

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TESLA, INC.,
Petitioner

v.

INTELLECTUAL VENTURES II LLC,
Patent Owner.

IPR No. 2025-00341

U.S. Patent No. 7,181,743
Filing Date: May 24, 2001
Issue Date: February 20, 2007

**DECLARATION OF DR. SANJAY RANKA IN SUPPORT OF
PATENT OWNER'S PRELIMINARY RESPONSE**

Table of Contents

I.	Introduction And Qualifications	3
II.	Summary Of Materials Reviewed And Considered	8
III.	Legal Standards.....	10
IV.	Level Of Skill in the Art and Perspective Applied in This Declaration	12
V.	Claim Construction	15
VI.	The ‘743 Patent	15
VII.	The Claims 10, 11, 14, 16 and 18 Of The ‘443 Patent Would Not Have Been Obvious.	29
A.	Overview of Welch.....	29
B.	Overview of Ravindran.....	35
C.	Overview of Hosokawa	37
D.	Overview of Welch 1996 (Ex 2006).....	38
E.	Ground 1: Claim 10 Would Not Have Been Obvious In View of Welch alone, or Welch in combination with Ravindran... 42	
1.	[10.2] a second function group which provides ... event correlation capabilities;	42
2.	[10.4] wherein the third function group comprises: a first function which determines a state and health of the N hosts, a network operatively coupling the N hosts to one another and the M managed characteristic application computer programs in the distributed environment.....	53
3.	[10.5] a second function which determines required allocation and reallocation actions needed to maintain a plurality of Quality of Service (QoS) requirements established for the M managed characteristic application computer programs, the QoS requirements dictating parameters regarding service quality of the M management characteristic application programs; and	59

4.	[10.7] a fourth function group which provides program control capabilities permitting starting, stopping, and configuring of selected ones of the M managed characteristic application computer programs on respective ones of the N hosts in the resource managed system, ...	62
5.	Ravindran does not cure the shortcomings of Welch..	68
F.	Ground 2: Claim 10 Would Not Have Been Obvious In View of Welch in combination with Ravindran and Hosokawa.....	69
VIII.	Conclusion.....	71

I. Introduction And Qualifications

1. I, Sanjay Ranka, submit this declaration on behalf of Intellectual Ventures II LLC, (“Patent Owner”) to provide my opinions concerning the validity of claims 10 (the “Challenged Claims”) of U.S. Patent No. 7,181,743 (the “ ‘743 patent,” Ex.1001), in support of Patent Owner’s Preliminary Response to the Petitioner for *Inter Partes* review of the ‘743 patent, which I understand was filed by Tesla Inc., (“Tesla”, the “Petitioner”) and accompanied by a Declaration of Dr. Erez Zadok (the “Zadok Declaration”), among other exhibits.

2. I am more than eighteen years of age, and I am a citizen of the United States, currently residing in the State of Florida.

3. In formulating my opinions, I have relied upon my knowledge, training, and experience in relevant art. My qualifications, along with a more detailed summary of my background and publications, are stated more fully in my *curriculum vitae*, which has been provided as Exhibit 2002. Here, I provide a concise summary of my qualifications.

4. I am a Distinguished Professor in the Department of Computer Information Science and Engineering at the University of Florida.

5. Between 1999 and 2002, I served as Chief Technology Officer at Paramark, located in Sunnyvale, CA. During this tenure, I led the development of a real-time optimization system known as PILOT (Paramark Interactive and

Learning Optimization Technology), designed specifically for marketing campaign enhancement. PILOT achieved the delivery of over 10 million optimized decisions daily by 2002 using a parallel and distributed system, maintaining an operational uptime of 99.99% or higher. The PILOT system incorporated proprietary algorithms that continuously refined marketing campaigns through real-time data collection and performance analysis. The PILOT technology represents a significant advancement in adaptive marketing optimization based on dynamic data-driven decision making. Paramark's technological innovation was recognized by VentureWire and Technologic Partners, ranking the company among the Top 100 Internet technology firms in both 2001 and 2002, culminating in its acquisition in 2002.

6. I have also held positions as a tenured faculty member at Syracuse University, as an academic visitor at IBM, and as a summer researcher at Hitachi America Limited.

7. I have coauthored one book, five monographs, and 350+ journal and refereed conference articles. I have been recognized as a fellow of the Institute of Electrical and Electronics Engineers (IEEE), and the American Association for the Advancement of Science (AAAS) for my contributions to parallel and distributed computing. I was also a past member of International Federation for Information Processing (IFIP) Committee on System Modeling and Optimization. I won the

2020 Research Impact Award from IEEE Technical Committee on Cloud Computing. I was awarded the 2022 Distinguished Alumnus Award from Indian Institute of Technology, Kanpur.

8. I am currently an associate editor for Sustainable Computing: Systems and Informatics, Knowledge and Information Systems, and International Journal of Computing. Additionally, I am a book series editor for CRC Press for Bigdata. In the past, I have been an associate editor for IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, ACM Computing Surveys, and IEEE/ACM Transactions on Computational Biology and Bioinformatics. I was an associate editor-in-chief of the Journal of Parallel and Distributed Computing for the last five years. I was a general co-chair for the International Conference in Data mining in 2009, the International Green Computing Conference in 2010 and the International Green Computing Conference in 2011, a general chair for ACM Conference on Bioinformatics and Computational Biology in 2012, and a program chair for 2013 International Parallel and Distributed Processing Symposium and 2015 High Performance Computing Conference. I was a co-general chair for DataCom 2017 and co-program chair for ICMLDS 2017 and 2018. I was a program co-chair for the International Conference in Data Mining for 2022. My work has received 17500+ citations with an h-index of 65 (based on Google Scholar).

ScholarGPS awarded me with 2024 Highly Ranked Scholar status for my contributions to parallel computing.

9. I am a named inventor on several patents and published patent applications corresponding to the areas of my professional work. Accordingly, I have a general understanding of the U.S. patent prosecution process and of the novelty and non-obviousness requirements for patentability. Example patents include:

- United States Patent Number 8,805,715. Brian Jones and Sanjay Ranka. Method for Improving the Performance of Messages Including Internet Splash Pages, August 12, 2014.
- United States Patent Number 8,386,315. Aravind Bala, Richard Chatwin, Brian Jones, Ismail Ahmet Nalcacioglu, and Sanjay Ranka. Yield Management System and Method for Advertising Inventory, February 26, 2013.
- United States Patent Number 8,260,663. Sanjay Ranka, E. Diane Chang, and Daniel Veiner. Method, Algorithm, and Computer Program for Targeting Messages Including Advertisements in an Interactive Measurable Medium, September 4, 2012.
- United States Patent Number 7,415,423. Sanjay Ranka, Jason Lenderman, and James Weisinger. Method, Algorithm, and Computer Program for

Optimizing the Performance of Messages Including Advertisements in an Interactive Measurable Medium, August 19, 2008.

- United States Patent Number 7,406,434. E. Diane Chang, Richard Chatwin, Sachin Kumar, Sanjay Ranka, James Weisinger, and Jason Lenderman. System and Method for Improving the Performance of Electronic Media Advertising Campaigns through Multi-attribute Analysis and Optimization, July 29, 2008.

10. I believe that my extensive industry experience and educational background qualify me as an expert in the relevant field of parallel and distributed computing systems, including real-time systems with QoS as described in the '743 patent and related art. An example publication of a related paper that I co-authored is "J. In, P. Avery, R. Cavanaugh and Sanjay Ranka, Policy based scheduling for simple quality of service in grid computing, *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, Santa Fe, NM, USA, 2004."

11. I am knowledgeable of the relevant skill set that would have been possessed by a hypothetical person of ordinary skill in the art ("POSITA") at the time of the claimed invention of the '743 patent, which I have treated as May 25, 2000 for purposes of this proceeding. I have the necessary expertise to provide my

informed opinions in this IPR, and I am an expert in the art of the '743 Patent, due to the following:

- I received my undergraduate, master's, and doctoral degrees in computer science and engineering, computer and information science, and computer and information science, respectively;
- I have extensive experience in the parallel and distributed computing. I have been conducting state-of-the-art research in use of parallel, distributed and cloud computing for a variety of disciplines including scientific computing, transportation, and health care for the last three decades;
- I have been recognized by IEEE and AAAS as a fellow for my contributions in parallel and distributed computing;
- I have extensive experience in software design and development in both academia and industry; and
- I have supervised 50+ doctoral students in computer science and related fields during the last 30+ years and many of these students are currently employed by companies such as Facebook, Amazon, and Microsoft.

II. Summary Of Materials Reviewed And Considered

12. The opinions contained in this Declaration are based on the documents I reviewed and my knowledge and professional judgment. In forming

the opinions expressed in this Declaration, while drawing on my experience, I reviewed the following documents:

	Description
Paper 2	Petition For Inter Partes Review
EX1001	United States Patent No. 7,181,743 B2 (“the ‘743 Patent”)
EX1002	Prosecution history of the ‘743 patent (‘743 File History)
EX1003	Declaration of Dr. Erez Zadok (“Zadok Decl.”)
EX1005	“ <i>Distributed, Scalable, Dependable Real-Time Systems: Middleware Services and Applications.</i> ” Lonnie R. Welch et al., Proceedings 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, April 1999 (“Welch”)
EX1006	“ <i>Specification and Modeling of Dynamic, Distributed Real-time Systems.</i> ” Lonnie R. Welch et al. IEEE Real-Time Systems Symposium, 72-81, IEEE Computer Society Press, Dec. 1998 (“Ravindran”)
EX1007	U.S. Patent No. 6,088,727 to Hosokawa et al. (“Hosokawa”)
EX2006	“Challenges in engineering distributed shipboard control systems.” Welch, Lonnie R., et al. <i>The IEEE Real-Time Systems Symposium</i> . 1996
EX2007	L. R. Welch, B. Ravindran, B. Shirazi and C. Bruggeman, “Specification and Modeling of Dynamic, Distributed Real-Time Systems,” in Proceedings of the 19th IEEE Real-Time Systems Symposium, 72-81, IEEE Computer Society Press, 1998.

My opinions are additionally guided by my appreciation of how a person of ordinary skill in the art (POSITA) would have understood the claims of the '743 patent at the time of the invention, which I have been asked to assume is May 25, 2000. For purpose of this declaration, I will use the definition of POSITA as provided by Dr. Zadok (Ex 1003) and described in Section IV.

III. Legal Standards

13. I am not an attorney. However, I have been informed by counsel of certain legal standards and have applied those standards, described below, in formulating my opinions.

14. I understand that the claims of a patent define the limits of a patentee's exclusive rights. I understand that to determine the scope of the claimed invention, courts may construe claim terms, the meaning of which the parties may dispute. I understand that claim terms should generally be given their ordinary and customary meaning as understood by a person of ordinary skill in the art at the time of filing of the patent application.

15. I understand that claims must be construed in light of and consistent with the intrinsic evidence. In this context, I understand that intrinsic evidence includes the claims themselves, the written disclosure in the specification, and the patent's prosecution history. I understand that the specification is always highly

relevant to the claim construction analysis and often is the single best guide to the meaning of a disputed term. I understand that extrinsic evidence may also be considered when construing claims and may include, for example, technical dictionaries, technical publications and books, treatises, and expert testimony, though the intrinsic record will be more persuasive as to the meaning of a particular term.

16. I understand that one or more prior art references can render a patent claim obvious to a person of ordinary skill in the art if the differences between the subject matter set forth in the patent claim and the prior art are such that the subject matter of the claim would have been obvious at the time the claimed invention was made. In analyzing obviousness, I understand that it is important to consider the scope of the claims, the level of skill of a person of ordinary skill in the relevant art, the scope and content of the prior art, the differences between the prior art and the claims, and “secondary considerations” of non-obviousness, which I also understand can include, for example, evidence of commercial success of the invention, evidence of a long-felt need that was solved by an invention, evidence that others copied an invention, or evidence that an invention achieved an unexpected result.

17. I understand that the claimed subject matter can be obvious when a person of ordinary skill in the art would have been motivated to combine pre-

existing elements, and that combination yields no more than what a person of ordinary skill in the art would have expected. I also understand that, in assessing whether a claim is obvious, one must consider whether the claimed improvement is more than the predictable use of prior art elements according to their established functions. I understand that there need not be a precise teaching in the prior art directed to the specific subject matter of a claim because one can take account of the inferences and creative steps that a person of ordinary skill in the art would employ. However, I further understand that obviousness cannot be based on the hindsight combination of components selectively culled from the prior art.

18. I understand that for a reference to qualify as a relevant prior art reference in an obvious analysis, the reference must be “analogous”—meaning it is from the same field of endeavor as the patented technology, and if not, then the reference must be reasonable pertinent to the particular problem facing the inventors of the patent at issue. I also understand that a reference is reasonably pertinent if a person of ordinary skill in the art would have reasonably consulted those references and applied their teachings in seeking a solution to the problem facing the inventors.

IV. Level Of Skill in the Art and Perspective Applied in This Declaration

19. I understand that certain issues relating to validity must be judged from the perspective of a person of ordinary skill in the art (“POSITA”). I further

understand that factors that may be considered in determining the level of skill of a person of ordinary skill in the art include, but are not limited to, (1) the type of problems encountered in the art; (2) prior art solutions to those problems; (3) rapidity with which innovations are made; (4) sophistication of the technology; and (5) the educational level of active workers in the field.

20. I have been informed and understand that a person of ordinary skill in the art is a hypothetical person who is presumed to be aware of all pertinent prior art, thinks along conventional wisdom in the art, and is a person of ordinary creativity. Importantly, this hypothetical person is of ordinary skill in the art to which the claimed subject matter pertains.

21. With respect to the '743 patent, as per Dr. Zadok declaration, "A person of ordinary skill in the art ("POSITA") in the field of the '743 patent, as of May 25, 2000, would have been someone knowledgeable and familiar with the areas of networking and distributed computing that are pertinent to the '743 patent. That person would have a bachelor's degree in computer science, computer engineering, electrical engineering, or equivalent training, and approximately 2-3 years of professional experience relating to distributed computing. Lack of work experience can be remedied by additional education, and vice versa." EX1003, ¶33

22. I agree with Dr. Zadok, and have applied the above-defined perspective of a person of ordinary skill in the art, as of May 25, 2000, in forming

and expressing my opinions. Unless otherwise stated, my testimony below refers to the knowledge of one of ordinary skill in the art at that relevant time.

23. I understand that the parties dispute whether the '743 Patent is entitled to the priority date of the May 25, 2000, or is instead limited to the filing May 24, 2001 date. My opinions in this Declaration would not change if the Board were to adopt and apply either the May 2000 or May 2001 date.

24. At the time of the earliest priority date for the '743 Patent (May 25, 2000), I met the qualifications to be a POSITA, and I believe that I also have a sufficient level of knowledge, experience, and education to provide expert opinions in the field of the '743 Patent.

25. While my own level of skill exceeded that of the ordinary level of skill as of May 25, 2000, I am well-acquainted with the actual performance and capabilities of a person of ordinary skill in the art as defined above. For example, during the relevant timeframe, I was supervising and evaluating the performance of such individuals, such as my graduate students who were performing research in this area or software professionals who were working for Paramark Inc.. Furthermore, my participation in professional organizations and in conferences and other events exposes me to the state of the industry and the level of experience of individuals who, like I did, had at least the ordinary level of skill.

V. Claim Construction

26. In my analysis, I have applied the plain and ordinary meanings of the claim terms in the ‘743 Patent, as a person of ordinary skill in the art would have understood them in the context of that patent in May 25, 2000. And it is my opinion that the prior art set forth in the Petition fails to disclose or render obvious the subject matter of the challenged claims under any reasonable construction and interpretation of the challenged claims’ terms, including their plain meanings, as well as the constructions Petitioner’s expert, Dr. Zadok, proposes.

27. I reserve the right to further explain how skilled artisans would understand terms in the ‘743 Patent’s claims, including but not limited to the terms discussed above, in the event the Board institutes an *inter partes* review of the ‘743 Patent.

VI. The ‘743 Patent

28. Patent ‘743 introduces a highly innovative resource management system that dynamically controls and coordinates networked computers to collaboratively perform tasks beyond the capacity of a single machine. It leverages specialized software to transform any general-purpose computer network into a flexible, resource-managed environment, automatically allocating computing tasks based on real-time performance metrics, workload, and communication traffic across the network. Accordingly, the invention focuses on addressing the

challenges associated with managing large-scale, interconnected computer systems running multiple applications, where resource availability and operational conditions can change unpredictably.

29. It offers significant innovations and improvements over the state-of-the-art distributed systems as of 2001, particularly in managing multiple applications with real-time constraints by addressing the following key challenges (as compared to prior art):

- Resource management and deployment in prior art in distributed real-time systems typically required manual configuration and lacked dynamic adaptability.
- Scalability challenges in prior art emerged as applications grew complex, demanding rapid, automated deployment and resource reallocation.
- Prior art lacked management of multiple coexisting applications with real-time and quality of service constraints in an integrated framework.

30. The innovative approach and corresponding software description presented in the '743 Patent address these challenges in large-scale real-time systems where numerous computing machines and multiple applications make manual intervention impractical and error prone. Unlike previous distributed systems or static parallel processing approaches, this architecture supports

autonomous starting, stopping, and relocating application instances on different hosts to optimize efficiency as conditions change.

31. It further allowed users to define quality of service (QoS) requirements for each application in a user-friendly way, ensuring the system maintains optimal alignment between resources and workloads as demands fluctuate. It presents a resource manager that dynamically generates signals to start or shut down application instances across multiple hosts, enabling flexible, real-time-aware resource allocation while satisfying various managed characteristics required by each application.

32. Therefore, the system and software described in the '743 Patent was designed to operate dynamically, adapting in real time to maintain dependability, scalability, and survivability across distributed nodes.

33. The major functional groups of the Resource Management Architecture as described in the '743 Patent are described in the following paragraphs.

34. The Host and Network Monitoring Functional Group FG1 is designed to deliver comprehensive monitoring across distributed systems by tracking the statuses, configurations, performance metrics, and fault conditions of both hosts and network components. (Ex 1001, 7:29-39). This architecture ensures information is collected in near real-time using minimally intrusive mechanisms,

with data formatted into operating system-independent, standardized messages for consistency and ease of integration across different platforms. The group operates through four principal components (Ex 1001, 9:56-10:14): Host Monitors, which gather detailed system and process-level data on each host, including hardware, software, and network utilization; History Servers, which aggregate and archive this information, providing both real-time and historical records for analysis, visualization, and decision-making within the resource management framework; a Host Discovery function that utilizes SNMP and ICMP to automatically detect topology changes such as hosts going online or offline, ensuring up-to-date awareness of the distributed environment; and the Remos Network Data Broker, which relays network bandwidth and utilization data from specialized external monitoring tools to further inform the resource allocation process. This systematic approach by FG1 supports efficient resource management, accurate state assessment, and prompt detection of operational issues, making comprehensive host and network intelligence readily available for other components and users within the distributed system.

35. The Instrumentation functional group FG2 is designed to deliver comprehensive application event reporting and event correlation in distributed computing environments by collecting detailed application statuses, states, performance metrics, and error conditions; grammar-driven capabilities for

correlating, combining, and reformatting application data, and making this information easily accessible to other components of the Resource Management Architecture (Ex 1001, 7:40-54.). FG2 achieves low-intrusive, near real-time monitoring through open, non-proprietary APIs supporting multiple programming languages (C, C++, Ada) and a variety of platforms (including Solaris, IRIX, Linux) (Ex 1001, 12:15-20). The APIs allow seamless integration with instrumented applications, providing robust support for passing heterogeneous data types, data marshalling, and the ability to evolve instrumented data structures with minimal system recompilation (Ex. 1001, 12:23-33). To meet scalability demands, FG2 supports auto-configuration, multi-tenant operation, connectivity abstraction, and can run on large-scale infrastructures spanning 100+ hosts and many concurrent applications or threads, all while enabling flexible configuration of displays and correlation components (Ex 1002, 12:34-53). Technically, the architecture is composed of several interacting components (Ex 1002, 12:59-13:28). Instrument daemons reformat the event data into standard internal message formats and forward them to Instrumentation Collectors, which act as scalable, centralized relays. The Collectors distribute messages to both Instrumentation Correlators—which use grammar-driven engines to combine, correlate, and synthesize complex composite events from raw instrumentation data—and task-specific Instrumentation Brokers, which filter and further process messages for

displays or other management components. This modular design ensures robust, extensible, and efficient event monitoring and correlation capabilities that can be adapted to evolving system requirements with minimal operational disruption.

36. A critical enabling feature of the resource manager is its use of a System Specification File, which describes the hardware architecture, application structure, and quality of service (QoS) requirements of the distributed system. (Ex 1001, 16:19–45). The System Specification Language FG3 (Ex 1001, 7:55:8:4) is designed to let users comprehensively describe both software and hardware components of a distributed computing environment through user-created System Specification Files. This language allows the modeling of software structures—including systems, subsystems, applications, and processes—as well as hardware and network topology, their configurations, resource and QoS requirements, and survivability, (Ex 1001, 15:2-30). It's hierarchical structure supports grouping components into systems and subsystems, each with assignable priorities, enabling runtime assessment of application importance. At the application level, the language captures critical operational details such as environment variables, working directories, program paths, runtime arguments, startup/shutdown procedures, defined operational states, and specifications for survivability (restart restrictions, multi-instance support, scalability types, and resource estimates). Host and network level attributes—including operating system versions, hardware

architecture, network interface identifiers, benchmark ratings, and switch/router details— are also included (Ex 1001, 15-2:17-3).

37. The Resource Allocation Decision-Making functional group (FG4) serves as the central reasoning and decision-making engine within the Resource Management architecture, responsible for assessing the overall health and state of a distributed environment encompassing hosts, networks, and applications (Ex 1001, 8:5-8:30). By integrating diverse inputs such as system specifications, program control signals, host and network performance data from history servers and network monitoring tools, and application instrumentation metrics, this functional group dynamically determines where and when to allocate or reallocate system resources. The group continuously evaluates key triggers including application failures, host availability changes, inter-application priorities, and performance compliance relative to predefined quality-of-service (QoS) requirements. It orchestrates decisions on starting up, moving, scaling, or shutting down applications to maintain system readiness and adequacy, while resolving dependencies and adjusting system configurations to meet mission priorities (Ex 1001, 17:5-18:13). At the core of FG4 is the Resource Manager (FG42), which functions as the primary decision-making entity (Ex 1001, 18:16-18:36). This maintains a comprehensive global view of all system states and resource statuses, using this holistic insight to respond promptly to failures, dynamically allocate

scalable applications, resolve application dependencies, and adjust application deployments based on changing priority levels dictated by operators or system readiness inputs (Ex 1001, 18:16-18:36). FG42 leverages host fitness scores—computed by the Host Load Analyzer (FG40)—which quantifies host capabilities and resource loads, including CPU, memory, network utilization, and standardized benchmark ratings, aiding in optimal host selection for application placement (Ex 1001, 32:15-34:65). FG42 coordinates closely with Program Control functions by issuing directives to start, stop, or migrate applications, thereby managing resource usage in response to real-time operational demands and performance insights provided by QoS managers. The QoS Managers provide focused oversight of application level as well as data path performance relative to the requirements outlined in the system specifications (Ex 1001, 33:40-64). These managers gather detailed instrumentation data to ascertain compliance with performance thresholds, employing algorithms such as sliding window analysis to smooth transient anomalies and accurately detect sustained deviations that warrant scaling actions. Upon detecting underperformance, QoS Managers recommend scaling up, scaling down, or relocating application instances to the Resource Manager, which exercises final authority over execution of these recommendations (Ex 1001, 33:40:46). Their continuous monitoring ensures that applications operate within defined QoS parameters, optimizing distributed system performance and resource

efficiency while supporting redundancy and load-sharing capabilities intrinsic to scalable applications (Ex 1001, 33:40-64).

38. The Resource (Application) Control functional group FG5 is responsible for managing distributed application startup, configuration, and shutdown throughout the environment (Ex 1001, 23:29-65). This subsystem—referred to as Program Control—enables both operators and automated processes to control application lifecycles interactively across all managed hosts. It centralizes decision-making for application state, supports both operator-defined and dynamically computed application-to-host mappings, and integrates seamlessly with higher-level resource allocation and management components (Ex 1001, 24:1-63). Program Control is architected to provide quick visibility into system status, enables easy loading or editing of application groups, and ensures precise execution of startup and shutdown procedures while maintaining robust fault detection—such as monitoring of application failures or host unresponsiveness through periodic heartbeats.

39. The ‘743 Patent places particular emphasis on the system’s ability to dynamically configure applications. As described, configuring an application includes setting runtime parameters such as environment variables, working directories, executable paths, and command-line arguments, which may incorporate dynamically resolved information like hostnames or user identifiers.

See Ex 1001, 27:23–28:60. Proper configuration is necessary to ensure that applications operate correctly within the distributed environment and can appropriately interact with dependent applications and resources. (Ex 1001, 24:52-63). An application program configuration file is generally a structured text file that defines key parameters and settings controlling how an application is deployed, started, operated, and terminated within a computing environment. It typically includes fields specifying the application’s identity, the desired host(s) or environment(s) for execution, flags controlling automatic startup behavior, runtime host selection, and console requirements. It typically also defines timing controls, working directories, executable paths, command-line arguments (often supporting dynamic or environment variable substitution), shutdown procedures, and logging configurations such as log output types and storage locations. These files enable flexible customization and management of applications without altering their source code, facilitating deployment, automation, and integration across heterogeneous machines that are prevalent in a distributed environment (Ex 1001, 24:52-63).

40. Key innovations exclusive to the ‘743 Patent, as compared to Exhibit 1005, are described in the next few paragraphs.

41. The ‘743 Patent describes the Resource Manager FG 42 is the primary decision-making component of the Resource Management Architecture. For

example, the '743 Patent explains that Resource Manager FG42 is responsible for determining (1) how to respond to host and application failures, (2) where to place new applications, (3) which applications to start up in response to the detection of a new host, (4) how to resolve application dependencies, (5) what applications should be started, stopped, or moved in response to application system priority changes, and (6) based on recommendations from the QoS Managers FG44A-FG44N, when and where scalable application should be started or stopped (Ex 1001, 18:16-27). .

42. Instrument daemons reformat the event data into standard internal message formats and forward them to Instrumentation Collectors, which act as scalable, centralized relays. The Collectors distribute messages to both Instrumentation Correlators—which use grammar-driven engines to combine, correlate and synthesize complex composite events from raw instrumentation data—and task-specific Instrumentation Brokers, which filter and further process messages for displays or other management components. The event correlation process identifies meaningful patterns and conditions across distributed applications, such as combining individual performance warnings into a composite event. This modular design ensures robust, extensible, and efficient event monitoring and correlation capabilities that can be adapted to evolving system requirements with minimal operational disruption.

43. State of health measurements for the ‘743 Patent for each application as well at a path level, while Exhibit 1005 and 1006 discuss this measurement at a path level. For example,

(‘743, col 17: 6-14) “As illustrated in FIGS. 2A, 2B, the Resource Allocation Decision-Making functional group provides the reasoning and decision-making capabilities of the Resource Management architecture. The functions associated with this functional group employ information (listed below) to (1) determine the state and health of the distributed environment (hosts, networks, and applications), and (2) determine what allocation and reallocation actions need to be taken.”

and

(‘743, col 17: 60-65) “Based on application instrumentation data and performance requirements defined in the System Specification Files, determine whether applications are meeting performance requirements and whether an application can be scaled up or moved to attempt to improve performance.”

and

(‘743, col 17: 66-18:3) “Based on application instrumentation data and performance requirements defined in the System Specification Files, determine whether applications are performing well within performance requirements and can be Scaled down.”

44. The quality of service requirements in Exhibit 1005 are provided at the system level (as defined by paths), whereas the requirements in the ‘743 Patent are more granular and include each application. For example,

(‘743, col 18: 31-36) “QoS Managers FG44A-FG44N are responsible for monitoring application and path requirements as defined in the System

Specification Files FG32 and recommending that applications be either scaled up, scaled down, or moved in order to maintain acceptable performance.”

and

(‘743, col 21: 22-28) “The QoS Managers FG44A-FG44N of functional group FG4 are responsible for monitoring application-level performance requirements. These requirements are defined in the System Specification Files FG32 and are monitored primarily via instrumentation data obtained directly from the application code.

The resource manager in the ‘743 Patent actively maintains multiple Quality of Service (QoS) requirements by allocating and reallocating resources proactively. In contrast, Exhibit 1005’s manager only reacts after program failures or deadline misses, using best-effort correction without QoS guarantees.

45. The ‘743 Patent introduces Instrumentation Correlators that use grammar-driven techniques to correlate, combine, and reformat application data into higher-level metrics for composite events used in resource management—. For example,

(‘743, col 50: 58-64) “The Instrumentation Correlators provide grammar-driven capabilities for correlating, combining, and reformatting application data into higher-level metrics (composite events) for use by displays or other Resource Management components. Each Correlator reads in a user-specified correlation grammar file that is interpreted at run-time by the Correlators instrumentation correlation engine.”

46. Thus, the ‘743 Patent emphasizes event correlation as a critical distinguishing feature of the disclosed invention. Instead of simply logging discrete

events in isolation, the system is designed to analyze and correlate these events to derive higher-level conditions indicative of issues requiring intervention. See Ex 1001, 42:44–49. As a result, this capability enables the system to move beyond passive monitoring toward active management based on intelligent interpretation of distributed system behavior.

47. Overall, the ‘743 Patent represents a significant advancement toward automating real-time distributed application management, enhancing scalability, reliability, and real-time performance beyond prior art presented in Exhibit 1005 and 1006. It achieved this by filling critical gaps that existing distributed computing efforts lack, setting a new standard for the dynamic allocation and management of computing resources in complex network environments at the time of its submission.

48. Accordingly, the overall objective of the ‘743 Patent is to provide a comprehensive framework for real-time, adaptive control of distributed computing resources. The system achieves this by integrating hardware and software monitoring, application-level event correlation, dynamic resource allocation, and configuration management. Through this integration, distributed computing environments are able to maintain high levels of dependability, timeliness, and availability, even under unpredictable and adverse conditions. By employing a dynamic, QoS-driven approach, the system overcomes the limitations of prior

static management techniques and enables mission-critical environments to respond intelligently to failures, bottlenecks, and workload variations without requiring manual intervention. See Ex 1001, 2:15–19.

49. In summary, the ‘743 Patent discloses a distributed system capable of monitoring the health of hosts, networks, and individual application programs; correlating raw event data into higher-level composite events to guide intelligent decision-making; configuring application programs for both static and dynamic behavior; and managing overall system behavior in real time to satisfy defined QoS objectives for multiple applications. As a result, these capabilities represent significant improvements over prior static resource management systems, positioning the invention as a comprehensive solution for survivable, scalable, and self-managing distributed computing architectures.

VII. The Claims 10, 11, 14, 16 and 18 Of The ‘443 Patent Would Not Have Been Obvious.

50. Petitioner challenges claims 10, 11, 14, 16 and 18 (the “Challenged Claims”) of the ‘743 Patent based on several grounds.

51. In my opinion, none of the claims of the ‘743 Patent would have been obvious over Petitioner’s prior art, for reasons I explain below. I will first review the Petitioner’s prior art. Then I will address each of the grounds.

A. Overview of Welch

52. Welch is titled “*“Distributed, Scalable, Dependable Real-Time Systems: Middleware Services and Applications.”* Lonnie R. Welch et al., Proceedings 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, April 1999.”

53. The logical architecture of the QoS management software in “Welch” centers on monitoring and maintaining quality of service for real-time control paths that are part of the system. A metrics calculation component computes path-level QoS metrics and forwards them to the QoS monitor. The monitor verifies if the observed QoS meets the required standards and alerts the QoS diagnosis component when violations occur. This diagnosis component identifies causes of poor QoS and recommends corrective actions such as moving, shedding, or replicating programs. These actions are ranked and filtered by the action selection component, then passed to the allocation analysis component, which determines optimal resource allocation based on host and LAN load metrics before enacting the changes.

54. The physical architecture is built around the resource manager, which activates when programs fail, or deadlines are missed. It uses data from hardware and software monitors, along with a specification file describing QoS requirements and system structure, to make decisions. The system data repository collects and maintains all system information, while the program control component manages

program execution on hosts. The human-computer interface (HCI) provides users with system status, application information, and reallocation decisions.

55. The adaptive resource management system operates in three modes: start-up, QoS monitoring/enforcement, and program recovery. During start-up, the system loads user specifications, monitors hardware loads, and initiates real-time control programs on specified hosts. In monitoring mode, QoS monitors detect unhealthy paths and suggest corrective actions, which the resource manager implements by selecting hosts and LANs with the lowest load indices. In recovery mode, if a program terminates due to faults, the system detects failure, identifies suitable hosts, and restarts the program while limiting retries to prevent repeated failures.

56. Overall, this architecture integrates continuous monitoring, diagnosis, and adaptive resource allocation for fault tolerant and distributed computing at a system level (represented by paths and not at a per application basis). As described in the introduction (Ex 1005, 2):

Section 2 provides an overview of a middleware architecture for dynamic QoS management of path-based systems, and describes the adaptive resource allocation approach employed by the middleware.

57. It is also worth noting that Welch cites the following reference as[11]:

[11] L. R. Welch, B. Ravindran, B. Shirazi and C. Bruggeman,
“Specification and analysis of dynamic, distributed real-time systems,”

in Proceedings of the 19th IEEE Real-Time Systems Symposium, 72-81, IEEE Computer Society Press, 1998.

However, citation [11] in Welch has a typo. The paper as it appears on page 72 of the conference proceeding is actually “Specification and Modeling of Dynamic, Distributed Real-Time Systems”. This is referred to as Exhibit 1006 or Ravindran as described in the next section.

58. Welch further describes experimental validation conducted using a Navy High Performance Computing (HiPer-D) testbed, which is described in citation [10] of Welch. Citation [10] is “Challenges in Engineering Distributed Shipboard Control Systems” (EX2006) which discuss in more detail below.

59. Accordingly, Welch’s reference to the HiPer-D testbed indicates that Welch’s middleware service was tested in the context of missile tracking and weapons targeting. These capabilities are directed toward processing real-time sensor data for naval anti-air-warfare operations and are not concerned with monitoring or managing individual application programs. See EX1005, 5.

60. Welch further cites to “Specification and analysis of dynamic, distributed real-time systems,” in the Proceedings of the 19th IEEE Real-Time Systems Symposium, 72–81, IEEE Computer Society Press, 1998. Ex 1005, 5. As shown in Ex 2007, the paper presented at the conference was actually entitled “Specification and Modeling of Dynamic, Distributed Real-Time Systems.” I

agree with Petitioner that the citation in Welch is a typographical error and that the proper reference is to Ravindran, Ex 1006. See Petition, 39.

61. As noted in the Petition, Welch relies on Ravindran to explain that his “approach is based on the dynamic path paradigm” and that “[a] path-based real-time subsystem (see [Ravindran]) typically consists of a detection and assessment path, an action initiation path, and an action guidance path.” Petition, 38; see also Ex 1005, 1. Specifically, Ravindran explains that a “subsystem can be modeled using three dynamic paths: threat detection, engagement, and missile guidance.” Ex 1006, 2. Ravindran further explains that “[t]he threat detection path examines radar sensor data (radar tracks) and detects potential threats,” the “engagement path is activated, resulting in the firing of a missile to engage the threat,” and the “missile guidance path involves sensor hardware, software for filtering, software for evaluating and deciding, software for acting, and actuator hardware” (Ex 1006, 2).

62. Thus, the dynamic path paradigm described by Ravindran and cited by Welch analyzes radar data to detect a threat and subsequently guides a missile to destroy that threat once detected. This understanding is consistent with Welch’s description of the dynamic path paradigm, which notes that the paths interact with the environment by “evaluating streams of data from sensors, and by causing actuators to respond (in a timely manner) to events detected during evaluation of

sensor data streams” (Ex 1005, 1). Therefore, when read in the context of Ravindran, Welch clearly distinguishes the environmental data streams collected from radar sensors from the paths that perform real-time military functions such as threat detection, engagement, and missile guidance.

63. In sum, when properly understood in light of its cited references to the HiPer-D testbed and Ravindran (Ex 1006), Welch discloses a middleware system focused on managing end-to-end control paths in a distributed, real-time military environment, particularly for missile tracking and engagement operations. Welch’s reliance on the HiPer-D testbed and the dynamic path paradigm described in Ravindran confirms that Welch’s focus is on processing environmental sensor data for real-time system for military decision-making, and not on monitoring or managing the state, health, or configuration of individual distributed application programs.

64. While Welch mentions “track correlation and filtering algorithms,” these algorithms relate to radar data processing for missile tracking and do not disclose or suggest application-level event correlation capabilities that derive higher-level insights from software-generated performance metrics, statuses, or internal states.

65. Importantly, Welch does not disclose determining the health of distinct application programs, but rather monitors the health of complex, multi-

component control paths. Similarly, Welch does not apply QoS parameters to individual applications; instead, it discusses system-wide objectives such as dependability and responsiveness without tailoring QoS constraints to specific applications. Welch also does not describe configuring applications. The reference discusses starting, stopping, or moving programs, but it does not address technical configuration processes, such as setting runtime parameters, or preparing an application for distributed execution. In short, Welch is directed toward maintaining system-wide operational integrity in distributed real-time environments and does not disclose the application-level instrumentation, health monitoring, QoS enforcement, or configuration functions required by the challenged claims of the '743 Patent.

B. Overview of Ravindran

66. Ravindran is titled "*Specification and Modeling of Dynamic, Distributed Real-time Systems.*" Lonnie R. Welch et al. IEEE Real-Time Systems Symposium, 72-81, IEEE Computer Society Press, Dec. 1998."

67. Time-constrained systems operating in dynamic environments face significant challenges due to unpredictable worst-case scenarios and large variances in data and event sizes, which cause fluctuating execution latencies and resource demands. These systems cannot be accurately characterized by fixed or

time-invariant statistical models. To address this, “Ravindran” introduces a programming-language-independent specification language designed to describe environment-dependent features of real-time Quality of Service (QoS) for dynamic real-time paths in a system.

68. Specifically, the programming-language-independent specification language is designed to describe timing constraints and environment-dependent behaviors at the level of end-to-end “paths” through a distributed system. These paths span multiple application programs, network elements, and hardware components, enabling system designers to model real-time constraints across complex, multi-component flows, rather than focusing on the behavior of individual tasks in isolation. Accordingly, a “path” refers to an end-to-end control flow spanning multiple components— including software applications, networks, sensors, processors, and actuators. Specifically, in Ravindran, three key path types are defined: situation assessment (continuous evaluation of sensor data), action initiation (event-driven activation of responses), and action guidance (quasi-continuous monitoring and control of actuators).

69. Central to Ravindran’s framework is the dynamic real-time path paradigm, which models real-time systems as comprising multiple types of paths: situation assessment paths (such as continuous threat detection based on sensor data), action initiation paths (such as launching a missile upon threat detection),

and action guidance paths (such as tracking and guiding a missile in flight). These paths interact with the external environment by evaluating data streams received from sensors and issuing control commands to actuators. Each path type can have specific timing and QoS requirements, with the system adapting dynamically to varying workloads and deadlines.

70. The specification language and associated static and dynamic models enable adaptive QoS management by combining pre-defined system descriptions with real-time environment-dependent state information. This facilitates continuous monitoring, diagnosis, and resource reallocation to maintain acceptable QoS levels of the system despite environmental changes and faults. However, all of this was targeted for a path or a system and not for each application separately.

C. Overview of Hosokawa

71. Hosokawa refers to U.S. Patent No. 6,088,727 entitled “Proactive method for ensuring availability in a clustered system.”

72. Hosokawa presents a method and system designed to manage computer clusters that share a common data space. The invention addresses the complexities of coordinating multiple computers or servers operating together in a cluster environment, ensuring efficient and reliable data access among the nodes in the cluster. This coordination is crucial for maintaining data consistency and optimizing system performance in multi-computer setups.

73. A key feature of the patent is its focus on cluster control mechanisms that enable synchronized operation across the various machines. The system allows the cluster to function as a cohesive unit, managing access requests and resource allocation effectively. This coordination prevents conflicts and data corruption, which are common issues in shared data environments where multiple systems attempt simultaneous data manipulation.

74. Hosokawa discloses a cluster control system designed to monitor and manage resources across multiple computers within a cluster. The system enables the automatic transfer of application and service programs (“packages”) from one computer to another in the event of faults or failures, thereby minimizing downtime. Hosokawa improves upon prior approaches by supporting rapid and reliable failover, enabling prioritized and coordinated control of package execution, and maintaining system operations even if the cluster manager itself experiences a failure.

D. Overview of Welch 1996 (Ex 2006)

75. Welch reference [10] corresponds to “Challenges in engineering distributed shipboard control systems.” Welch, Lonnie R., et al. “. *The IEEE Real-Time Systems Symposium*. 1996”.

76. This paper presents infrastructure support for a shipboard anti-air-warfare (AAW) control system that was developed using High Performance

Distributed Computing Program (HiPer-D). This program was established jointly by DARPA and the Aegis Shipbuilding Program to explore and validate distributed computing architectures for Navy shipboard systems. HiPer-D combined top-down engineering studies with large-scale experiments using commercial computing technology, aiming to provide design concepts and engineering data that enable cost-effective scalability and high capacity. HiPer-D envisioned expanding computing roles to cover not only combat systems but also broader ship systems and administrative functions, treating computing as an integral ship infrastructure enabling ubiquitous computing. The program also emphasizes an evolutionary transition to a fully distributed architecture to manage risks associated with adopting COTS systems and to enable the gradual modernization of combat systems without incurring excessive cost or schedule delays.

77. The publication presents a tested implementation of the AAW with the following features (Ex 1008, pg. 2)

“the AAW system includes the following capabilities: a simulated track source; track correlation and filtering algorithms; track data distribution services; a doctrine server and three types of doctrine processing (semi-auto, auto-sm, and auto-special); an engagement server; a display subsystem including X-windows based tactical displays, submode mediation, and alert routing; surface operations (reengineered from CMS-2 to Ada), and simulated weapons control system and identification upgrade capabilities”

78. The details are provided in the following figure (from Ex1008, pg. 3)

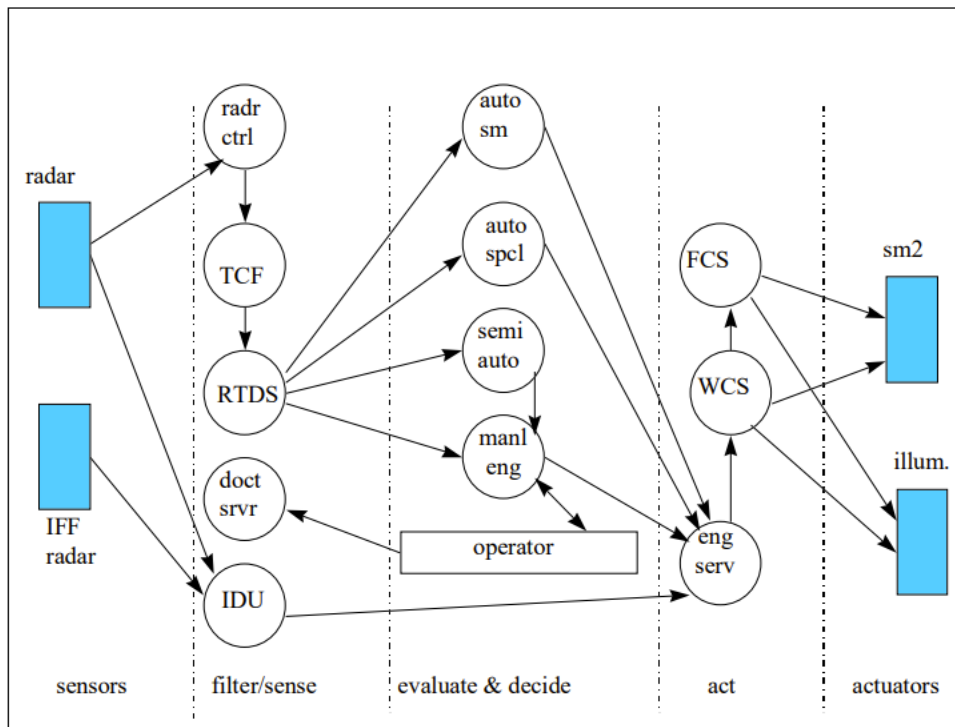


Figure 1. Elements of a shipboard AAW system.

79. It is worth noting that correlation as described above refers to object tracks detected by radar and is not related to application related data that is stored and used by QoS Management of distributed systems. Instead, Key capabilities of the HiPer-D testbed, as described in “Challenges in Engineering Distributed Shipboard Control Systems,” include managing weapon targeting systems and tactical display subsystems for operator interaction. The system includes track correlation and filtering components responsible for processing radar sensor data, performing track initialization and refinement, and distributing track information to other modules through the Radar Track Data Server (RTDS).

80. Accordingly, the paper addresses the need for scalable, high-capacity computing systems in response to evolving naval missions, where traditional Navy-standard architectures proved inadequate. To this end, HiPer-D implemented a distributed, real-time, fault-tolerant version of a shipboard AAW control system within a testbed environment.

81. Notably, the track correlation and filtering functions focus on sensor data processing—specifically, filtering radar tracks and distributing sensor-based track information to other components—not on application-level event correlation or health monitoring of software applications. The paper explains that the system architecture relies on replication of tactical programs for fault tolerance and scalability, using middleware services to monitor system health and to restart failed components. Timing constraints are managed at the level of large-grain control paths spanning sensors, filters, decision-making modules, and actuators, rather than at the level of individual software applications.

82. Ultimately, the distributed AAW system described in this paper was designed for radar tracking, weapons engagement, and shipboard battle management—not for distributed application program event correlation, application health monitoring, application-specific QoS enforcement, or application configuration as required by the challenged claims in the present case.

E. Ground 1: Claim 10 Would Not Have Been Obvious In View of Welch alone, or Welch in combination with Ravindran.

83. The opinion of the Dr. Zadok (Ex 1003) and Tesla the Petition (Paper 2) for inter partes review does not address the key innovations and claim 10 in the ‘743 Patent. In addition to the distinctions between the ‘743 Patent and the Exhibit 1005 and 1006 which I have described in the previous section, the references do not disclose the following.

1. [10.2] a second function group which provides ... event correlation capabilities;

84. In the context of the ‘743 Patent, “correlation capabilities” refer to the system’s ability to analyze and interpret raw instrumentation data from distributed applications in order to generate higher-level, meaningful insights— referred to in the patent as composite events.

85. The ‘743 Patent explains that this functionality may be provided by the Application-Level Instrumentation functional group (FG2), particularly through components such as the Instrumentation Correlators (FG26A–FG26N). These correlators may use a grammar-driven engine to identify patterns in application-generated data, combining and reformatting individual event messages into structured summaries that reflect significant system behaviors or conditions. Instrumentation Brokers and the QoS Monitor (FG28 and FG29) further support

this capability by transforming instrumentation data for consumption by monitoring displays or Quality-of-Service (QoS) Managers.

86. This correlation process goes beyond basic communication or message passing; it actively derives context and meaning from distributed events, enabling more informed decisions by other system components, such as triggering application scaling or reallocation. In this way, correlation capabilities form a critical link between raw telemetry and the intelligent, adaptive behavior of the resource management system.

87. The phrase “composite event” as used in the ‘743 Patent refers to a higher-level, derived metric or condition that is generated by correlating and combining multiple lower-level instrumentation events reported by distributed applications.

88. Composite events are produced by the Instrumentation Correlators (FG26A–FG26N), which use a grammar-driven engine to interpret sequences or patterns of application-generated data. These raw data points may include status messages, performance metrics, internal application states, or error reports. The correlators process this data based on user-defined rules to recognize when a meaningful higher level condition has occurred.

89. For example, the ‘743 Patent states:

The Instrumentation Correlators provide grammar-driven capabilities for correlating, combining, and reformatting application data into higher-level metrics (composite events) for use by displays or other Resource Management components. Each Correlator reads in a user-specified correlation grammar file that is interpreted at run-time by the Correlators instrumentation correlation engine. Ex 1001, 50:56-63

90. Accordingly, instead of merely logging that Application A emitted “low level event X” and Application B emitted “low level event Y,” the system may recognize that when these events occur within a certain time window or frequency, they together signify a high level composite event like “data processing pipeline overload.” Higher-level metrics are important because they abstract away raw data noise and expose semantically rich insights that can guide automated resource management decisions. These insights are used by components such as the QoS Managers and Resource Manager to determine when to start, stop, or move applications in order to maintain system-wide quality-of-service goals. In short, higher-level metrics serve as intelligent, synthesized representations of system health and behavior, enabling the distributed environment to operate adaptively and efficiently.

91. Dr. Zadok appears to misapprehend the “event correlation capabilities” recited in the claims of the ‘743 Patent that form a critical link between raw telemetry about the components in the distributed system and the

intelligent, adaptive behavior of the system. Instead, Dr. Zadok contends that Welch discloses “event correlation capabilities” by describing correlation radar data with the flight of missiles or inherently by including two applications that communicate with each other. As described below, both of these arguments are inaccurate, flawed and conflates these distinct technical domains, reading event correlation capabilities into Welch where none exist.

92. For the first argument, Dr. Zadok, uses Welch’s disclosure of “track correlation and filtering algorithms” as evidence of the claimed event correlation functionality. In particular, in Ex 1003 at Par. [0158], he states:

Next, Welch’s metrics calculation component (part of the “second function group”) provides “event correlation capabilities” because the QoS management software “implementation includes the following capabilities: ... (2) track correlation and filtering algorithms.” Ex 1005, Welch, 4.

93. However, Dr. Zadok appears to have failed to consider the first sentence of above quote from Welch that states “The technology described in this paper was evaluated within the Naval Surface Warfare Center High Performance Computing (NSWC HiPer D) Testbed, which contains the experimental Navy system described in [10].” When read in the context of reference [10] (“Challenges in Engineering Distributed Shipboard Control Systems”), the “(2) track correlation and filtering algorithms” are for tracking missiles in an anti-air-warfare control

system and not with applications in a QoS context as “collecting and correlating application-provided data such as application statuses, states, performance, and internally detected errors” Ex 1001, 7:43-45.

94. This is corroborated by the publication entitled “Challenges in Engineering Distributed Shipboard Control Systems” (Ex 2006), these algorithms pertain to a radar-based missile tracking system implemented within a naval anti-air-warfare (AAW) context. Their function is limited to processing sensor data derived from simulated or actual targets and distributing such data to other system components, rather than collecting or analyzing software application-level data, including statuses, errors, or performance metrics.

95. Dr. Zadok’s expert report (Ex 1003, Par. [0158]) uses Ex 1005, Welch, 4 as follows:

“The technology described in this paper was evaluated within the Naval Surface Warfare Center High Performance Computing (NSWC HiPer D) Testbed, which contains the experimental Navy system described in [10]. The implementation includes the following capabilities: (1) a simulated track source, **(2) track correlation and filtering algorithms,** (3) track data distribution services, (4) a doctrine server and three types of doctrine processing, (5) an engagement server, (6) a display subsystem including Xwindows based tactical displays, submode mediation, and alert routing surface operations, (7) a simulated weapons control

system, and (8) identification upgrade capabilities.” Ex 1005, Welch, 4.

96. Reference [10] in Exhibit 1005 describes the development and implementation of a distributed anti-air warfare (AAW) control system within the HiPer-D (High Performance Distributed Computing) program, a joint effort by DARPA and the U.S. Navy. Faced with the limitations of legacy military computing systems, HiPer-D explores scalable, fault-tolerant architectures using commercial off-the-shelf (COTS) hardware to support real-time operations across heterogeneous networks. The publication describes a modular system architecture comprising components such as sensors for data collection, advanced decision-making algorithms, engagement management modules, and operator interfaces, all seamlessly unified within a distributed platform designed to enable the real-time monitoring of thousands of targets and the effective orchestration of multiple concurrent engagements. As described in Figure 1 (reproduced below) from [10], Track Correlation and Filter (TCF) algorithms correlate and filter the radar data and not system or application monitoring as per ‘743 Patent

Track Correlation and Filter (TCF) - TCF takes input from a sensor simulator and performs track initialization, radar track filtering, and related activities. The TCF component provides track data to the Radar Track Data Server (RTDS) for distribution to all other components of the system that require it.

Radar Track Data Server (RTDS) - Clients of the RTDS register for distribution of track data, specifying a minimum periodicity and a maximum tolerable latency. The RTDS then distributes the data received from the TCF function to the various clients at their specified rates. Ex 2006, 3

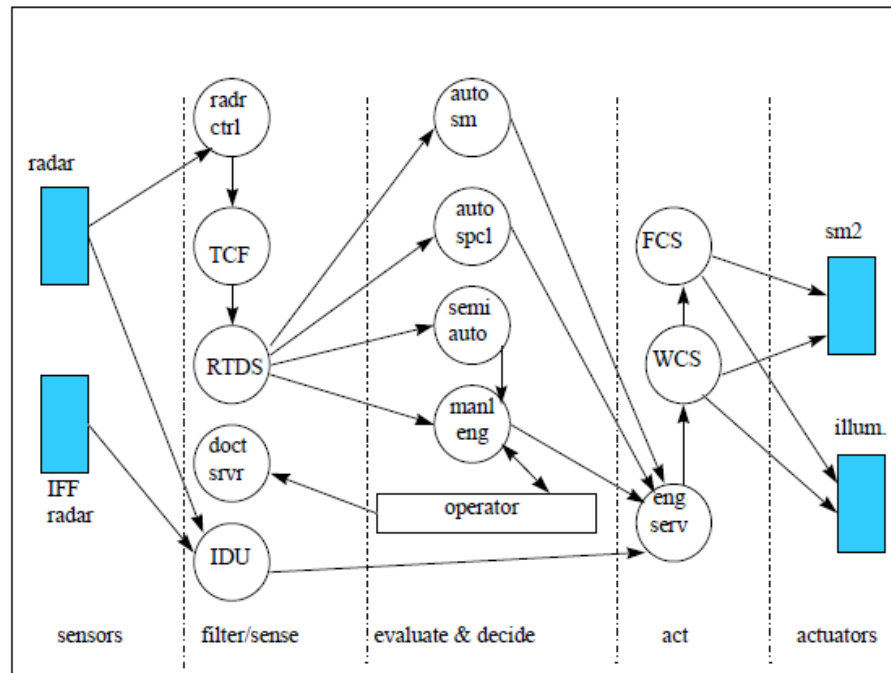


Figure 1. Elements of a shipboard AAW system.

97. In particular, as shown in the table below the track filtering in [10] utilizes the same language as Welch

“Challenges in Engineering Distributed Shipboard Control Systems” (Ex 2006)	Welch (Ex 1005)
The Testbed implementation of the AAW system includes the	The implementation includes the following capabilities: (1) a

following capabilities: a simulated track source; <u>track correlation and filtering algorithms</u>; track data distribution services; a doctrine server and three types of doctrine processing (semi-auto, auto-sm, and auto-special); an engagement server; a display subsystem including X-windows based tactical displays, submode mediation, and alert routing; surface operations (reengineered from CMS-2 to Ada), and simulated weapons control system and identification upgrade capabilities. Ex 2006, 2	simulated track source, <u>(2) track correlation and filtering algorithms</u>, (3) track data distribution services, (4) a doctrine server and three types of doctrine processing, (5) an engagement server, (6) a display subsystem including Xwindows based tactical displays, submode mediation, and alert routing surface operations, (7) a simulated weapons control system, and (8) identification upgrade capabilities.
---	--

Thus, the above description has absolutely nothing to do with monitoring how the AAW Software System is operating. Therefore, when Welch references “track correlation and filtering algorithms,” Welch is referring to the capabilities of the anti-air-warfare (AAW) control system described in “Challenges in Engineering Distributed Shipboard Control Systems.”

98. Based on the above analysis, I disagree with Dr. Zadok’s conclusion “In sum, Welch discloses “a second function group [e.g., software monitors] which provides application computer program [e.g., RTCS applications] event reporting [e.g., unhealth sub-paths, trend of load, etc.] and event correlation [e.g., correlation and filtering algorithms] capabilities,” Ex 1003, ¶164. This conclusion is

inaccurate as there are no similarities between correlating and filtering radar data to track missiles with tracking the operations of a software system. Accordingly, Petitioner’s arguments fail to show that Welch discloses “a second function group which provides application computer program event reporting and event correlation capabilities” as required by the challenged claims.

99. Dr. Zadok further attempts to satisfy the claim limitations by asserting that, because Welch states that applications in the system are “dependent (communicate with each other).” In the context of the ‘743 Patent and “Welch”, dependent applications in the system rely on each other to complete a process. This means one application’s output or state is a prerequisite for another. Workflow engines use these dependencies, ensuring proper sequencing in a consistent manner. Communication between applications does not, by itself, imply the presence of a mechanism capable of observing and analyzing events to detect patterns or causal relationships.

100. Specifically, Dr. Zadok states, “given that Welch’s “applications are dependent (communicate with each other),” a POSITA would have understood that the performance information of those applications includes “event correlation capabilities”— the ability to communicate messages and events between applications. Ex 1005, Welch, 2.” Ex 1003, ¶159. The portion of Welch, he appears to be relying upon is as follows:

A major difference between the traditional load balancing techniques [7] and dynamic QoS-based resource management services lies in the overall goals. While load balancing systems (see Load-Leveler [5], LSF [12], NQE [2], PBS [4], Globus [3], and Condor [1]) attempt to achieve system performance goals such as minimized response time or maximized throughput, dynamic QoS-based resource managers strive to meet the QoS requirements of each application they manage. Another major difference between these systems is their workload models. Traditional load balancing systems assume independent jobs with known resource requirements. **In a dynamic resource management system, the workload requirements of applications can vary, based on environmental conditions; additionally, applications are dependent (communicate with each other).** Ex 1005, 2-3

101. The mere fact that two applications can communicate with each other does not inherently mean the system that includes the applications has correlation capabilities.

102. Imagine two friends— Alice and Bob— are having a phone conversation. They're clearly communicating with each other, exchanging information in real time. However, just because they're talking doesn't mean there's someone on the line analyzing both sides of the conversation to detect patterns, infer causes of misunderstandings, or summarize key points. For that, you'd need a third party— say, a translator or a meeting assistant— who listens to both sides, compares their statements, identifies important events, and connects the

dots between them. More importantly, the phone does not have the ability to detect that since both Alice and Bob coughed more than a predetermined number of times during the phone conversation that the area in which Alice and Bob are located is experiencing a COVID outbreak.

103. Similarly, in a distributed system, two applications might send messages back and forth as part of a control workflow, but that communication alone doesn't mean the system has "correlation capabilities." Correlation requires a separate mechanism that observes, collects, and analyzes events from multiple sources to identify relationships, causes, or trends. Without that layer, the system may facilitate communication but lacks the intelligence to interpret or correlate the events.

104. Accordingly, the mere fact that two applications can communicate with each other does not inherently mean the system that includes the applications has correlation capabilities. Specifically, event correlation requires more than mere communication; it necessitates a distinct component (or mechanism) that observes, collects, and processes events from multiple sources to identify patterns or causal relationships. Welch contains no disclosure of any such functionality, and Petitioner has failed to demonstrate otherwise.

105. Thus, Dr. Zadok's assertion that such capabilities are inherently present simply because the applications described in Welch "communicate with

each other” is unsupported by the cited disclosure. The cited portion of Welch merely notes that the applications are interdependent within a dynamic resource management system, but says nothing about correlating application-level events such as performance metrics, states, or internal errors.

106. Thus, his declaration and the petition has not demonstrated that the referenced prior art discloses or suggests the claimed limitation of event correlation capabilities and the above clause of the challenged claims.

2. [10.4] wherein the third function group comprises: a first function which determines a state and health of the N hosts, a network operatively coupling the N hosts to one another and the M managed characteristic application computer programs in the distributed environment

107. The above element of challenged claims requires determining the “state and health” of three distinct components: (1) the hosts, (2) the network, and (3) the application programs. Ex 1001, 56:60-64. Dr. Zadok, states that in Welch, “The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor” and “If a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program) to the Resource Manager.” Ex 1003, ¶188 (internally quoting Welch, 2-3). From this discussion of “paths,” he incorrectly concludes that “the resource

manager determines the state and health of the RTCS applications.” Ex 1003, ¶189. In fact, Welch clearly states “Our approach is based on the dynamic path paradigm. A path-based real-time subsystem (see [11]) typically consists of a detection & assessment path, an action initiation path and an action guidance path.” (Reference [11] corresponds to Ex 1006).

108. The “dynamic real-time path paradigm” (which is central to Ex 1006) is described as follows (including the figure below):

A path-based real-time subsystem (see Figure 1) typically consists of a situation assessment path, an action initiation path and an action guidance path. The paths interact with the environment via evaluating streams of data from sensors, and by causing actuators to respond (in a timely manner) to events detected during evaluation of sensor data streams.

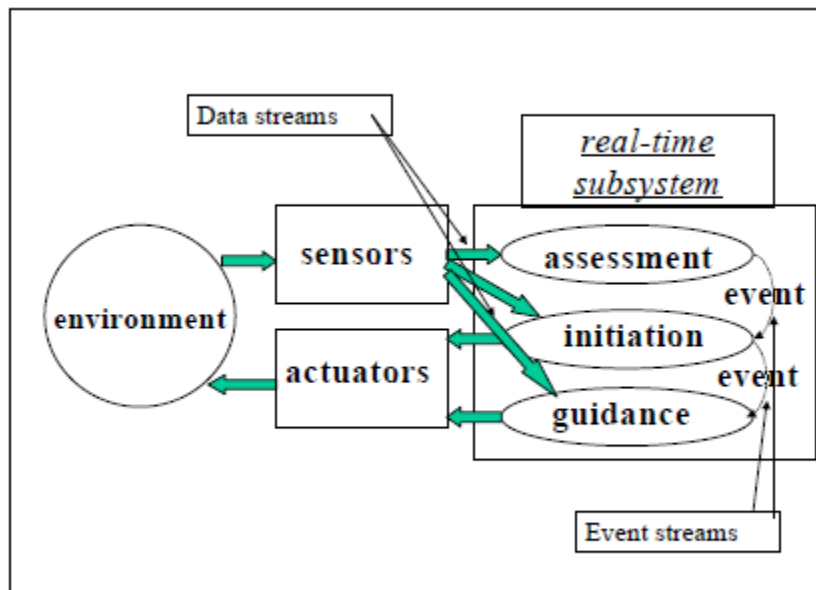


Figure 1. A real-time subsystem.

Ex 2007, 2

109. Exhibit 1006 clearly distinguishes “application programs” from “paths” as follows:

Rather than providing real-time support within a particular application language, it provides support for expressing timing constraints for systems of application programs, written in a melange of programming languages. Unlike previous work, in which timing constraints are associated at a relatively small granularity (e.g., task-level), our language allows timing constraints to be expressed the granularity of end-to-end paths which span many communicating programs.

Ex 1006, 1-2

110. Although Ravindran uses the term “events,” the reference is to the outcomes of processing radar sensor data through the “assessment,” “initiation,” and “guidance” real-time paths that collectively determine whether to engage a threat by launching a missile. For example, the assessment path “continuously evaluates the elements of a sensor data stream to determine if environmental conditions are such that an action should be taken (if so, the action initiation path is notified).” See Ex 1005, 2. The initiation path “uses inputs from sensors to determine which actions should be taken and how the actions should be performed, notifies actuators to start performing the actions, and informs the action guidance path that an action has been initiated.” Ex 1005, 3. The guidance path “is activated by the missile launch event” and, once activated, “continuously issues guidance

commands to the missile until it detonates (the deactivation event).” Id. Thus, the “events” described in Ravindran pertain to external military operations—such as detecting a target, initiating missile launch, and guiding the missile to detonation—not to monitoring, reporting, or correlating internal statuses or performance metrics of distributed application programs, as required by the challenged claims.

111. In this setting, a “path” encompasses the entire journey of data and control signals across a distributed system, involving various software applications along with network and hardware elements such as sensors, processors, and actuators. Each path serves as a functional channel, dedicated to fulfilling a particular real-time goal like assessing situations, initiating actions, or guiding those actions. To illustrate, consider a naval missile defense system: a situation assessment path would be tasked with continually analyzing radar inputs to identify threats. An action initiation path would manage the processes involved in launching a missile, while an action guidance path would provide the necessary real-time instructions to direct the missile during flight.

112. This usage of “path” in Exhibit 1006 is consistent with its usage in Welch. For example, Welch states “[t]he paths interact with the environment via evaluating streams of data from sensors, and by causing actuators to respond (in a timely manner) to events detected during evaluation of sensor data streams.” Ex 1005, 1. The word path in the statement “Our approach is based on the dynamic

path paradigm,” is used to describe end-to-end, real-time data and control flows that traverse multiple distributed components— including software applications, networks, sensors, processors, and actuators.

113. Dr. Zadok does not provide any explanation of why “end-to-end, real-time data and control flows that traverse multiple distributed components— including software applications, networks, sensors, processors, and actuators,” is an “application program” with the meaning of the ‘743 Patent.

114. It is worth noting that in his declaration, Dr. Zadok (Ex 1003, ¶190) says:

The resource manager determines whether the paths (i.e., applications) are healthy or not as well. “If a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program) to the Resource Manager (step 8 of Figure 3).” EX1005, Welch, 3.

Ex 1003, ¶190

In the first sentence Dr. Zadok equates paths and applications, while in a later reference to by him to Welch “If a path becomes unhealthy.... diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program)” suggests that application is not the same as path.

115. If, as Dr. Zadok alleges, a path and an application were equivalent, Welch's statement would logically read: "If a path becomes unhealthy... suggest an action such as moving or replicating the path." Yet that is clearly not what Welch says. Instead, Welch's statement that "[i]f a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program) to the Resource Manager" reinforces that Welch is adopting the dynamic path paradigm described in "Specification and analysis of dynamic, distributed real-time systems." Welch expresses timing constraints with respect to end-to-end paths, which span multiple communicating application programs, network components, and actuators, rather than individual applications themselves.

116. Based on the above analysis, Dr. Zadok's declaration and the petition have not demonstrated that prior art discloses or suggests the claimed limitation of determining the state and health of characteristic application computer programs. Thus, the petition has not demonstrated that Welch discloses or suggests the claimed limitation of this clause.

3. **[10.5] a second function which determines required allocation and reallocation actions needed to maintain a plurality of Quality of Service (QoS) requirements established for the M managed characteristic application computer programs, the QoS requirements dictating parameters regarding service quality of the M management characteristic application programs; and**

117. Dr. Zadok's declaration has not established that Welch describes "QoS requirements dictating parameters regarding service quality of the M management characteristic application programs," as specifically required by the above clause. The portions of Welch referenced by Dr. Zadok addresses broad, system-wide QoS goals such as dependability, timeliness, and availability, but only in the context of overall real-time system performance— not in relation to distinct and multiple application programs as required by the above clause and described in the '743 Patent. In his report (Ex 1003 ¶199), Dr. Zadok simply reiterates Welch's general discussion of system objectives and makes a conclusory claim that these objectives also pertain to RTCS applications, without identifying any direct support for this connection. A closer reading reveals that paragraph Ex 1003 ¶199 does not provide any evidence or explicit link tying the system-level QoS requirements described by Welch to any particular RTCS applications.

118. Welch clearly states that:

"If a sub-path (RTCS program) is terminated due to some hardware/software failure, Program Control detects such a condition and informs the Resource Manager accordingly "

and

“The Resource Manager informs the Program Control to start the Real-Time Control System (RTCS) application programs on the specified hosts (step 5 of Figure 3).”

and

“The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor.”

119. The implication of all the above is that in referenced prior art, QoS requirements were at path level and not an individual application level.

120. Additionally, Dr. Zadok concludes that “the functions of Welch’s software that allocate and reallocate resources to match QoS requirements are thus a “second function” as claimed.” Ex 1003, 200, He uses the following disclosure from Welch that “If a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program) to the Resource Manager (step 8 of Figure 3)” Id for reaching his conclusion. However, this again clearly demonstrates that it is the health of the path that is being monitored not the health of the RTCS application.

121. The specification file referenced in Welch outlines the overall architecture of the path but does not provide details on application-specific QoS parameters. Additionally, the statement that the resource manager may relocate or replicate an application based on the health of a “path” emphasizes that Welch tracks QoS at the level of system-wide execution flows rather than individual applications. In Welch, paths refer to complex, end-to-end data and control flows that span multiple components, including applications, networks, and hardware, so monitoring these paths’ health is not the same as monitoring the QoS of specific applications.

122. Therefore, the Petition’s and Dr. Zadok claim that Welch’s resource management functionality corresponds to the claimed “second function” lacks support as the referenced disclosures discuss QoS in terms of overall system behavior (represented by one or more paths) rather than individual application requirements. Thus, the petition has not demonstrated that the referenced prior art discloses or suggests the claimed limitation.

4. **[10.7] a fourth function group which provides program control capabilities permitting starting, stopping, and configuring of selected ones of the M managed characteristic application computer programs on respective ones of the N hosts in the resource managed system,**

123. The discussion of Application configuration in Ex 1005 and Ex 1006 is missing. Configuration is an important requirement for dynamic QoS distributed systems. As per the ‘743 Patent (ex 1001, 25:50-63)

“Configuration Files FG56—contain an ordered set of applications that can be loaded at the Program Control display and then either edited or executed. The Configuration Files can contain both dynamic and static application-to-host mappings. For static application-to-host mappings, an application will, by default, be started on a specified host. For dynamic application to-host mappings, the application will have a default host to start on but the Resource Manager FG42 will be queried at run-time to determine where the application actually should be placed. The Configuration Files FG56 also contain all information on how to start, stop, and configure an application, with the exception of environment variable settings for the application which are set based on the System Specification Files FG32.”

124. In fact, none of the cited passages from Welch in Dr. Zadok’s declaration include any form of the word “configure,” and the isolated reference to the Human Computer Interface (HCI) “provid[ing] information ... regarding the system configuration” refers merely to displaying the current state of the system—not to actively setting parameters or preparing applications for execution (it is worth noting that the clause in the challenged claim refers to configuring an

application and not a system). Dr. Zadok in his declaration (Ex 1003, ¶218, emphasis and marking original) states:

The program control can “start” and “configure” a program on a particular host. Ex 1005, Welch, 3. “The program control (PC) component consists of a central control program and a set of startup daemons. When resource manager needs to **start** a program on a particular host, it informs the control program, which notifies the startup daemon on that host.” EX1005, Welch, 3. “The Resource Manager needs to decide on which host(s) and LAN(s) the unhealthy sub-path needs to be **replicated or moved**.” EX1005, Welch, 3. “The HCI provides information to the user regarding the system configuration, application status, and reallocation decisions.” EX1005, Welch, 3. The software can also “shed a program” (“stop”) an application. Ex 1005, Welch, 2

125. As shown above, the emphasis is on the words “start”, “replicated or moved” in the excerpts from Welch. Notably absent is any identification of any variant of the word “configure” (and as related to an application). Instead, the spurious citation of the Human Computer Interface (HCI) of Welch as teaching the configuring of the M managed characteristic application.

126. Welch states “The HCI **provides information to the user regarding the system configuration**, application status, and reallocation decisions.” Ex 1005, 3. Although the quoted passage does contain the word “configuration” it describes that the HCI displays the information regarding the system’s current

configuration (and not an application as required by the clause in the challenged claim).

127. An application program configuration file is a generally a structured text file that defines key parameters and settings controlling how an application is deployed, started, operated, and terminated within a computing environment. It typically includes fields specifying the application's identity, the desired host(s) or environment(s) for execution, flags controlling automatic startup behavior, runtime host selection, and console requirements ('743 patent (col 27 and 28)).

128. As per '743 Patent information about applications is stored in a configuration files defined as "Configuration Files FG56— contain an ordered set of applications that can be loaded at the Program Control display and then either edited or executed. The Configuration Files can contain both dynamic and static application-to-host mappings. For static application-to-host mappings, an application will, by default, be started on a specified host. For dynamic application-to-host mappings, the application will have a default host to start on but the Resource Manager FG42 will be queried at run-time to determine where the application actually should be placed. The Configuration Files FG56 also contain all information on how to start, stop, and configure an application, with the exception of environment variable settings for the application which are set based on the System Specification Files FG32."

129. As described in '743 Patent 27:37-40 "A Configuration File is an ASCII file containing a list of applications. The format of an entry in a Configuration File is shown in Table 1 below.

TABLE 1

Application	TACFIRE: tacfire
Host	electra1
Display	umbriel1: 0.0
Auto_Start	0
RM_Start	0
Console	1
Time_Delay	1
StartupDir	"\$ENV_SIM_VERSION/TACFIREprocessor"
StartupExe	"\$ENV_SIM_VERSION/TACFIREprocessor/tacfire"
StartupArgs	"-disport \$DIS_PORT_NUM -cflhost % (HOSTNAME, AAW: Tactical_Sims:CFF_Broker)"
ShutdownExe	SIGINT
LogType	STDOUT
LogDir	"/usr/tmp"

130. Additionally as described in '743 Patent (col 27 and 28):

"The Configuration file advantageously can include the following fields:

- 1) The Application field, which identifies the full application name as defined in the System Spec. Files FG32 (i.e., System:Subsystem:Application).
- 2) The Host field, which is the desired or default host that this application should be started on.
- 3) The Display field, which is an optional field used when graphical display output from an application needs to be rerouted to a display on a different host.
- 4) The Auto_Start flag, which identifies whether the application is to be started automatically if the "Start All" action is selected by the operator from the Program Control Display. (If the flag were set to "1", then the application would be started. If the flag were set to "0," it would not be started.)

- 5) The RM_Start flag, which identifies whether the Resource Manager should be queried at run-time to determine what host the application should be started on. The valid values are “0” for “NO” and “1” for “YES”.
- 6) The Console flag, which identifies whether the application needs to be started in an Xterm window. The valid values are 0 for “NO” and 1 for “YES”.
- 7) The Time_Delay field, which identifies how many seconds to wait after the previous application has been started before starting this application.
- 8) The StartupDir field, which identifies the current working directory that is to be set prior to starting up the application. This directory is usually the same as the directory where the executable for the application resides but does not have to be. As this example shows, environment variables may be used in the path.
- 9) The StartupExe field identifies the entire path and name of the application executable.
- 10) The StartupArgs field, which contains all the argument values needed for this particular application. As this example indicates, the argument values can be dynamically set at run time if needed. Environment variables may also be used within the argument list. In this example, the % (UNIQUE, 1, 40, Isis) argument would yield a number from 1 to 40 which is unique within a context named “Isis”. Another resolution of % (UNIQUE, 1, 40, Isis) would yield a different number.
- 11) The ShutdownExe field, which identifies which signal defined within the application that program control is to use to shutdown this application. Some examples would be SIGINT, SIGTERM, or SIGKILL. A shutdown script can also be used to shutdown the application. (In that case, there would be ShutdownDir, ShutdownExe, and ShutdownArgs fields listed. The usage for the shutdown fields would be used exactly the same as the startup fields.)
- 12) The LogType field, which identifies which outputs are to be written to the specified log file. The valid values are STDOUT, STDERR, and LOG_ALL. STDOUT is the normal output of the application (stdout). STDERR is the error output

of the application (stderr). LOG_ALL writes both stdout and stderr outputs to the file.

- 13) The LogDir indicates the directory where the log file will be written. Again, environment variables may be used here. The log file name will be “<userid>_<appname>_<pid>.log” where <appname> is the full application name as specified in the Application field, <userid> is the userid of the current user under which the program control application is running, and <pid> is the system assigned process id of the application being executed.”

131. Thus, the term configuring an application program, in general, and as specifically defined in ‘743 Patent refers to the process of preparing an application to run properly on a host within a distributed computing environment. This process includes setting environment variables required by the application at runtime, specifying the working directory, and identifying the correct executable path and filename. It also involves supplying necessary command-line arguments, some of which may be dynamically resolved at runtime— for example, hostnames of dependent applications, the current date, or user-specific identifiers. Additionally, configuring may include determining whether the application should run within a terminal emulator (such as an Xterm) and defining the appropriate shutdown procedure, such as a specific script or signal. Beyond these basic setup tasks, specifying the resources the application is expected to consume (such as CPU, memory, or network bandwidth), and ensuring compatibility with the host’s operating system and hardware architecture.

132. Simply displaying the current configuration of a system is not “configuring ... the M managed characteristic application computer programs” as recited in the challenged claims. Instead, “configuring” involves a technical setup process that includes specifying things like runtime parameters, hardware requirements or determining execution behavior.

133. Based on the above analysis, Dr. Zadok’s declaration has not shown that Welch discloses the claimed limitation of “configuring ... the M managed characteristic application computer programs.” Critically, none of the cited passages in Welch use the term “configure,” nor do they describe the specific technical actions associated with configuration— such as setting environment variables, or specifying command-line arguments. Hence, Dr. Zadok’s analysis and the petition has failed to demonstrate that the referenced prior art discloses or suggests the limitation of “configuring ... the M managed characteristic application computer programs,” as required by the above clause.

5. Ravindran does not cure the shortcomings of Welch

134. The Petition does not allege that Ravindran discloses a second function group providing application-level event correlation capabilities or determining the state and health of distinct application programs. Nor does the Petition allege that Ravindran describes applying QoS requirements to individual RTCS application programs, as opposed to system-level resource management, or

performing the distinct step of configuring application programs. Instead, the only citation to Ravindran in the Petition is the statement that “Ravindran (Ex 1006) describes hardware monitors as follows: Hardware (HW) Monitors (which run on each host) periodically calculate load indices for hosts and LANs, and pass this information to the Hardware Data Repository (HDR).” See Ex 1006, 7; Ex 1003, ¶147; Petition, 38 (citations and emphasis original). This disclosure pertains solely to system resource monitoring—specifically, the calculation of host and network load metrics for hardware-level performance tracking. It does not involve any monitoring, evaluation, or management of application programs, nor does it disclose correlation of application-level events such as statuses, performance metrics, or internal states, as required by the challenged claims. Likewise, Ravindran does not describe the distinct act of configuring application programs.

135. Accordingly, Ravindran does not remedy the deficiencies in Welch, nor does it provide the missing limitations necessary to establish unpatentability of the challenged claims.

F. Ground 2: Claim 10 Would Not Have Been Obvious In View of Welch in combination with Ravindran and Hosokawa

136. The Petition’s reliance on Hosokawa does not cure the deficiencies in Welch and Ravindran, nor does it establish the unpatentability of the challenged claims. As the Petition itself concedes, Hosokawa is cited only as “additional evidence as to the obviousness of limitation 10.8” and to provide examples

involving numerical relationships between M and N (e.g., $M > N$, $M = N$, or $M < N$). See Petition, 79. Hosokawa is not cited for, and does not disclose, any of the missing limitations identified above, including application-level event correlation capabilities, determination of the state and health of distinct application programs, enforcement of QoS requirements at the application level, or the distinct act of configuring applications.

137. Moreover, the Petition's assertion that it would have been obvious to store Welch's QoS management software on one of the "N networked hosts" as taught by Hosokawa is irrelevant to the critical deficiencies in Welch's disclosures. The location of the QoS management software does not address whether Welch teaches or suggests the key functional requirements expressly recited in the challenged claims. Hosokawa's disclosure regarding software storage does not bridge the fundamental gaps in Welch's teachings concerning event correlation, application health monitoring, application-specific QoS management, or application configuration.

138. Accordingly, because Hosokawa does not address—let alone cure—any of the substantive shortcomings in Welch and Ravindran, the Petition fails to establish that the challenged claims are unpatentable.

VIII. Conclusion

139. For the foregoing reasons, it is my opinion that claim 10 of the '743 Patent would not have been obvious over the prior art in Petitioner's Petition for *inter partes* review.

140. Since Claims 11, 14, 16 and 18 are dependent claims corresponding to Claim 10, they would not have been obvious over the prior art in Petitioner's Petition for *inter partes* review for the same reasons as Claim 10.

141. All statements made herein of my own knowledge are true, and all statements made on information and belief are believed to be true. Further, I am aware that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001. I declare under penalty of perjury that the foregoing is true and correct.

Executed on April 29, 2025, in Gainesville, Florida.



Dr. Sanjay Ranka

