
  - OpBlocks and the Classifier itself can be parameterized and their values can be adjusted to improve the classification of the signal analyzer.

Signal Analyzer Model in DQME





Search space “controller” – tweaks parameter values systematically, checks against ground truth, and computes utility

Parameters and their references into OpBlocks

Feature Extractor

Structure of Signal Analyzer

“Defender” View (thanks GME 4!)

Structure of Feature Extractor – constructed of OpBlocks

QoS Modeling

Classification Outcome Definitions

True Signal Type	Classification Output		
	PSK	FSK	NOTA
PSK	Hit	FalseAlarm	Miss
FSK	FalseAlarm	Hit	Miss
Other	FalseAlarm	FalseAlarm	Reject

Cost Factor for Classification Outcomes

Classification Outcome	Cost
Hit	0
Miss	1
Reject	0
FalseAlarm	1

Therefore, the QoS can be defined as:

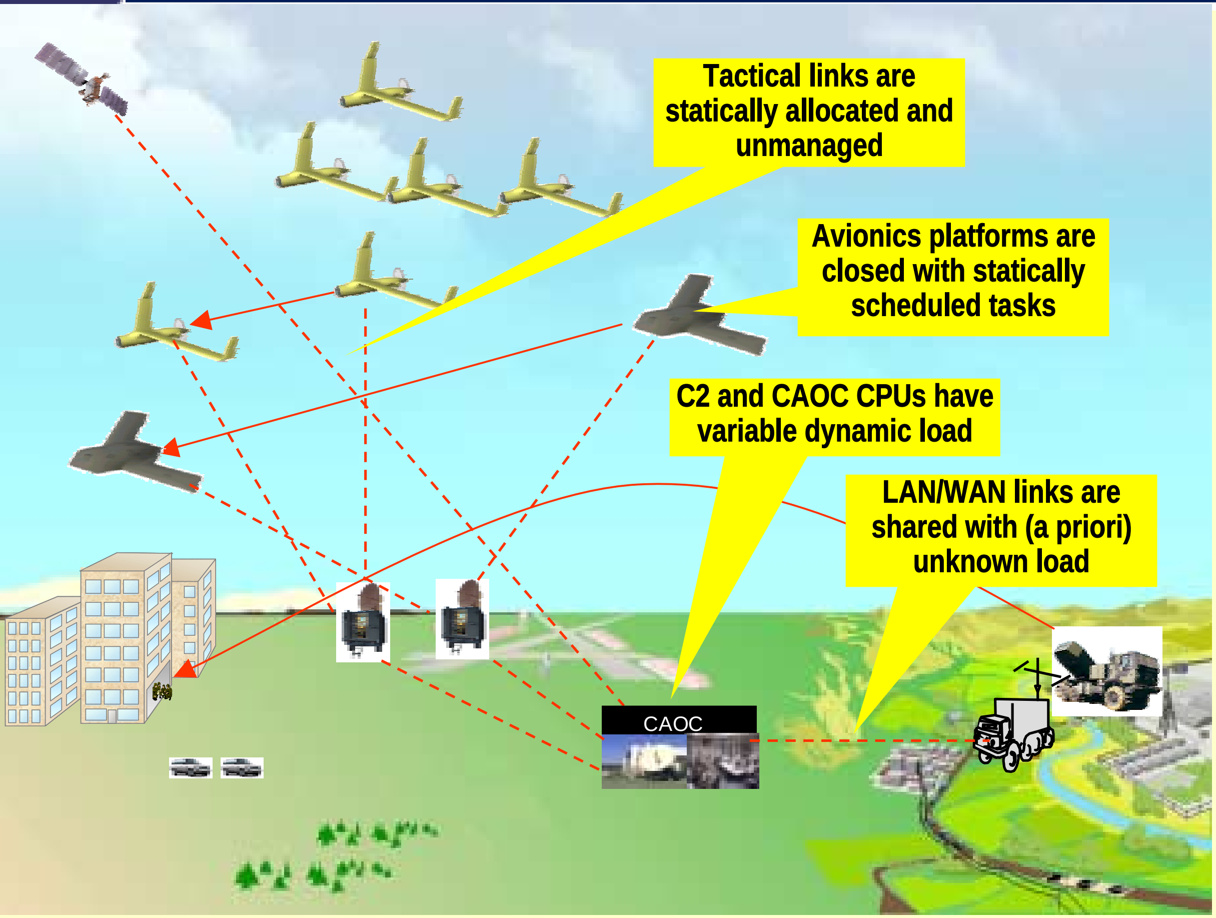
$$QoS = 1 - \tfrac{1}{N}\sum_{i=1}^{N} v_i$$

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

Signal Analyzer Optimization Results

Experiment	Problem	Measure of QoS	Experiment Parameters	What Experiment Evaluated	Metrics (Qualitative/ Quantitative)	Significant Results
1	Recognize PSK signals	Percent of correct recognition	non-optimized algorithms and initial parameter values	Improvement in QoS after parameter tuning using local search	3 parameters tuned Total 540 searches Initial QoS: .750 Final QoS: .971	**10x reduction in error** (incorrect classification reduced from .25 to 0.029)
2	Recognize PSK and FSK signals	Percent of correct recognition	non-optimized algorithms and initial parameter values	Improvement in QoS after tuning; speedup of global search + grouping; tuning of more params	Initial QoS: .6875 4 params (ls): 1620 searches, final QoS: .8819 4 params (gs): 93 searches, final QoS: .845 10 params (gs): 429 searches, final QoS: 0.8608	**2.2x reduction in error** (error reduced from .3125 to .1392) **2.5x parameters in ½ the time**
3	Identify PSK, FSK, multi-carrier PSK	Percent of correct recognition	system parameters with **near optimal** values	Reduction in manual tuning effort Improvement in QoS Ease of use	3 parameters tuned 9 hours labor to tune by hand; 2.25 labor to tune using tool Initial QoS: 89.12% Final QoS: 89.22% Some usage issues identified	**4x reduction in manual labor** **1% improvement over optimized values**

Application of DQME to the PCES Capstone Demonstration



Role-defined requirements and tradeoffs (e.g., rate, image size, fidelity)

Heterogeneous, shared, and constrained resources

Changing modes, participants, roles, and environmental conditions

End-to-End Mission-Driven QoS Management

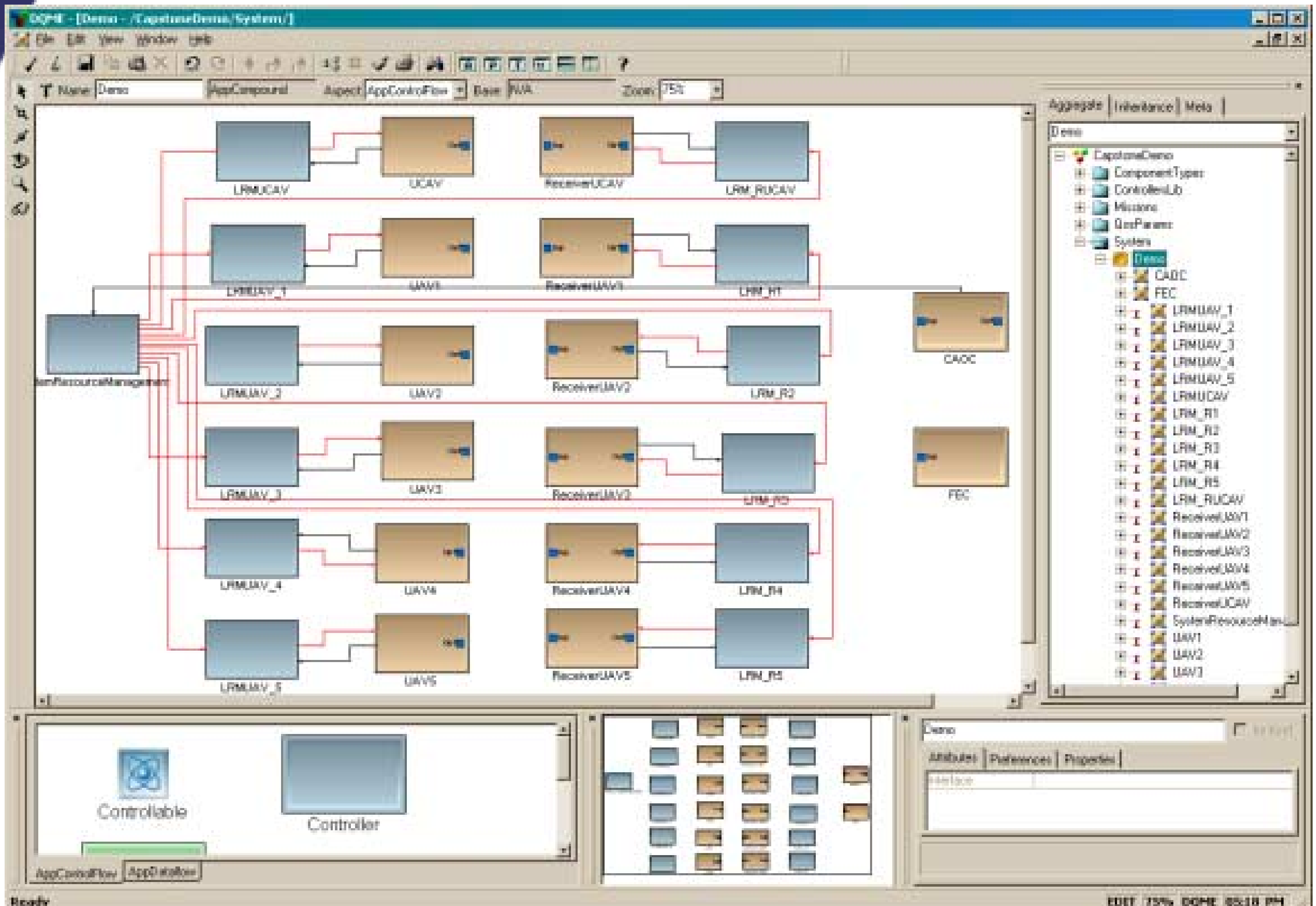
Surveillance
- Maximize surveillance area
- Sufficient resolution in delivered imagery to determine items of interest

Target Tracking
- UAV observing target provides high resolution imagery so that target or threat identification is possible

Battle Damage Assessment
- UCAV must provide high resolution imagery until a human operator has determined that it is sufficient
- UAV over target area must continue to provide target acquisition and engagement mission

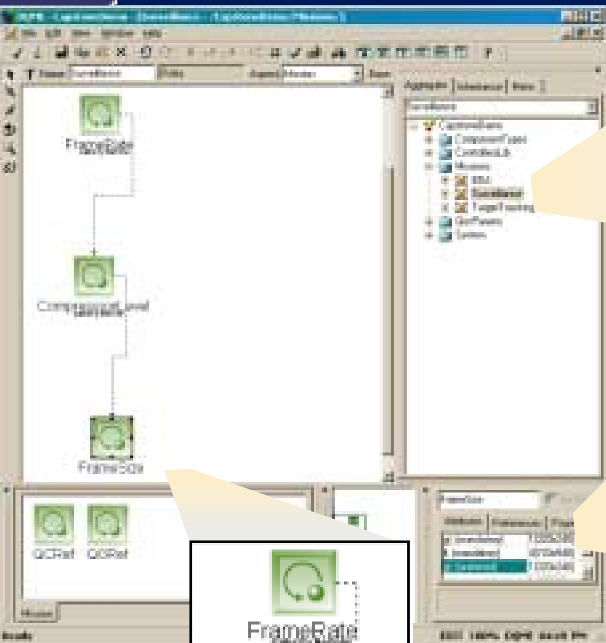
The challenge is to program the dynamic control and adaptation to manage and enforce end-to-end QoS.

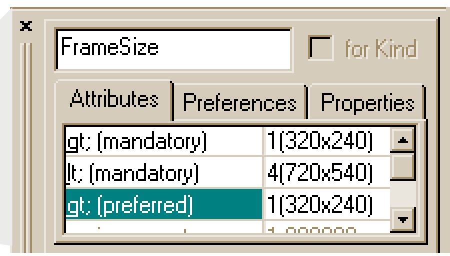
Component Structure of the PCES Capstone Demonstration DQME Model

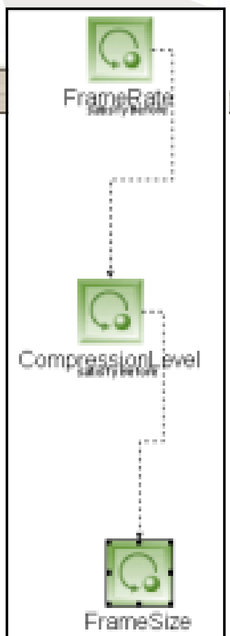


- Five RUAVs
- One UCAV
- CAOC
- Fire Effects Cell (Army Command)
- Local resource managers (LRMs)
- A System Resource Manager (SRM)

Mission Requirements of the PCES Capstone Demonstration







- Three participant roles:
  - *Surveillance*
  - *Target Tracking* – Observing an area of interest
  - *Battle damage assessment* (*BDA*)
- Each role specifies minimums and maximums
  - Minimum rate in *surveillance* must ensure no gaps in surveillance (function of the UAV camera scan size, altitude and speed of the UAV)
  - Minimum size (360x240) and resolution determined by domain experts
- Tradeoffs within roles
  - *Surveillance* adjusts rate of imagery, resolution (i.e., compression), followed by size
  - *Target Tracking* adjusts rate and allows cropping (if centered on area of interest)
  - *BDA* adjusts rate, size, and allows tiling of images

Observable and Controllable Parameters and Adaptation Behaviors

Observable and Controllable Parameters

- QosParams
  - Bandwidth
  - CPUReservationLevel
  - CompressionLevel
  - FrameDataSize
  - FrameRate
  - FrameResolution
  - FrameSize
  - Importance
  - NumOfRSUV
  - NumOfSUAV
  - NumOfUCAV
  - Tiles

Adaptation Behaviors

We have a set of off the shelf encapsulated behaviors (*qoskets*)

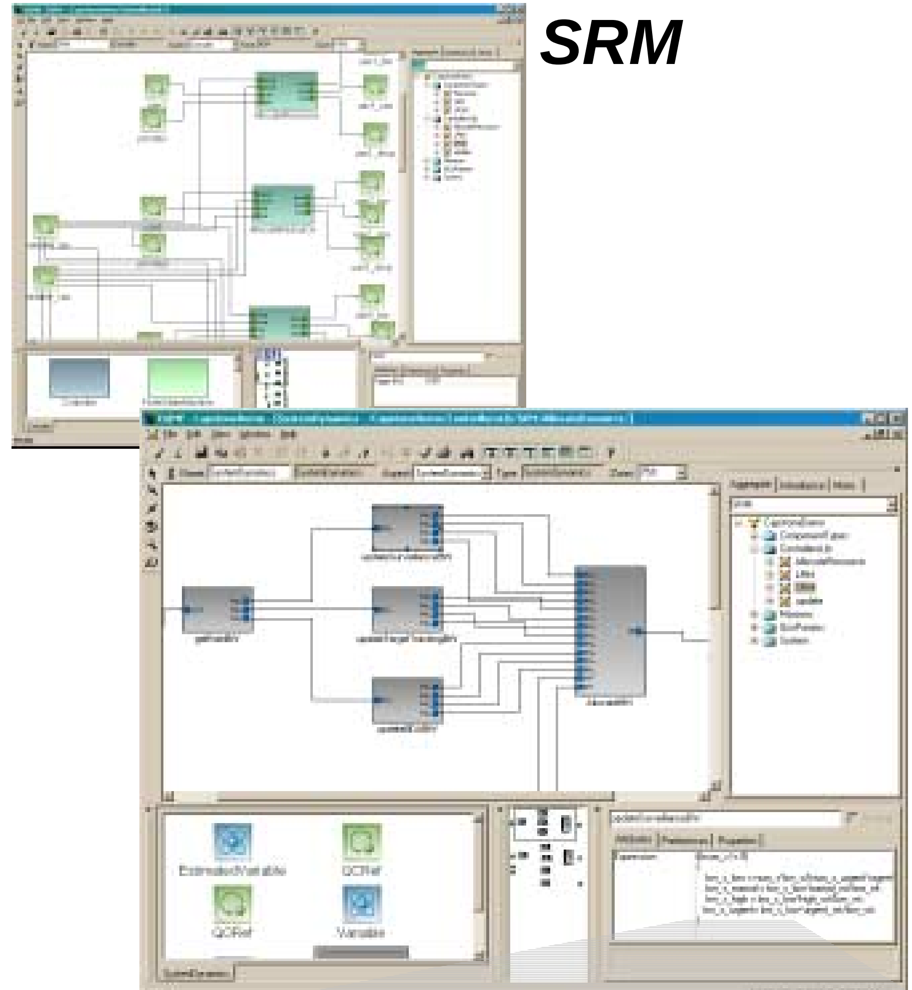
- CPUQoS
- CompressionQoS
- CroppingQoS
- DiffServQos
- FrameRateQoS
- FrameSizeQoS
- TilingQoS

- CPU Broker (Univ. Utah)
- Compression
  - Wavelet, JPEG, PNG
- Cropping
- Setting Diffserv Codepoints
- Pacing (changing frame rate)
- Scaling
- Tiling

Multi-Layered Control

- System Resource Manager
  - A supervisory controller that responds to changes in role and relative priority within a role
  - Allocates available resources to participants
  - Provides *policy* of role, relative priority, bandwidth allocation, and CPU allocation to participants (UAVs, UCAV)
- Local Resource Managers
  - Determine how to effectively use the resources allocated to a participant to perform its role
  - Select an appropriate CPU reservation, network priority, and data manipulation (rate, compression, size) to fit the allocated resources and fulfill its role
  - Monitors actual resource usage and adjusts parameters to keep usage within allocation
  - Bases decisions on allocated resources, tradeoffs (from mission requirements), available adaptation strategies, and system dynamics

SRM

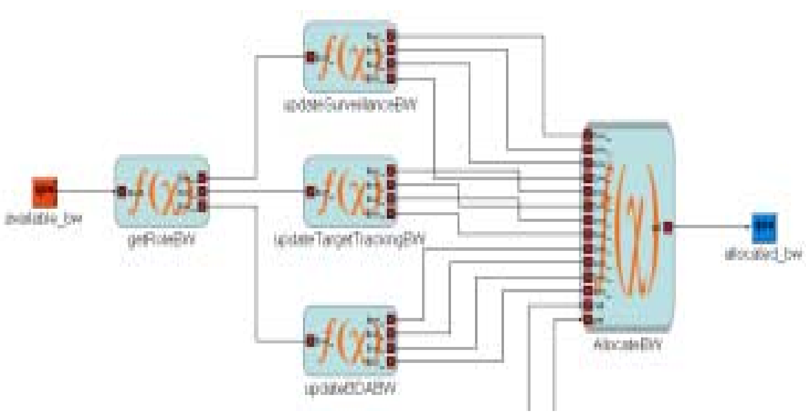


System Dynamics compute resource unit

```
bw_s_low = num_s*bw_s/(num_s_urgent*urgent_wt + num_s_high*high_wt
+num_s_normal*normal_wt + num_s_low*low_wt)*low_wt;
 bw_s_normal = bw_s_low*normal_wt/low_wt;
 bw_s_high = bw_s_low*high_wt/low_wt;
 bw_s_urgent= bw_s_low*urgent_wt/low_wt;
```

PCES Capstone Demonstration Code Generation

- System-level resource allocation strategies are captured in SRM
- Model interpreter scans and extracts information from the model
- Generate equivalent C++ code that implements the logic in the model
- Tested and successfully used in SRM CIAO component in the capstone demo application
- We are working on generating more code, including the LRM



Logic in SRM model

```
void SystemResourceManagement::AllocateResource(
          PCES_UAV::Priority priorityValue,
          PCES_UAV::Role roleValue,
          double available_bw,
          double available_cpu,
          double& allocated_bw,
          double& allocated_cpu,
          long& code_point)
ACE_THROW_SPEC(( CORBA::SystemException )){
  AllocateDSCP(code_point,roleValue);
  getRoleBW(available_bw);
  getRoleCPU(available_cpu);
  updateBDABW();
  updateSurveillanceBW();
  updateTargetTrackingBW();
  updateBDACPU();
  updateSurveillanceCPU();
  updateTargetTrackingCPU()
  AllocateBW(allocated_bw,roleValue,priorityValue);
  AllocateCPU(allocated_cpu,roleValue,priorityValue);
}
```

Generated SRM code

BBN TECHNOLOGIES

Conclusions

- We have designed, built and evaluated significant elements of a QoS adaptive modeling environment
  - Developed domain specific, semantically rich modeling language (DQME) that allows QoS adaptation modeling in distributed, real-time, embedded systems
    - Identified essential elements of dynamic QoS adaptation
    - Demonstrated the ability to model QoS in different application domains
  - Code generation capability through GME interpreters and code-level QoS adaptation support through QuO middleware
    - Simulation code (MATLAB) for signal analysis
    - Runtime code (C++, CORBA, QuO) for both signal analysis and PCES capstone demo
  - Works with other tools in a combined tool chain, including SRML and CADML
- Generally made QoS adaptation design easier and provide productivity/quality improvement
- Remaining difficult problems
  - User interface issues, making modeling in DQME easier
  - Extraction of patterns of use