Cover page for the 2002 Command and Control Research and Technology Symposium
June 11-13, 2002, Naval Postgraduate School, Monterey, CA

Possible focus topic:

C2 Experimentation

Paper Title:

Analyzing Quality of Service Specification Through System Event Trace

Author:

John Drummond
Space and Naval Warfare Systems Center, San Diego
Code D4121
Spawarsyscen
53560 Hull Street
San Diego Ca 92152-5001
Phone (619)553-4131
Fax (619)553-4808
drummond@spawar.navy.mil

Analyzing Quality of Service Specification Through System Event Trace

John Drummond[*]
Space and Naval Warfare Systems Center, San Diego
Code D4121
Spawarsyscen
53560 Hull Street
San Diego Ca 92152-5001
(619)553-4131
drummond@spawar.navy.mil

Abstract

The distributed system presents an enigmatic set of requirements to the software engineer. The added complexity of command & control constraints can coalesce to represent an environment that can soon overwhelm many distributed command & control software development efforts. These conditions are especially acute when multiple competing applications are required to share the environment system resources. Software development efforts that have been targeted at the distributed command & control environments have focused upon the approach of providing adequate quality of service levels to the requesting applications for mitigation of the resource sharing dilemma. Providing efficient resource utilization can satisfy many of the quality of service issues, which present difficulties in resource sharing. However, the quality of service analysis methods currently in place to determine efficient resource utilization are either to narrowly focused upon specific resource managers/controllers or are not sufficiently equipped to provide a detailed dynamic examination during application/system execution. Therefore, this paper presents a comprehensive method to achieve a viable resource utilization analysis based upon specific dynamic quality of service events.

[*] **This work sponsored by the Defense Advanced Research Projects Agency, Information Technology Office (DARPA-ITO)**

Introduction

The distributed processing environment provides added benefits over the non-distributed approach due to the capacity for improvement in program accessibility, overall performance, additional sharing of limited resources, and the increased fault tolerance capabilities. This distribution of the processing load does however increase the overall complexity of the environment. The processing of command & control elements also exhibits perplexing difficulties such as abrupt mission changes, and dynamic tactical surprises.

Despite this expanded complexity, the augmentation to command & control environments of distributed processing is nevertheless desired. However, this distributed command & control environment does present an expanded assemblage of requirements and constraints for the software engineer. The efficient allocation of system resources can be considered a major element of these requisites. This dynamic complexity that is found in the distributed command & control environment indicates that a determined level of service for the applications utilizing these distributed resources would be advantageous to assure mission critical processing.

Directly implementing the quality of service features into the distributed command & control infrastructure is not a trivial task. Additionally, subsequent to implementing the quality of service features, an examination must be performed upon the effectiveness of the implementation. To properly ascertain that the essential quality of service based system resources are being reasonably utilized and efficiently shared among the distributed command & control programs some evaluation of the resource distribution approach needs be conducted.

However, current analysis techniques for evaluation of quality of service specifications are somewhat lacking in that there is no exacting method to determine precisely what quality of service based resource inconsistencies take place during actual program execution. Therefore, the analysis of proper employment and dispersion of available resources is the focus of this research in the area of distributed command & control processing. This direct analysis has been carried out though the use of quality of service based behavioral models. The development characteristics of the behavior model are described in the next section.

Approach

The approach to this work is based upon the presumption that systems within a distributed end-to-end command and control path should support some form of negotiated quality of service levels through efficient employment of available resources to ensure proper execution of mission critical distributed command & control programs. This focus of efficient quality of service is a much needed element for current DoD systems as stated by the Defense Advanced Research Projects Agency Quorum program manager [Koob 99] “While emerging network-level QoS mechanisms (such as RSVP) are an essential enabling technology for Quorum, they are insufficient in that they are limited to communications QoS. Quorum defines "end-to-end" as being the quality-of-service seen by the application, which calls for coordinated QoS management across middleware, operating systems, and networks.”

This research work initially begins through an investigation of how a specific system can aid in the mitigation of resource utilization difficulties within end-to-end quality of service environments. This investigation is accomplished by an in-depth look at various quality of service and resource deployment characterizations as well as the application of high level modeling. The detailed analysis is attained through the utilization of the SPAWAR System Center DARPA Quorum Integration Test & Exploitation project (Quite) testbed environment located a the SPAWAR System Center. The domain illustration for the project testbed is shown in Figure 1. Domains Within The SSC-SD DARPA Testbed.

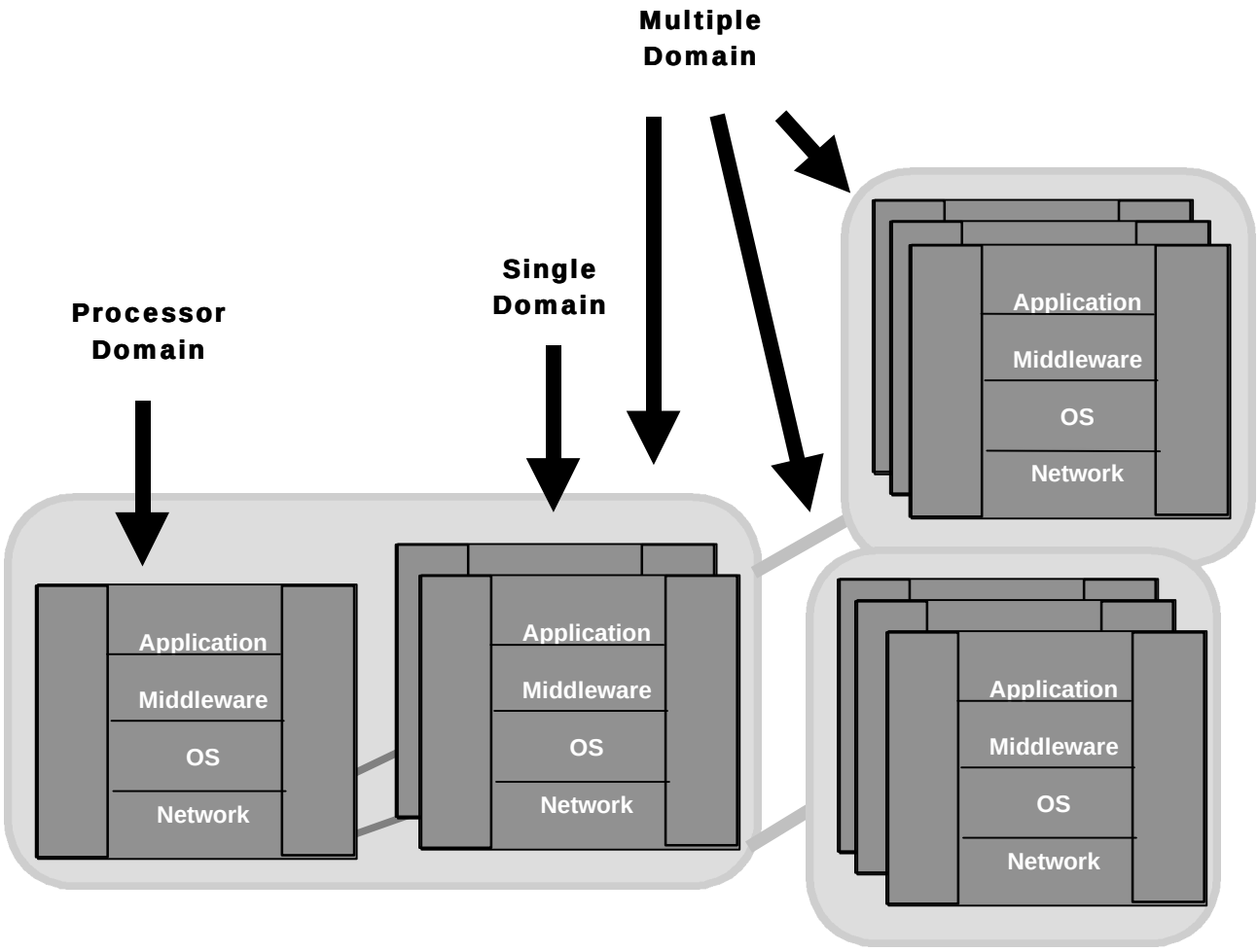

Figure 1. Domains Within The SSC-SD DARPA Testbed

To achieve a precise analysis of quality of service procedures an approach has been implemented to examine the exact quality of service execution path during program operation. The evaluation approach utilized in this research is based upon an event trace concept employed by [Auguston

00] originally as an analysis tool for focusing upon correctness in C language programs. This event trace concept discusses the idea that testing and debugging are mostly concerned with the program run-time behavior, and states that developing a precise model of program behavior becomes the first step towards any dynamic analysis.

The quality of service analysis domain of the targeted program for this implementation is centered upon the following characteristics: distributed computing environment, multiple heterogeneous systems, network medium connections, software applications with specific requirements (QoS resource needs), resource management software (with quality of service awareness), and metrics data gathering instrumentation software. The tactics to be pursued for this work have pursued the following approach sequence:

- Production of a program behavior model based upon quality of service factors. This will be implemented by developing abstractions for quality of service events based upon specific quality of service actions that occur during typical program execution which include:
    - Quality of service request statement execution which requests resource reservation within the Application software.
    - Specific procedure execution focused upon the evaluation and negotiation of available resources to be applied to the originating resource request.
    - Software statement execution of procedures for proper utilization of the assigned resources.
    - Execution of statements responsible for the detection of any resource needs change within the application software.
    - Execution of procedures focusing upon the re-negotiation based on increase or decrease of available and previously assigned resources.
    - Execution of reallocation statements for specific resources by the resource controller software.
    - Sending and receiving of quality of service related messages by both the application and resource controller software.

- Prepare quality of service specific application program points, which directly relate to appropriate resource utilization. Such elements will have direct consequences upon quality of service and for example will include:
    - QoS specific message passing
    - Application QoS violations
    - QoS negotiations
    - QoS resources and management of resources(RM)
    - QoS re-negotiations
    - QoS levels

- Instrument the targeted program based upon these previously identified specific quality of service program points. This direct invasive source code instrumentation will allow for

effective event trace recording at the precise location of the quality of service actions of interest.

- Use this analysis to develop an overall characterization of distributed command & control systems for any mitigation of discovered quality of service related efficiency problems.

The specific logical conditions and constraints for this work include a distributed system, heterogeneous environment, multiple diverse quality of service levels essential for program execution (i.e. application requirements vary high/low needs), and available resources include network bandwidth, CPU, memory, etc. The informal abstraction of events mentioned above includes specific actions performed during program execution. The majority of these events will occur within specific application and resource management software. As stated earlier this approach focuses upon the analysis of quality of service associated events. Events are key to this research and are utilized as a basis for building program behavior analysis. Again the work of [Auguston 99] states, an event can occur when an action is attained while the program execution process is underway. This work of Auguston is primarily directed toward overall program improvements, which do not specifically cover quality of service issues. By extending this work into the resource utilization domain the "event trace" approach can be aptly applied to quality of service execution pathways.

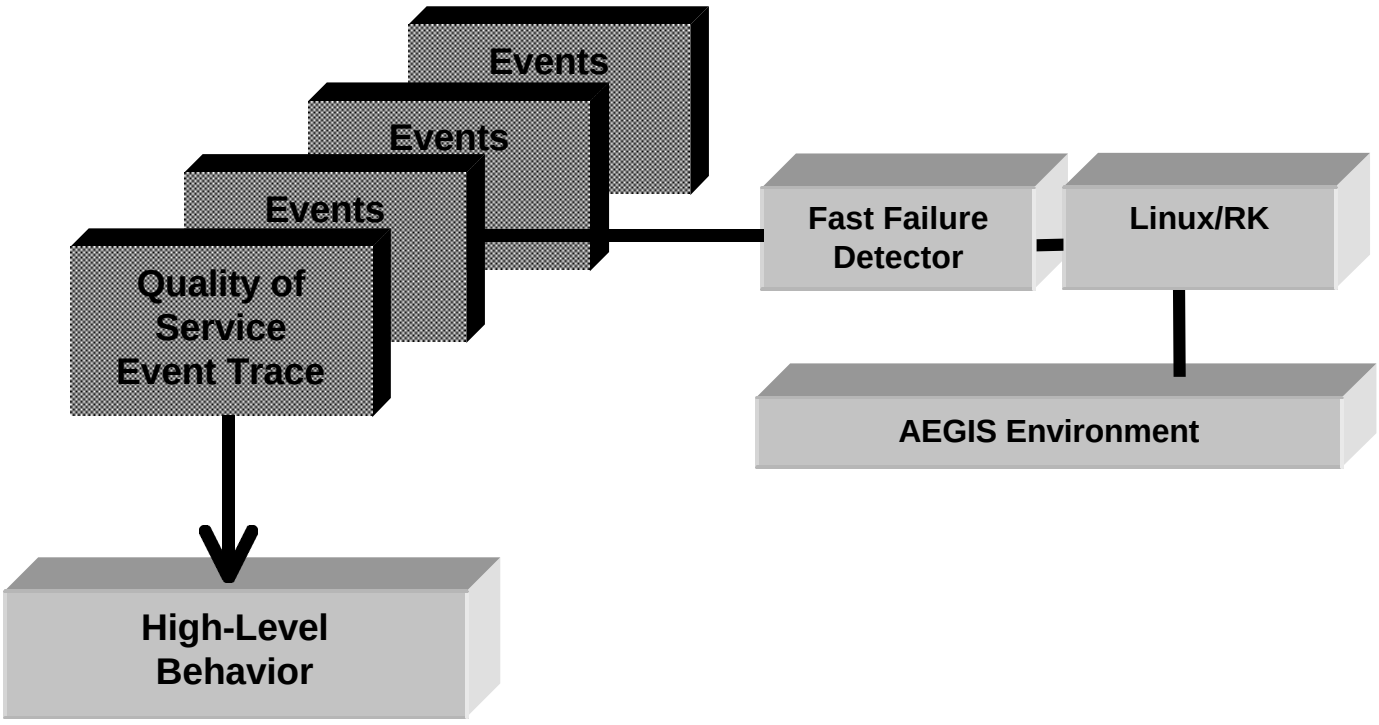

Figure 2. Target Program Analysis

The objective of this research comprises the development of accurate high-level behavioral models, which focus upon domain specific areas of a distributed environment. The essence of these characterizations concentrate upon the quality of service treatment of resources within this environment. To achieve this objective, a target program has been selected for dynamic analysis. This quality of service event trace has directly examined the fast failure detection application. The fast failure detection program has been implemented within the AEGIS testbed environment as illustrated in Figure 2.

This fast failure detection program was designed and developed under the DARPA-ITO Quorum Integration, Testbed and Exploitation (Quite) project efforts. The failure detection software was created within the Quite project testbeds, tested, experimented upon, and has ultimately been implemented within the Naval Surface Warfare Center Hiper-D AEGIS testbed. The primary objective of this failure detection software program is to efficiently and promptly detect failures within the group communication software utilized within distributed mission critical systems such as the AEGIS environment.

As noted in [Drummond 02] this program can be setup to take advantage of a resource management system based upon quality of service procedures or operate as a simple non-quality of service application "The Fast Failure Detector can be built and executed on its own or it can be executed while taking advantage of facilities like Linux/RK and Ensemble group communication." For the purpose of this event trace analysis research the Fast Failure Dector program has been implemented using both the Linux/RK resource kernel and the Ensemble group communication software. A diagram of the Fast Failure Dector program operations and the specific program elements are illustrated in Figure 3.

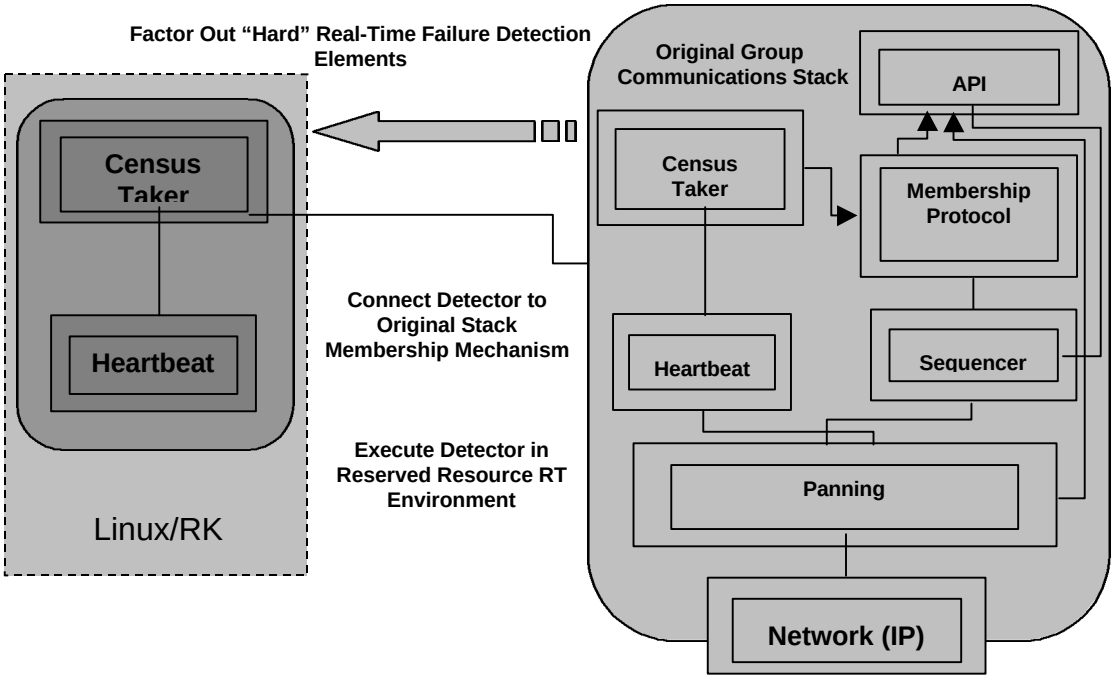

Figure 3. Fast Failure Detection Program

As illustrated in Figure 4 the event trace analysis effort essentially focuses upon the actions, which are fundamental to resource utilization. This includes all path length data as well as resource competition actions. During this event trace the failure detection program processes and threads compete for the resources that are being managed by the resource kernel. While dynamically executing two of three failure detection program threads continue their execution and specifically request distinct resources from the resource kernel. The third thread acquires resources from the parent process, which have previously been allocated from the resource kernel.

However, this resource competition can also extend to other concurrently executing applications, which request and in turn are provided with resources by the resource kernel. To intensify this competition a competing application has been executed simultaneous to the failure detection program. This competing application requests exceptional quantities of resources from the resource kernel. The explicit resources, which have been allocated by the resource kernel for the competing application after processing its requests, are also noted and recorded within the event trace. In turn this previous resource allocation to the competing application has the potential to force re-negotiations of the resources being requested by the failure detection program threads and main processes. Any denial of the initial resource request and any re-negotiation actions for these resources by the failure detection processes and threads are recorded within the event trace.

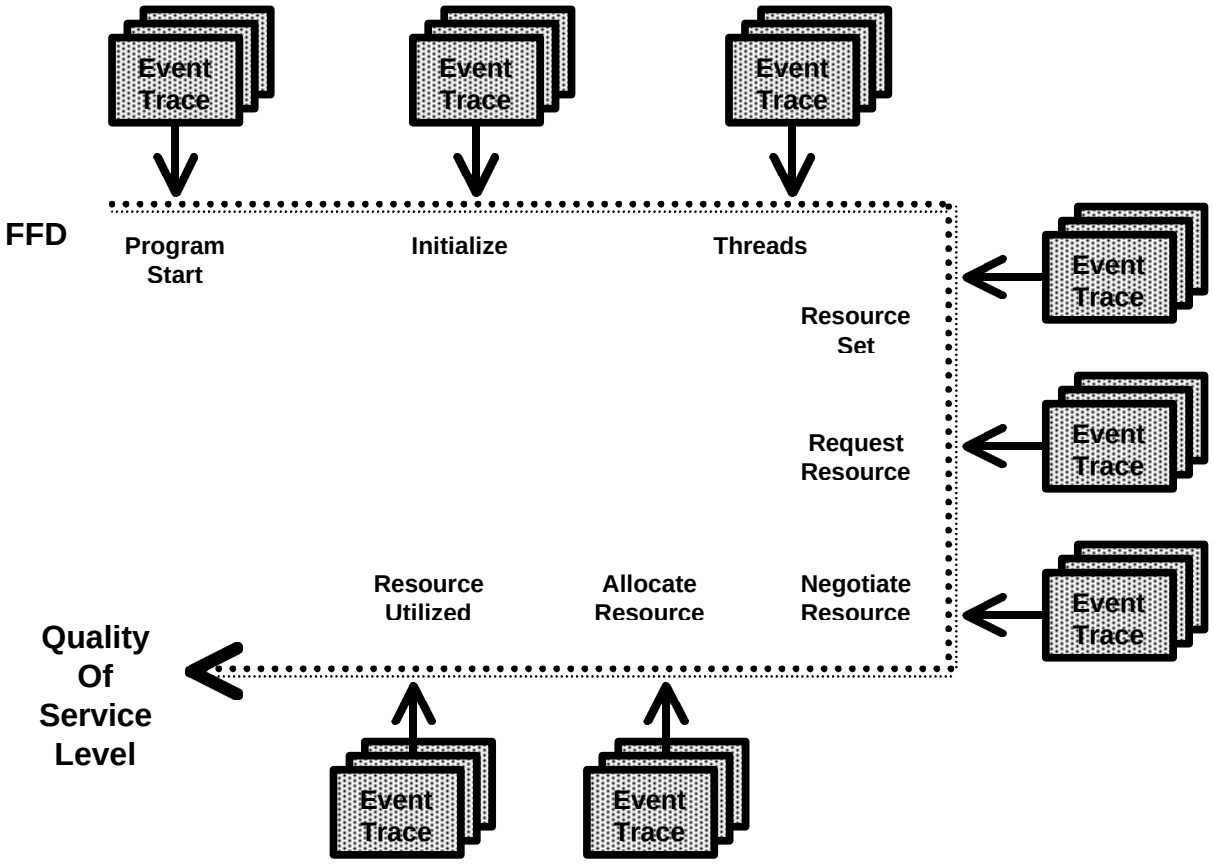

Figure 4. Event Trace Sequence

Conclusion

The concluding results of this examination have produced accurate high-level behavior representations of the fast failure detection program. This quality of service event trace analysis has shown the capacity to specifically reveal various exact failure points, potential resource re-negotiation inefficiencies, and lengthily quality of service path calls. All of these elements have a direct bearing upon the quality of service based resource management efficiency for the distributed command & control fast failure detection application program and environment.

The resulting findings from the quality of service event trace analysis include numerous elements, which have directly addressed the original goals and objectives of this research effort. The initial specific event trace analysis case has focused upon a precisely targeted failure detection program

within a distributed command & control environment, however it can also be extended to the general case. This utilization of the quality of service event trace analysis can be applied to other programs within the distributed command & control domain. These prospective targets of the quality of service event trace analysis would ideally be within the design and development phases of the software engineering process.

This work can also be used to provide support for the correct characterization of a software engineering design approach for distributed systems with respect to efficient employment of system resources. These research findings can also be directly applied to the design and development phases of the software engineering process as illustrated Figure 5. The approach of correctly obtaining effective quality of service resource levels has also effectively been characterized by this research and can further be applied to this software engineering design and development process. This is accomplished by utilization of the quality of service event trace to enhance quality of service awareness, which will in turn augment the creation of the program prototyping. In this way the quality of service event trace analysis can facilitate the creation of distributed command & control programs which have quality of service requirements.

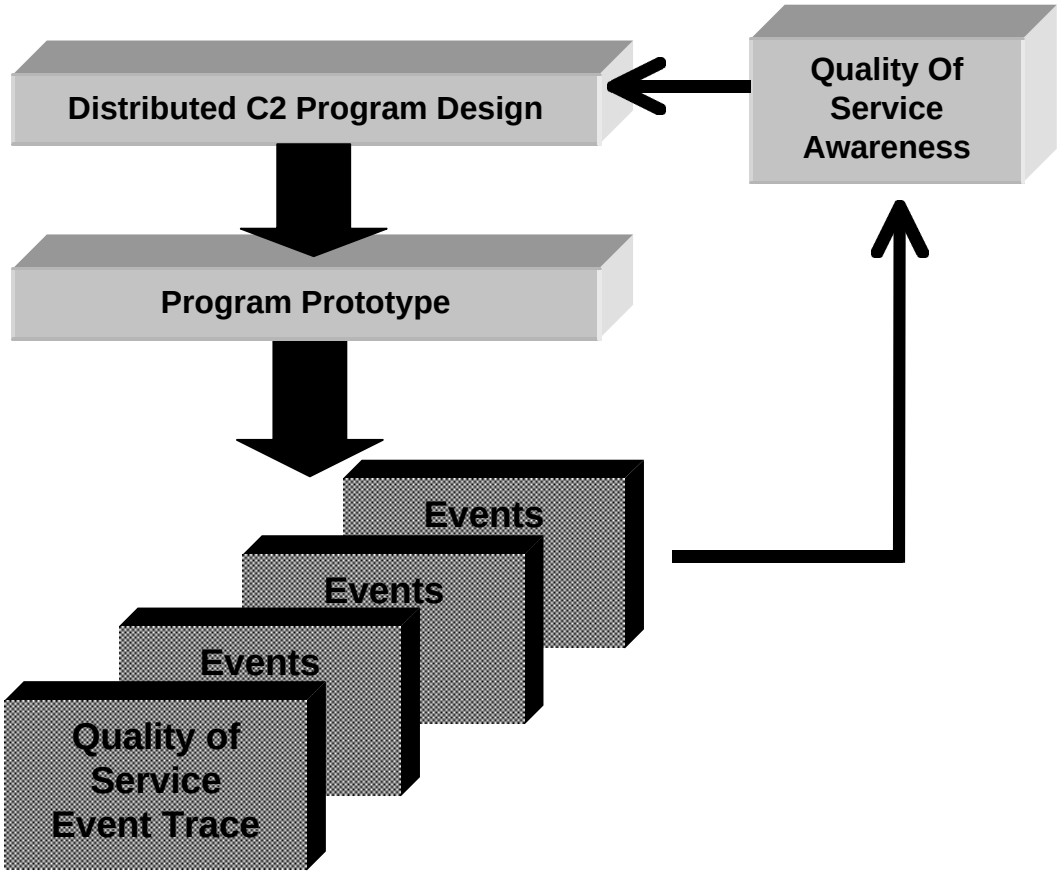

Figure 5. Quality Of Service Awareness

The event trace has previously been illustrated for employment in the design and development phases of the software engineering process. However, the quality of service event trace efforts

may also be effectively utilized in the implementation phase of the software engineering process, and can also assist in refinement of the overall requirements analysis for quality of service constraints as illustrated in Figure 6. During the implementation phase it is typically a complex task to ascertain the correct quality of service specifications that are to be utilized for a given distributed command & control program. For this method of employing the quality of service event trace analysis the results can be applied directly to the implementation of specific command & control applications. Any failures or potential for quality of service errors can be revealed prior to deployment of any mission critical applications which exhibit a priority need to maintain significant quality of service levels. The application of the quality of service event trace may also include legacy programs as targets, which have the capability for utilization of resource management controls and quality of service cognizance.

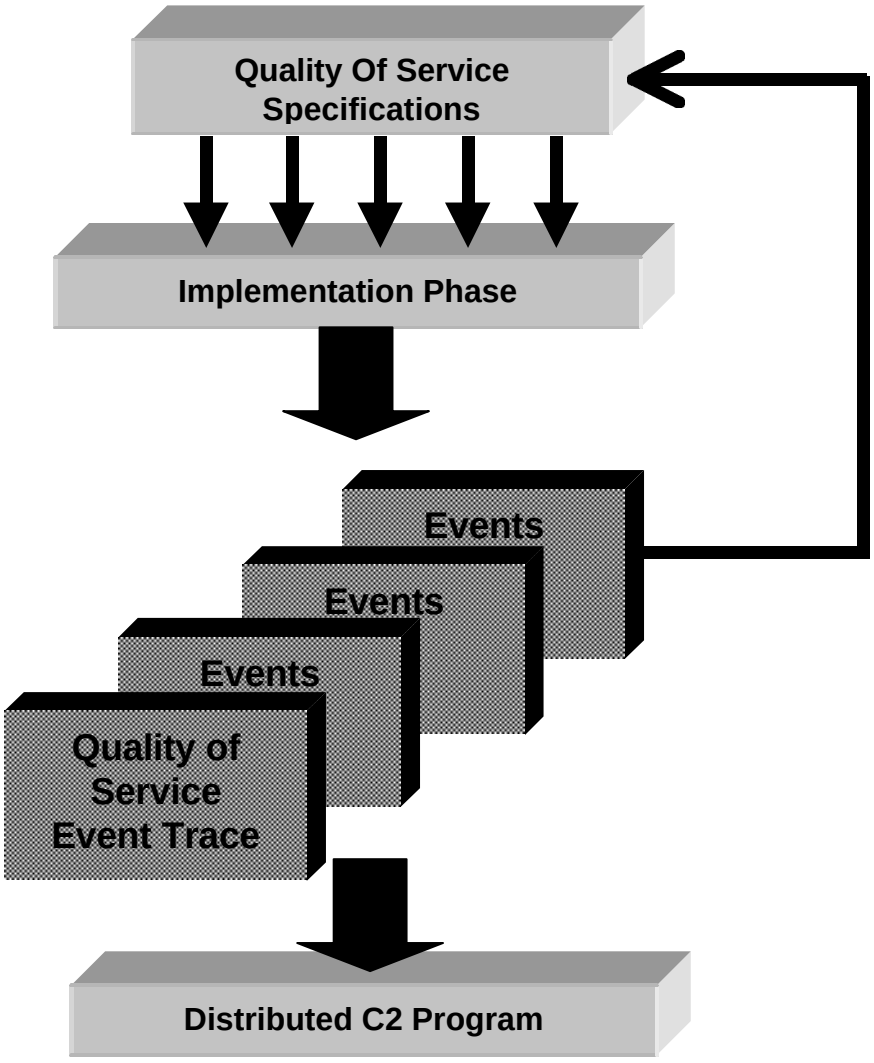

Figure 6. Distributed C2 Program Implementation

Thus far this work has specifically been directed towards the area of developing quality of service behavior models for targeted distributed command & control programs. This utilization of the quality of service event trace approach to behavioral modeling can be further expanded for inclusion into a development/analysis framework. Additionally as suggested in [Raje 2001] the possibility of integrating quality of service analysis into a larger framework or Unified Meta-component Model will provide even grater benefit for distributed environments, which utilize heterogeneous software components.

References

[Auguston 99] Auguston, M., *Tools For Program Dynamic Analysis, Testing, And Debugging Based Upon Event Grammars*, Technical Report, NMSU CSTR-9906, Department of Computer Science, New Mexico State University, New Mexico, June 1999.

[Auguston 00] Auguston, M., *Assertion Checker For The C Programming Language Based On Computations Over Event Traces*, Fourth International Workshop on Automated Debugging, AADEBUG2000, Munich, Germany, August 2000.

[Drummond 02] Drummond, J., Wells, D., Rahman, M., *Detecting Failure Within Distributed Environments*, SPAWAR Technical Paper, Space and Naval Warfare Systems Center, San Diego, Ca. 2002.

[Koob, 1999], Koob, G., *Background for DARPA-ITO Quorum Mission Statement,* Defense Advanced Research Projects Agency Information Technology Office, 1999.

[Raje, 2001], Raje, R., Auguston, M., Bryant, B., Olson, A., Burt, C., *A Unified Approach for the Integration of Distributed Hetrogeneous Software Components,* Monterey Workshop 2001, June 2001.