UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

ServiceNow, Inc.
Petitioner

v.

BMC Software, Inc.
Patent Owner

U.S. Patent No. 8,646,093
Filing Date: December 9, 2009
Issue Date: February 4, 2014

TITLE: METHOD AND SYSTEM FOR CONFIGURATION MANAGEMENT DATABASE SOFTWARE LICENSE COMPLIANCE

DECLARATION OF TAL LAVIAN, PH.D.

Table of Contents

Page

Table of Contents
(continued)

Page

Table of Contents
(continued)

Page

I, Tal Lavian, Ph.D., declare as follows:

1. I have been retained by counsel for ServiceNow, Inc. (Petitioner) in this case as an expert in the relevant art.

2. I have been asked to provide my opinions relating to claims 1, 5, 10-13, and 16 of U.S. Patent No. 8,646,093 to Myers et al. ("the '093 patent"), which I understand is owned by BMC Software, Inc.

I. BRIEF SUMMARY OF MY OPINIONS

3. Claims 1, 5, 10-13, and 16 purport to recite a method and system for managing software license compliance. They do not describe anything new or non-obvious when the earliest application for the '093 patent was filed in March 2009. As explained in detail in **Part VI** of this Declaration, the features described in these claims are nothing more than the combination of two known prior art technologies: (1) a system for determining compliance with software license contracts; and (2) a configuration management database (CMDB) for storing information about software assets. Each of these features is described or suggested by Meyer (Ex. 1003) and Best Practice (Ex. 1004). Because claims 1, 5, 10-13, and 16 do not recite anything inventive or non-obvious, and each limitation is disclosed or suggested by the prior art as described below, each of those claims is obvious. The bases for my opinions are set forth below.

II. INTRODUCTION AND QUALIFICATIONS

A. Qualifications and Experience

4. I possess the knowledge, skills, experience, training and the education to form an expert opinion and testimony in this case. A detailed record of my professional qualifications, including a list of patents and academic and professional publications, is set forth in my curriculum vitae attached to this declaration as **Exhibit A**.

5. I have more than 25 years of experience in the networking, telecommunications, Internet, and software fields. I received a Ph.D. in Computer Science from the University of California at Berkeley in 2006 and obtained a Master's of Science ("M.Sc.") degree in Electrical Engineering from Tel Aviv University, Israel, in 1996. In 1987, I obtained a Bachelor of Science ("B.Sc.") in Mathematics and Computer Science, also from Tel Aviv University.

6. I am currently employed by the University of California at Berkeley and was appointed as a lecturer and Industry Fellow in the Center of Entrepreneurship and Technology ("CET") as part of UC Berkeley College of Engineering. I have been with the University of California at Berkeley since 2000 where I served as Berkeley Industry Fellow, Lecturer, Visiting Scientist, Ph.D. Candidate, and Nortel's Scientist Liaison, where some positions and projects were done concurrently, others sequentially.

7. I have more than 25 years of experience as a scientist, educator and technologist, and much of my experience relates to computer networking technologies. For eleven years from 1996 to 2007, I worked for Bay Networks and Nortel Networks. Bay Networks was in the business of making and selling computer network hardware and software. Nortel Networks acquired Bay Networks in 1998, and I continued to work at Nortel after the acquisition. Throughout my tenure at Bay and Nortel, I held positions including Principal Scientist, Principal Architect, Principal Engineer, Senior Software Engineer, and led the development and research involving a number of networking technologies. I led the efforts of Java technologies at Bay network and Nortel Networks. In addition, during 1999-2001, I served as the President of the Silicon Valley Java User Group with over 800 active members from many companies in the Silicon Valley.

8. Prior to that, from 1994 to 1995, I worked as a software engineer and team leader for Aptel Communications, designing and developing mobile wireless devices and network software products. From 1990 to 1993, I worked as a software engineer and team leader at Scitex Ltd., where I developed system and network communications tools (mostly in C and C++).

9. I have extensive experience in the area of network communications and Internet technologies including design and implementation of computer-based

systems for managing communications networks, including the ability to monitor and provision networks. While with Nortel Networks and Bay Networks (mentioned above) my work involved the research and development of these technologies. For example, I wrote software for Bay Networks and Nortel Networks Web based network management for Bay Networks switches. I developed Simple Network Management Protocol (SNMP) software for Bay Network switches and software interfaces for Bay Networks' Optivity Network Management System. I wrote software for Java based device management including software interface to the device management and network management for the Accelar routing switch family network management system.

10. I have extensive experience in network communications, including control and management of routing and switching architectures and protocols in layers 1-7 of the OSI model. Much of my work for Nortel Networks (mentioned above) involved the research and development of network communications technologies. For example, I wrote software for Bay Networks and Nortel Networks switches and routers, developed network technologies for the Accelar 8600 family of switches and routers, the OPTera 3500 SONET switches, the OPTera 5000 DWDM family, and the Alteon L4-7 switching product family. In my lab, I installed, configured, managed and tested many network communications equipment of competitors such as Cisco Systems, Juniper Networks, Extreme

Networks, Lucent and Alcatel.

11. I am named as a co-inventor on more than 80 issued patents and I co-authored more than 25 scientific publications, journal articles, and peer-reviewed papers. Furthermore, I am a Senior Member of the Institute of Electrical and Electronics Engineers ("IEEE").

12. I currently serve as a Principal Scientist at my company Telecomm Net Consulting Inc., where I develop network communication technologies and provide research and consulting in advanced technologies, mainly in computer networking and Internet technologies. In addition, I serve as a Co- Founder and Chief Technology Officer (CTO) of VisuMenu, Inc., where I design and develop architecture of visual IVR technologies for smartphones and wireless mobile devices in the area of network communications. The system is based on cloud networking and cloud computing utilizing Amazon Web Services.

13. Additional details of my background are set forth in my curriculum vitae, attached as **Exhibit A** to this Declaration, which provides a more complete description of my educational background and work experience. I am being compensated for the time I have spent on this matter. My compensation does not depend in any way upon the outcome of this proceeding. I hold no interest in the Petitioner (ServiceNow, Inc.) or the patent owner (BMC Software, Inc.).

B. Materials Considered

14. The analysis provided in this Declaration is based on my education and experience in the field of computer systems and service management tools, as well as the documents I have considered including U.S. Patent No. 8,646,093 ("'093 patent") **[Ex. 1001]**. The '093 patent states on its face that it issued from an application filed on December 9, 2009. The '093 patent also claims priority to and incorporates by reference U.S. Provisional Application Ser. No. 61/165,505 filed on March 31, 2009, which I have reviewed. ('093, Ex. 1001, 1:9-12; *see also* Ex. 1007 (provisional application).) For purposes of my analysis, I have assumed March 2009 as the priority date for the '093 patent.

15. The prior art documents I rely upon in this Declaration are:

Exhibit	Description of Document
1003	U.S. Patent No. 6,810,389 B1 to Marc A. Meyer
1004	Excerpts from Best Practice for Software Asset Management, IT Infrastructure Library (ITIL) (2003)
1005	Excerpts from Rob Addy, *Effective IT Service Management, to ITIL and Beyond!* (2007)
1006	U.S. Patent Application Publication No. 2005/0071276 A1 to Joachim Bruchlos et al.

III. PERSON OF ORDINARY SKILL IN THE ART

16. I understand that an assessment of claims of the '093 patent should be undertaken from the perspective of a person of ordinary skill in the art as of the earliest claimed priority date, which I understand is March 31, 2009.

17. In my opinion, a person of ordinary skill in the art as of March 2009 would have possessed at least a bachelor's degree in computer science (or equivalent degree or experience) with at least four years of practical experience or coursework in the design or development of systems for management of network-based systems and network management databases, such as configuration management databases (CMDBs). Such a person would also have general familiarity with service management tools, software license contracts and techniques for ensuring compliance with those licenses. As I will explain below, all of these areas were well-developed and mature well before March 2009.

18. My opinions regarding the level of ordinary skill in the art are based on, among other things, my over 25 years of experience in the field of network communications, computer science and engineering, my understanding of the basic qualifications that would be relevant to an engineer or scientist tasked with investigating methods and systems in the relevant area, and my familiarity with the backgrounds of colleagues and co-workers, both past and present.

19. Although my qualifications and experience exceed those of the

hypothetical person having ordinary skill in the art defined above, my analysis and opinions regarding the ’093 patent have been based on the perspective of a person of ordinary skill in the art as of March 2009.

IV. STATE OF THE ART OF THE RELEVANT TECHNOLOGY AT THE TIME OF THE ALLEGED INVENTION

20. The ’093 patent generally discloses a method and system for monitoring compliance with software license contracts. In this section, I provide a brief background of the state of software license contract compliance technology prior to March 2009 pertinent to the ’093 patent.

A. Managing Software License Compliance

21. By March 2009, computer software had become a common fixture of everyday life, and integral to the functioning of most business enterprises. Computer software has long been made available to customers pursuant to a contract known generally as a “software license,” which governs the customer’s use of the software. A software license may specify, among other things, the number of users within an enterprise who are permitted to use the licensed software (including the number of users who may use the software concurrently), or how long the licensed software may be used before the license must be renewed. Software license contracts can also be one component of a larger service contract with the provider, and thus, can include a number of complex provisions.

22. It is important for a number of reasons for an enterprise to comply with the terms of software license contracts. Violating a software license contract could not only expose the enterprise to liability for breach of contract, but in some

cases, civil or criminal liability for copyright infringement. (Best Practice for Software Asset Management (2003) ("Best Practice"), Ex. 1004, at 20-21.) Although violations of software license contracts are often unintentional, those violations can still carry significant consequences for the enterprise. (*Id.* at 21.)

23. However, monitoring compliance with software licenses is not always a straightforward endeavor. Larger enterprises may have hundreds of software products governed by different software license contracts, each presenting varying terms. Additionally, software products "typically have complex legal conditions that can be misunderstood even by people working in the area." (*Id.* at 20.) To respond to these concerns, industry has developed a variety of techniques for monitoring and managing compliance with software license contracts. Some of these techniques involve common sense processes of taking an inventory of installed software and comparing it against the governing software license contracts, and taking corrective action if violations are found.

B. Using Configuration Management Databases to Manage Software

24. As the number and variety of software licenses increased, industry recognized a need for a more automated and computer-assisted approach to manage software assets and compliance with software license contracts. One solution was to use a database known as a **"Configuration Management Database**" or "**CMDB**," to keep track of these licenses. Generally speaking,

"CMDB" is an industry-standard term referring to a database that stores information about the Information Technology ("IT") assets used by an enterprise, such as servers, workstations, software programs, documentation, and other computing resources. The CMDB contains a series of records, known as "**configuration items**" or "**CIs**," for storing information about the IT assets.

25. The Background of the '093 patent acknowledges that CMDBs and CIs are not an invention of the patent. ('093, 1:18-42.) The Background further acknowledges that CMDBs "are emerging as a prominent technology for Enterprise Management software." ('093, 1:24-26.) "The CMDB serves as a point of integration between various IT management processes," including software asset management, which is a "core component of an overall asset management policy." ('093, 1:39-40; 1:49-51.) The patent acknowledges that "[o]ne kind of CI that may be managed in a CMDB is a software asset." ('093, 1:43-44.)

26. Many of the processes and techniques for managing software assets using CMDBs were described in *Best Practice for Software Asset Management* (2003), a well-known publication in the field published by the IT Infrastructure Library ("ITIL"). (Ex. 1004 ("Best Practice").) ITIL is a division of the Office of Government Commerce of the United Kingdom government. (*Id.* at 1 (under "The IT Infrastructure Library").) The publications disseminated by ITIL, such as Best Practice, provide industry standards and practices for managing IT assets. ITIL's

standards are often considered authoritative by many persons of ordinary skill in the art. For example, the entire Background section of the '093 patent is devoted to discussing and discussing ITIL CMDBs and ITIL-defined processes for management of software assets. ('093, 1:18-2:8.)

27. As its name implies, the Best Practice publication defines a set of preferred processes for managing software assets. (Best Practice, Ex. 1004, at p. xi (stating that ITIL "is the most widely accepted approach to IT Service Management in the world.").) Best Practice provides a comprehensive guide to management of software assets, including guidance on how to establish processes to track and monitor software license compliance. (Best Practice, pp. 50-51, § 5.4.) "Licensing compliance processes are responsible for ensuring that the use of all software within the organisation remains within all legal and contractual terms and conditions." (*Id.* at p. 52, § 5.4.2.) Best Practice also provides an exemplary CMDB/CI schema for storing information about the organization's software assets and their corresponding software license contracts. (*Id.* at 121-23, Appendix D.)

V. THE '093 PATENT'S TECHNIQUE FOR SOFTWARE LICENSE COMPLIANCE

A. The Specification

28. The '093 patent generally describes a method and a system to "monitor and verify software license compliance in an enterprise." ('093, Ex. 1001, 2:66-67.) The patent discusses managing software license compliance by using databases and modeling to compare the deployment of software within an enterprise to the enterprise's software license contracts. ('093, Abstract.)

29. Figure 2 provides a general overview of one embodiment of the computer-implemented method and system described in the '093 patent:

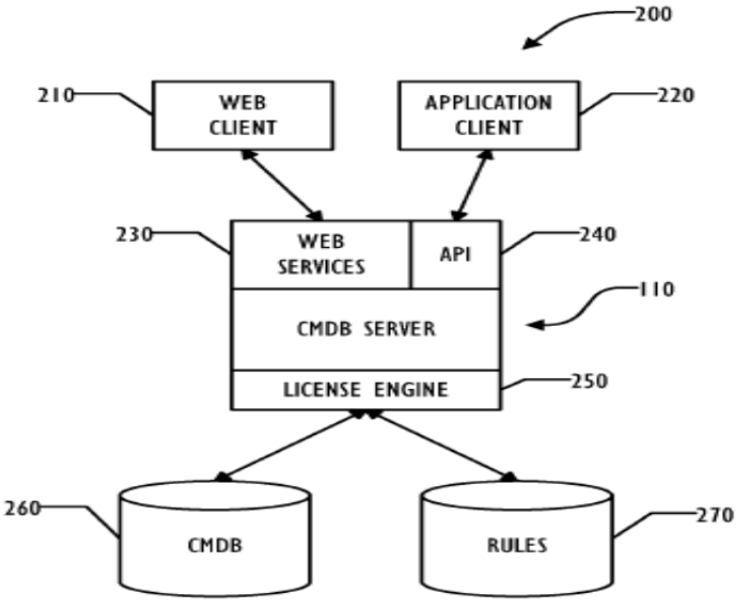


FIG. 2

('093, Fig. 2.)

30. Figure 2 includes management system **200** that has a Configuration Management Database ("CMDB") **260**. As explained in the Background and noted above, the CMDBs were well-known and "a prominent technology for Enterprise Management Software." ('093, 1:18-26.) A CMDB "contains data about managed resources known as Configuration Items (CIs)." ('093, 1:29-30.).

31. The patent uses these configuration items (CIs) to store information about software products and their associated software contracts. "Information about the software contracts is stored as CIs in the CMDB datastore **260** [of Figure 2] using one or more of the clients **210**/**220**." ('093, 5:1-3.)

32. The management system in Figure 2 also includes a license datastore **270**. Although shown as a separate database in Figure 2 above, license datastore **270** "may be integrated with the CMDB datastore **260**." ('093, 4:11-13.) "The license datastore **270** provides storage for [sic] to model software contracts, including rules against which the CIs are evaluated for software license compliance and other information necessary for processing those rules." ('093, 4:13-17.) The '093 patent identifies at least two types of information used to evaluate compliance with a software license contract: (1) information about the **software license contract**, and (2) a **license certificate** corresponding to the license.

33. First, configuration information about a **software license contract** may be stored in the CMDB, including the term of the license, its current status

(draft, executed, expired, etc.), the company and other information associated with the contract. ('093, 5:10-56 (Table 1).) The CMDB may also provide a number of pre-defined "license types" that can be essentially used as templates in identifying the characteristics of a software license contract. ('093, 6:1, 6:33-35.) Exemplary "license types" include "enterprise," "site," and "per instance" software licenses (among others), each having certain pre-defined characteristics. ('093, 6:40-55 (Table 2).) The user can also create new license types, if needed. ('093, Fig 4 (step 420).) As shown in **Part V.B** below, this information relating to the software license contract is part of the "**first model**" recited in the claims.

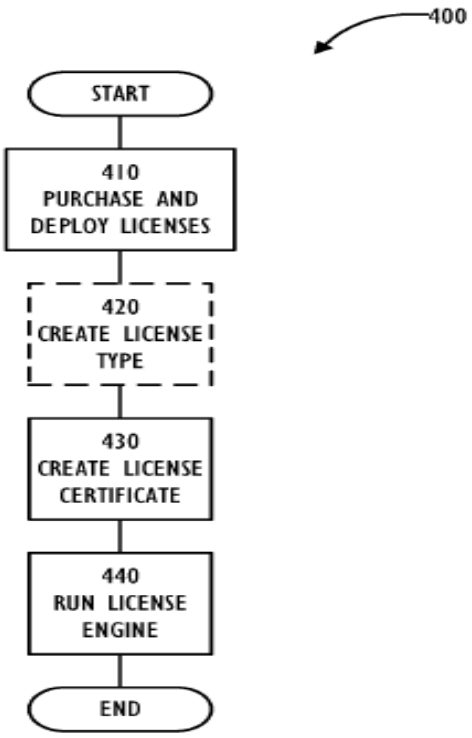
34. Second, the '093 patent describes a "**license certificate**" that is linked to its corresponding software license contract. ('093, 3:1-2.) "A license certificate indicates the right to deploy software in the environment managed by the CMDB server **110**." ('093, 8:59-63.) A "license certificate" may comprise a variety of information relating to the right to use the software, including the license category (client, server, mainframe), effective date and expiration date, among other information. ('093, 9:1-20 (Table 3).) The system may ask the user to enter additional information such as "how many licenses were purchased and how many copies per device are allowed under each license." ('093, 9:35-36.) "Other questions may be asked depending typically on the license type. The additional information supplied in response to those questions may be included in the license

certificate as it is stored in the license datastore **270**." ('093, 9:36-40.) This license certificate information is part of the "**second model**" in the claims.

35. Once these two categories of information – **software license contract** and **license certificate** information – are compiled, the system uses them to evaluate the status of the licenses:

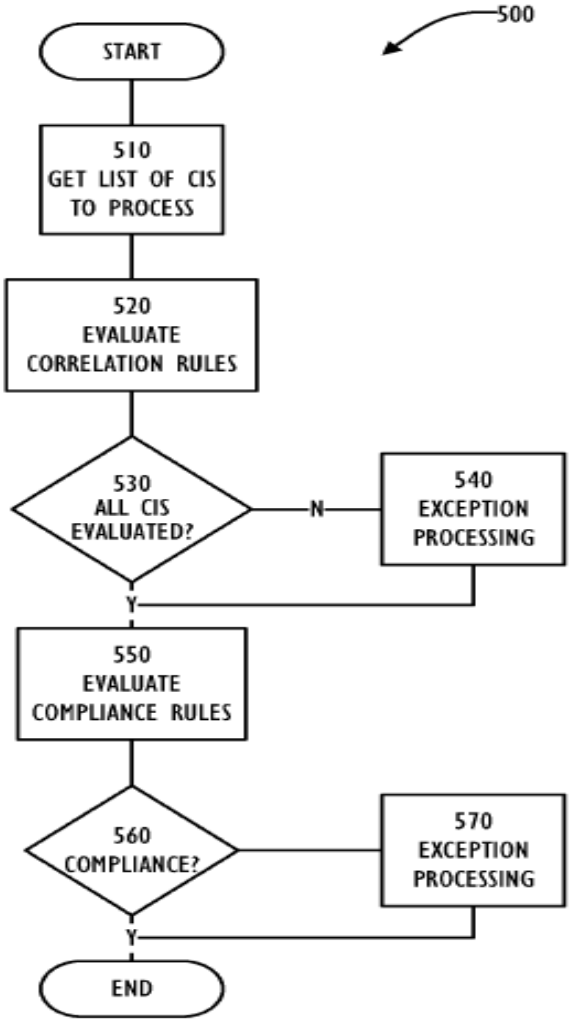
> FIG. **4** is a flowchart illustrating a technique **400** for preforming software license compliance monitoring and verification. . . . [A]fter the license certificates are created, then in block **440**, the license engine **250** is run. The license engine evaluates the status of the software licenses modeled in the CMDB **260** against the license certificates created in block **430**.

('093, 4:58-59, 10:28-32.)



('093, Fig. 4.)

36. One exemplary process for comparing the licenses against the certificate is shown in Figure 5. ('093, Fig. 5.) That process involves, among other steps, evaluation of "compliance rules" in order "to determine whether each of the software CIs complies with the terms of the software contract. In block **560**, if any CI is not in compliance, then any desired exception processing may be performed." ('093, 10:49-53.) This exception processing "in one embodiment may be simply producing an error message or report indicating the exception." ('093, 10:55-56.) "Where the organization is not in compliance, the license engine identifies the non-compliance and provides information that may allow the contract or asset manager to address the problems and bring the organization into compliance." ('093, 13:20-24.)



('093, Fig. 5.)

37. The specification provides a specific example of evaluating compliance with a "per copy per device license."

> In FIG. 6, a graph **600** illustrates compliance with a per copy per device license. Company **610** has purchased two licenses for some software, which is used at two different sites (**620** and **622**) by four users. In this example, the software contracts do not limit the number of users that may use a given copy of the licensed software. . . . The license engine **250** connects certificate **670** to contract **660**. Because

> the 2 instances of the software found installed in the CMDB **260** match the 2 licenses purchased by company **610**, only 2 licenses are in use, and the company **610** complies with software license contract **660**.

('093, 10:28-42.)

B. The Claims of the '093 Patent

38. The two independent claims addressed in this Declaration—i.e., claims 1 and 16—purport to recite a method and a system, respectively, for managing software license contracts. The first independent claim addressed in this Declaration is claim 1, which recites:

> 1. A computer-implemented method, comprising:
> **[a]** modeling deployment of a software product and a software license contract for the software product;
> **[b]** storing a first model of the modeled deployment of the software product in a configuration management database (CMDB) by storing information related to the software product as a first configuration item in the CMDB and by storing information related to the software license contract as a second configuration item in the CMDB;
> **[c]** storing a second model of the modeled software license contract for the software product in a license database by generating a license certificate corresponding to the software license contract and storing the license certificate in the license database; and
> **[d]** evaluating the deployment of the software product for

compliance with the software license contract, comprising:

[d][1] connecting and comparing the first model and the second model by comparing the first configuration item with the license certificate and connecting the license certificate with the second configuration item responsive to comparing the first configuration item with the license certificate; and

[d][2] generating an exception indication if the act of comparing the first model and the second model indicates non-compliance with the software license contract.

('093, 13:44-14:3 (Claim 1).) I added bracketed notations (e.g., "[a]," "[b]," etc.) for the purpose of identifying these limitations in my Declaration. Claims 5, 10, 11, 12, and 13 depend from independent claim 1 listed above. I address each claim in more detail in **Part VI** below. The second independent claim addressed in this Declaration is claim 16, which generally recites a system claim for performing the method of independent claim 1 discussed above.

C. Claim Construction

39. I have been informed by counsel that invalidity analysis is a two-step process. In the first step, the scope and meaning of a claim is determined by construing the terms of that claim. In the second step, the claim as interpreted is compared to the prior art. Therefore, before I address the application of the prior art to the claims of the '093 patent in **Part VI** below, I provide constructions for certain terms in those claims.

40. I have also been informed by counsel that a claim in an unexpired patent subject to *inter partes* review must be given its "broadest reasonable construction in light of the specification of the patent in which it appears," which is different from the manner in which the scope of a claim is determined in litigation. I apply the "broadest reasonable construction" standard in my analysis below.

1. "license certificate"

41. The term "license certificate" is recited multiple times in independent claim 1. In my opinion, the broadest reasonable construction of "**license certificate**" to one of ordinary skill in the art is "**information relating to the right to deploy software**." I derive this definition from the following passage of the specification:

> After any new license types are created to handle the terms of the new software contracts terms, license certificates may be created in block

> **430**, to link software contracts to CIs. A license certificate indicates the right to deploy software in the environment managed by the CMDB server **110**. In one embodiment, a license certificate comprises the information listed in Table 3 below.

('093, 8:59-65 (underlining added).) Table 3 in the specification provides an exemplary license certificate that includes a number of fields or attributes including a summary description of the certificate, the expiration date of the license, and other fields. ('093, 9:1-20 (Table 3, reproduced below).)

TABLE 3

License Certificates

Field name	Description
Company Software Contract ID	This information comes from the software contract.
Certificate ID	The certificate ID identifies the license certificate in listings and reports. It does not have to be unique.
Summary	This field provides additional space to describe the certificate.
Status	Initially set to Draft.
License Category Type	Select from Client, Server, or Mainframe.
License Type	The appropriate license type. The license type determines the connection questions and the compliance questions.
Cost Center	This information comes from the software contract, but can be changed.
Effective Date	The date that the license becomes effective.
Expiration Date	The date that the license expires. If the license does not expire, this field may be left blank.

42. Based on my analysis of the specification, I believe the broadest reasonable construction of "**license certificate**" to one of ordinary skill in the art is "**information relating to the right to deploy software**," which is derived from the underlined sentence quoted above. ('093, 8:61-63 ("A license certificate indicates the right to deploy software in the environment managed by the CMDB

server **110**.") (underlining added).) I note that my construction removes a portion of that sentence, "in the environment managed by the CMDB server **110**," because claim 1 does not recite a server, let alone a "CMDB server." They do recite a CMDB itself, but do not require a CMDB server.

2. "model" and "modeling"

43. The terms "model" and "modeling" are used throughout the claims addressed in this Declaration. Claim 1, for example, recites the step of "modeling deployment of a software product and a software license contract for the software product," and storage of a "first model" and a "second model" containing information about a software product and its software license contract. The broadest reasonable construction of "**model**" is "**an organized collection of information about an object**," and in the verb form "**modeling**," that simply refers to "**creating a model**."

44. The specification does not define "model" but provides examples of how a software license contract can be modeled. ('093, 2:13-17.) For example, the specification explains that "the CMDB server **110** may model the software product packages or components installed on each of the computer systems **120**, as well as the software contracts under which that software is licensed." ('093, 3:42-45 (underlining added).) Table 1 of the '093 patent provides an illustration of such a model. ('093, 5:3-5.) It includes a number of fields for a particular software

license including the ID, Summary, Term, Status, Expiration Date, and other fields. ('093, 5:10-57.) This example shows that creating a "model" of an object, such as a software license contract, involves storage of an organized collection of information about that object. Although the term "model" in other contexts has been used to describe a mathematical or graphical representation of an object, the specification does not require any such representation.

45. In my opinion, therefore, the broadest reasonable construction of "**model**" is "**an organized collection of information about an object**," and the verb "**modeling**" means "**creating a model**."

3. "exception indication"

46. The term "exception indication" appears in the final limitation of claim 1, which recites the step of "generating an exception indication if the act of comparing the first model and the second model indicates non-compliance with the software license contract." In my opinion, the term "**exception indication**" under its broadest reasonable construction is an "**indication of a condition or warning**."

47. Persons of ordinary skill in the art often use the word "**exception**" to refer to a problem or error in the operation of a computer program or system. Exceptions can cause a program or system to cease functioning, or can cause the program or system to perform other operations based on exception handling. Many persons who have used the Windows operating system may recognize the

infamous "Blue Screen" that would commonly report that a "fatal exception error" occurred and caused the software to stop functioning.

48. The '093 patent does use the term "exception" to refer to an error or problem, but more broadly describes the term as including warnings that occur prior to the occurrence of the actual error or problem. For example, the written description explains that an exception indication may be generated as a warning prior to non-compliance with the software license contract:

> Exception processing as performed in blocks **540** and **570** in one embodiment may be simply producing an error message or report indicating the exception. The exception may indicate a non-compliance condition. In one embodiment, additional exceptions may be triggered when a software license contract is in compliance but is within a window of near non-compliance, such as when nearly all of the purchased licenses have been deployed. Such warning indications may allow preventative measures to be taken before a non-compliance condition exists.

('093, Ex. 1001, 10:52-63 (underlining added).)

49. As the excerpt above explains, an exception "may indicate a non-compliance condition," but may also be triggered even when the system is "in compliance." In fact, the '093 patent refers to "non-compliance exceptions," which suggests that "exceptions" are not simply limited to non-compliance conditions. ('093, e.g., 10:66, 11:20.) In my opinion, therefore, the broadest

construction of "**exception indication**" is an "**indication of a condition or warning**."

50. The following table identifies the above claim constructions:

Claim Term	Broadest Reasonable Construction
"license certificate"	"information relating to the right to deploy software"
"model"	"an organized collection of information about an object"
"modeling"	"creating a model"
"exception indication"	"indication of a condition or warning"

VI. APPLICATION OF THE PRIOR ART TO THE CLAIMS OF THE '093 PATENT

51. I have been asked to provide my opinions as to whether certain prior art discloses and renders obvious the limitations recited in claims 1, 5, 10-13, and 16. I have grouped my opinions into two groups based on the prior art references to which I have referred.

52. First, with respect to claims 1, 5, 10 and 16, in my opinion each limitation is disclosed or suggested by Meyer (Ex. 1003) in view of Best Practice (Ex. 1004) and Addy (Ex. 1005). Second, with respect to dependent claims 11-13, each limitation is disclosed or suggested by Meyer, Best Practice and Addy, but in further view of Bruchlos (Ex. 1006). Counsel has informed me that each of these references qualifies as prior art to the '093 patent. Before applying the references, I will provide a brief summary of them.

A. Brief Summary of the Prior Art Applied in this Declaration

1. Meyer [Ex. 1003]

53. Meyer, U.S. Patent No. 6,810,389, entitled "System and Method for Flexible Packaging of Software Application Licenses," discloses a software licensing system (SLS) for determining whether or not a requested use of licensed software will violate the governing software license contract. (Meyer, Ex. 1003, Abstract.) I am informed by counsel that Meyer properly qualifies as prior art because it issued on October 26, 2004, before the earliest filing date of the '093

patent. I have relied on Meyer for certain aspects of claim 1, including among others the generation of a "**license certificate**" and the step of "**comparing**" the first and second models.

54. Meyer discloses a licensing system that determines whether or not to allow a client computer system to use licensed software in response to a request from the client. Most of the disclosures from Meyer that I rely upon in this Declaration are actually contained within Meyer's discussion of the prior art. Figure 1 of Meyer discloses a prior art software licensing system (SLS):

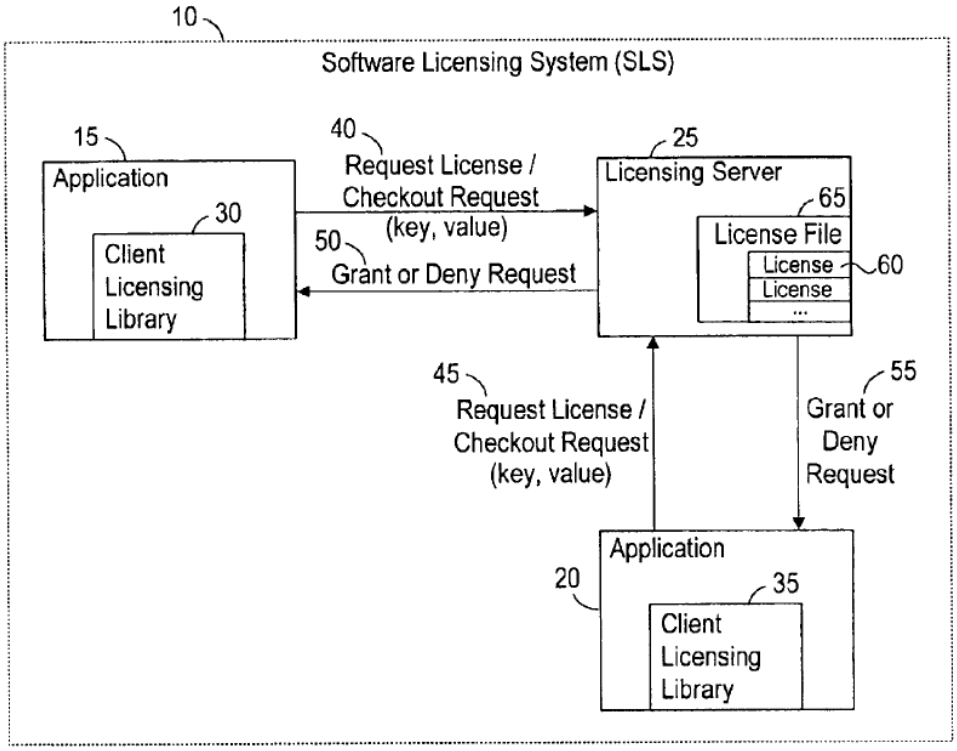


Fig. 1 - Prior Art

(Meyer, Ex. 1003, Fig. 1.)

55. The prior art software licensing system **10** in Figure 1 includes a

licensing server **25** that stores a licensing database or file **65** that stores information about software licenses **60**. An application program **15** issues a request **40** to the licensing server **25** to use a particular feature. "This request is referred to as performing a 'checkout' and is typically performed over a secure channel." (Meyer, 1:47-49.) The licensing server **25** responds to the request by either granting or denying access to the software. (Meyer, 1:56-59.)

56. The license file **65** shown in Figure 1 stores records about individual software licenses. Each record, as I explain in more detail below, can qualify as the claimed "**license certificate**." Figure 3 shows the contents of one such record:

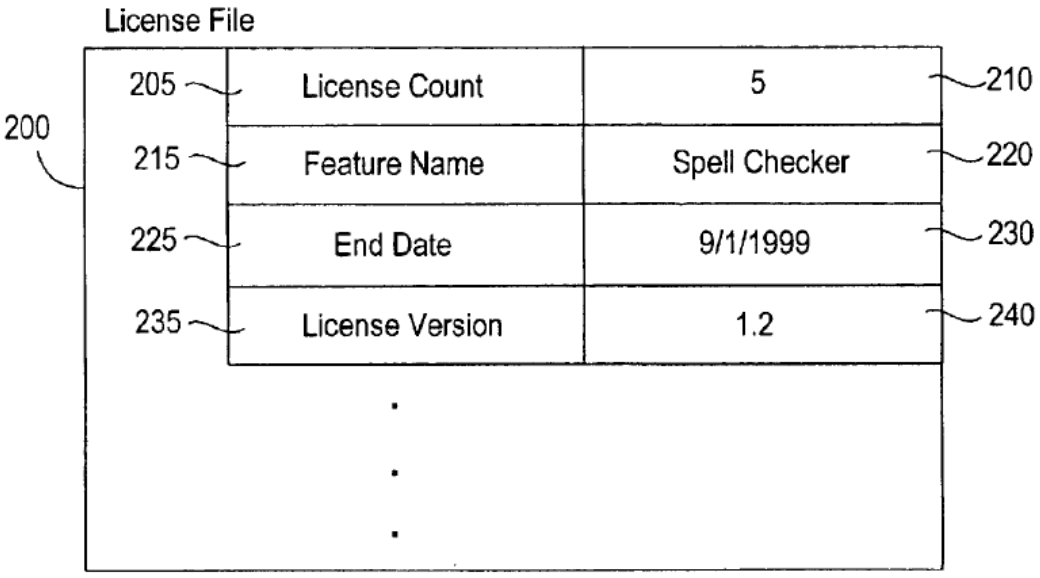


Fig. 3 - Prior Art

('093, Fig. 3.)

57. The licensing record in Figure 3 contains a number of attributes and

values, including License Count **205**, Feature Name **215**, End Date **225** and License Version **235**. When a client computer issues a request to use the licensed software, the licensing server compares the information about the software (which is carried in the request) against the attributes in the license record. ('093, 2:8-23.) For example, if a client computer requests to use the "Spell Checker" feature in Figure 3, the license server **25** compares the feature name and version in the license request against the name and version in the license record, and ensures that the date of the request is before "End Date" of September 1, 1999. ('093, 2:13-23.)

58. In the event the requested use of the software is not permitted under the software license contract, the software license system can inform the operator and take further action depending on the nature of the enforcement policy in place. ('093, 7:62-8:19.) "For example, if a lenient enforcement technique is implemented, and if a license request for a designated program is denied, then the violating computer system can display a simple warning to the user while still allowing the designated program to be run on the computer system." ('093, 8:6-11.) On the other hand, "[i]f strict enforcement is provided, and a license request is denied, the computer system **700** may immediately cause the designated program to quit on the computer system and not allow the program to be activated and used as long as the license policy is violated." ('093, 8:11-16.)

2. Best Practice [Ex. 1004]

59. *Best Practice for Software Asset Management* (2003) was published by the IT Infrastructure Library ("ITIL"). (Ex. 1004 ("Best Practice").) ITIL is a division of the Office of Government Commerce of the United Kingdom government. (*Id.* at 1 (under "The IT Infrastructure Library").) ITIL publications are well-known and provide industry standards and practices for managing IT assets. The entire Background section of the '093 patent, in fact, is devoted to describing ITIL standards, including the use of configuration management databases (CMDBs) and configuration items (CIs). ('093, 1:18-2:8.) I am informed that Best Practice qualifies as prior art because it was published in 2003, before the earliest application for the '093 patent.

60. I have relied on Best Practice for claim limitations relating to storing software and software license information in Configuration Items (CIs) in a Configuration Management Database (CMDB). As its name implies, Best Practice publication defines a set of preferred processes for managing software assets in an organization and is generally acknowledged as an industry standard reference on this topic. (Best Practice, Ex. 1004, at p. xi (stating that ITIL "is the most widely accepted approach to IT Service Management in the world.").) These best practices were already widely known and utilized by the industry at least by 2003, over six years before the time of the purported invention of the '093 patent (2009). The year

2003 is not the time of the conception of these practices, but rather the time when a mature and acceptable technology was standardized and recognized as the best practice among industry players. The contents of Best Practice were developed and reviewed in association a wide range of well-known companies, including Microsoft, IBM and HP. (Best Practice, Ex. 1004, at p. ix.) Best Practice also provides an exemplary CMDB/CI schema for storing information about software products and their corresponding license contracts. (*Id.* at 121-23, Appendix D.)

3. Addy [Ex. 1005]

61. *Effective IT Service Management, To ITIL and Beyond!* (2007), by Rob Addy, is a textbook providing practice guidance on various aspects of IT service management. I am informed that Addy qualifies as prior art because it was published in 2007, before the earliest application for the '093 patent. Springer has a reputation as an excellent publisher with a long process of selecting, writing and carefully reviewing the quality of the materials they publish.

62. My Declaration relies on Addy for its discussion of the advantages and uses of CMDBs, and also to provide a further express motivation to combine Meyer and Best Practice. As I will explain below, Meyer discloses a software license system but does not expressly disclose a CMDB or the storage of information about software products or licenses in CIs, for which I have cited Best Practice. Addy provides a further express motivation to incorporate the CMDB

from Best Practice into the system of Meyer by explaining the distinct benefits of using a CMDB. (Addy, Ex. 1005, at pp. 237, 238, §§ 19.8.4, 19.8.8.)

4. Bruchlos [Ex. 1006]

63. Bruchlos, entitled "Method for Automatic Creation and Configuration of License Models and Policies," discloses a system for generating software licenses based on particular license types. (Bruchlos, Ex. 1006, Abstract.) I am informed that Bruchlos qualifies as prior art because it was published on March 31, 2005, before the earliest application for the '093 patent.

64. My Declaration relies on Bruchlos for dependent claims 11-13 relating to selection of a "license type" for the software license contract. Bruchlos discloses the ability to provide a variety of different types of licenses based on the needs of the customer. (Bruchlos, Abstract, ¶¶ 0041-0048.)

B. All Limitations of Claim 1 Are Disclosed or Suggested by Meyer in view of Best Practice and Addy

65. The preamble of claim recites: "A computer-implemented method, comprising," followed by the steps in the body of claim 1. As explained below, the prior art discloses a computer-implemented method that meets each limitation.

(a) "modeling deployment of a software product and a software license contract for the software product" (Claim 1[a])

66. Best Practice discloses the recited modeling step. As shown below, the step of "**modeling**" takes place through the creation of a configuration management database (CMDB) including at least a configuration item (CI) for a software product, and another CI for the software license contract for the product.

67. As noted previously, the Background section of the '093 patent acknowledges that CMDBs and CIs were well-known in the art, and acknowledges that "CMDBs are emerging as a prominent technology for Enterprise Management Software." ('093, 1:24-26.) The specification further acknowledges that "[o]ne kind of CI that can be managed in a CMDB is a software asset." ('093, 1:43-44.) The patent therefore indicates that the "modeling" limitation of claim 1[a] is satisfied by using known CMDBs to store information about software products and software license contracts, which is disclosed by Best Practice.

68. Best Practice describes a series of processes collectively referred to as

Software Asset Management (SAM) for monitoring deployment of software programs. Best Practice "recommends the use of a DSL [definitive software library] and a CMDB for the management of software assets." (Best Practice, Ex. 1004, at 81 (underlining added).) Best Practice explains that "each component contained within the CMDB is referred to as a Configuration Item (CI). Each CI record within the CMDB contains all of the attributes and information relating to a component necessary for managing it, whether software, hardware, contracts, etc." (Best Practice, Ex. 1004, at 81-82 (underlining added).)

69. Appendix D of Best Practice provides an example of how information about the deployment of software assets, and the corresponding licenses, can be stored in a "SAM database" such as a CMDB. (*Id.* at 119-25.) "This Appendix gives suggestions for the possible physical storage contents and corresponding electronic databases for SAM [software asset management]." (*Id.* at 119.) Best Practice explains that a "SAM database" contains information about "the management of all software assets," and "could form part of an overall CMDB." (*Id.* at 101 (definition of "SAM database") (underlining added).)

70. Best Practice provides two tables that show the claimed step of modeling deployment of software products and license contracts. First, **Table D.2** in Best Practice discloses an "Installed software inventory" CI with more than a dozen attributes about the software products installed in the enterprise, including

"Product," "Version," "Installation status," and "Licence used," the latter attribute linking to the license that governs use of the software. (*Id.* at 123.)

71. The second table shown in **Table D.1** (*id.* at 119-122) of Appendix D provides an exemplary a "Software licence inventory" CI that includes more than 40 attributes relating to software licenses, including "Licence status and counts" (*id.* at 119), "Licensing basis" such as "per PC," "per device," "per user," etc. (*id.* at 122), "Transferability" information about the license (*id.*), "Source references" to specific the documentation of the licensing terms (*id.*), and many other attributes.

72. As noted in **Part V.C.2** above, the term "modeling" in claim 1 refers to creating an organized collection of information about an object. The capabilities in Best Practice, including storing CIs for software products (Table D.2) and their corresponding software license contracts (Table D.1), satisfy the step of "modeling deployment of a software product and a software license contract for the software product," as recited in claim 1. The CIs in Best Practice model the deployment of a software product and its corresponding software license contract by, among other things, recording information about the software product and its installation status, and information about its corresponding software license contract.

(b) "storing a first model of the modeled deployment of the software product in a configuration management database (CMDB) by storing information related to the software product as a first configuration item in the CMDB and by storing information related to the software license contract as a second configuration item in the CMDB" (Claim 1[b])

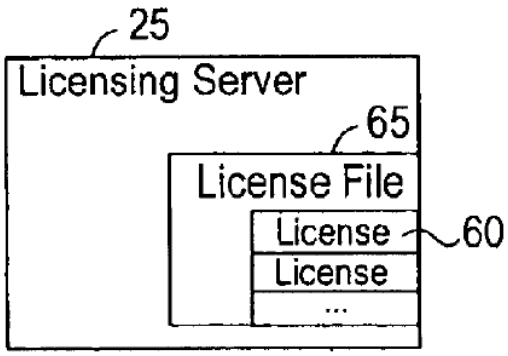
73. This limitation is disclosed by Best Practice for the same reasons as claim 1[a] above. The claimed "**first model**" takes the form of: (1) a first configuration item (CI) for a software product (such as Table D.2 in Best Practice); and (2) a second configuration item (CI) for the software product's license contract (such as Table D.1), both of which are stored in a CMDB.

74. As explained previously, Appendix D of Best Practice provides a CMDB schema with CIs for software products (Table D.2) and software license contracts (Table D.1). These CIs disclose the "**first configuration item**" and "**second configuration item**," respectively, in claim 1[b]. (Best Practice, Ex. 1004, at 119-22 (Table D.1, "Software licence inventory"), *id.* at 123 (Table D.2, "Installed software inventory.").) The "**first model**" recited in claim 1[b] comprises both of these CIs. Best Practice therefore discloses claim 1[b].

(c) "storing a second model of the modeled software license contract for the software product in a license database by generating a license certificate corresponding to the software license contract and storing the license certificate in the license database" (Claim 1[c])

75. For the claimed "**second model**," I have relied primarily upon the disclosures of Meyer. The "**second model**" in Meyer includes a license record (the "**license certificate**") stored in a license database. As I will explain below, the "license certificate" in Meyer is generated by analyzing the terms of an underlying software license contract.

76. Meyer discloses a licensing server **25** that stores a license file **65** containing licenses **60**, as shown in the excerpt of Figure 1 shown on the right. As explained in **Part V.C.1**, the term "**license certificate**" means information relating to the right to deploy software. The "**license certificate**" in Meyer corresponds to a license record stored in the license file.



77. An example of such a license certificate is shown in Figure 3, which shows a license record containing four key-value pairs (**205**, **210**, **215**, **220**, **225**, **230**, **235**, **240**) for a software license for a "Spell Checker" feature. The license includes attributes including License Count **205**, which indicates "that five licenses

may be checked out." (Meyer, 2:10-11.) The license also includes an attribute **215** for the Feature Name corresponding to the software,[1] an attribute **225** for the End Date, and an attribute **235** indicating License Version of the software feature, each with corresponding values. (Meyer, 2:11-13, 2:17-18, 2:20-21.)

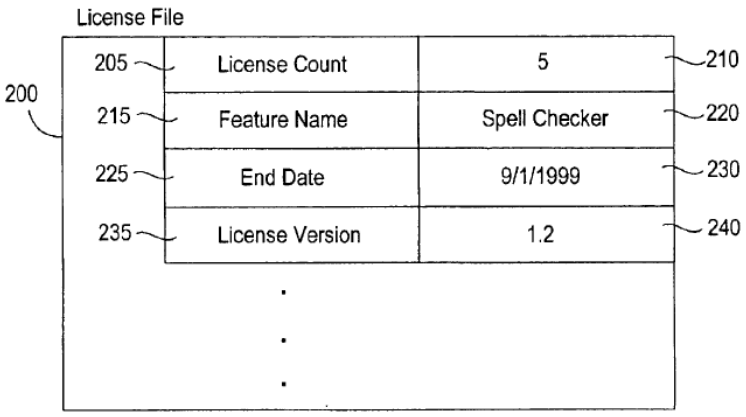


Fig. 3 - Prior Art

78. The licensing record in Meyer qualifies as a "license certificate" because it indicates the right to deploy the software. As shown above, the record contains information about the software and the number of licenses that can be checked out, the end date for the license, and other information that indicates whether the software may be deployed consistent with the license contract. As I will explain in more detail below, the software licensing system in Meyer evaluates the attributes in the license record when it receives a "license request" to request use of the software. (Meyer, 2:13-23.) Among other things, the system responds to this request by comparing the attributes in the licensing record against the

[1] Meyer explains that a typical software licensing system "maintains licenses for multiple features, or functional subsets, of an application." (Meyer, 1:61-62.)

information in the license request. (*Id.*) Because the attributes in the license record are used to indicate the right to deploy the software, the license record in Meyer qualifies as a "**license certificate**."

79. Claim 1 also requires the step of "generating a license certificate corresponding to the software license contract and storing the license certificate in the license database." Meyer discloses an internal license database **765** that stores the licensing records, which correspond to software license contracts. In fact, the license server **760** generates these license records by examining external files that contain information about the underlying software license contracts:

> Internal database **765** can be implemented on a standard storage device or memory device coupled to the license server **760**, as is well-known to those skilled in the art, and can be organized as a license file. For example, a hard drive can store license data. Preferably, as described below, the license server creates the internal license database **765** after receiving standard licenses and/or package certificates from an external file or other input source, where each package certificate includes a package description and package licensing data. For the purposes of this disclosure, the term "license database" refers to the internal license database **765**.
>
> The internal license database **765** stores entries for each license received from the external file. If a package certificate is received from the external file, the license server examines the package description and determines how many license records are written into

> the internal license database in an initialization procedure, where each license stored in the internal license database is stored as a license record. The license server examines these license records to determine whether a requesting computer system should receive a license for a designated software product, as described below.

(Meyer, 7:40-61 (underlining added).) As this passage above confirms, the license server can generate a license record ("license certificate") by analyzing an external file containing (among other things) information about an underlying software license contract. Meyer therefore discloses the step of storing a second model of a software license contract in a license database by generating a license certificate corresponding to the software license contract and storing the license certificate in the license database.

80. **Rationale for Combining Meyer and Best Practice**: I note that this limitation requires that the second model be for "**the** software product," and that the generated license certificate correspond to "**the** software license contract," the article "the" referring back to the "first model" in the claim 1[b]. The "first model" in claim 1[b] recited storage of configuration items (CIs) in a configuration management database (CMDB). Because I have relied upon Best Practice for the "first model," and Meyer for the "second model," the two references may not necessarily refer to the same "software product" and "software license contract."

81. Nevertheless, it would have been obvious to one of ordinary skill in

the art to combine the disclosures of Meyer with Best Practice, with no change in their respective functions. This would have predictably resulted in the software license system of Meyer with the ability to maintain a CMDB with CIs for storing information relating to "the software product" and "the software license contract," respectively. This combination therefore discloses that the claimed "license certificate" is generated based on the license information in the CMDB.

82. A person of ordinary skill in the art would have regarded this combination as straightforward for a number of reasons. As explained above, a license record (the "license certificate") in Meyer is generated based on information about the license — and this information may come from any source. In fact, Meyer states that the license information used to generate the license certificates can come from an "**external file**" or "**other input source**." (Meyer, 7:44-49 ("Preferably, as described below, the license server creates the internal license database **765** after receiving standard licenses and/or package certificates from an external file or other input source, where each package certificate includes a package description and package licensing data.") (underlining added), 7:51-52 ("The internal license database **765** stores entries for each license received from the external file.").) Meyer therefore places no limits or restrictions on how license information can be provided to the licensing server, and expressly contemplates that this information comes from an external source. It would therefore have been

obvious to one of ordinary skill in the art that the "external file" or "other input source" in Meyer could have included license information from the CMDB described in Best Practice.

83. As a further motivation to combine, a person of ordinary skill in the art would have understood the benefits of storing software license contract information in a CMDB. In fact, Best Practice provides an express motivation to combine by stating that "ITIL recommends the use of a DSL [definitive software library] and a CMDB for the management of software assets." (Best Practice, at p. 81 (emphasis added).)

84. The Background section of the '093 patent, in fact, specifically cites a version of Best Practice for the proposition that using CMDBs to manage software contracts was known to persons of ordinary skill in the art. ('093, Ex. 1001, 1:50-56, 1:56-57 & 2:4-5 ("ITIL indicates that the following processes make up the holistic approach to software asset management: [. . .] Relationship management processes—Software contract management.").) Storing software asset information in CMDBs and CIs, as I explained in **Part IV.B** above, is an industry standard technique for keeping track of IT assets in an enterprise.

85. One of ordinary skill in the art by March 2009 would have found nothing remarkable or inventive about using a CMDB to store license contract information for the software license system in Meyer, or using the information

from the CMDB to generate a license certificate.

86. Addy provides a further express motivation to combine Meyer with Best Practice. Addy also answers a reasonable question that some may ask when reading claim 1 – why would an enterprise want to maintain a CMDB containing software contract information, and a separate license database also containing license information?

87. Addy provides a straightforward answer to this question by explaining that one of the key purposes of a CMDB is to provide a centralized repository for IT asset information that can be integrated with other more specialized systems used by the enterprise:

> There are very few green field sites in the world that have no incumbent ITSM solution or IT related tools in place. There will inevitably be multiple silos of information around the organisation, holding similar data about the same configuration items. This is why CMDB vendors talk of federated data models where legacy systems retain ownership and control of their data but are leveraged by the central master CMDB to give a holistic view of the world.

(Addy, Ex. 1005, p. 237, § 19.8.4.)

88. One of ordinary skill in the art would have understood that the system in Meyer would have benefitted from a CMDB because, as confirmed by Addy, the CMDB would have allowed the software license system to be integrated into a

larger IT management system that provides "a holistic view" of the enterprise. (*Id.*) The software license system would thus experience greater interoperability with other IT configuration management tools, thus facilitating more effective and centralized IT management. As Addy further explains:

> Every single process within the ITSM arena has the potential to touch the CMDB to access information or to use it to store information related to specific instances of the process inputs and outputs. In short, the CMDB should be an embedded part of every IT process and should be used by every member of the IT function on a daily, if not hourly basis.

(Addy, Ex. 1005, p. 239, § 19.8.8.)

89. A person of ordinary skill in the art would therefore have found it obvious to add a CMDB to the license management system of Meyer. And although the software management system in Meyer relies upon its own local license database, one of ordinary skill in the art would have understood the benefits of also storing license information in a standardized CMDB as recognized by Addy. A skilled artisan would therefore have been amply motivated to combine Meyer and Best Practice in the manner I described above.

(d) "evaluating the deployment of the software product for compliance with the software license contract, comprising . . ." (Claim 1[d])

90. Meyer and Best Practice also disclose the evaluating step of claim

1[d]. In particular, Meyer discloses a technique for using the attributes in a licensing record (the "**license certificate**") to determine if the software can be used. (Meyer, 2:15-23.) I provided a detailed description of such a licensing record in my discussion of Figure 3 for claim 1[c] above.

91. Meyer discloses many ways in which the deployment of the software product for compliance with the software license contract is evaluated. For example, if the license record includes an "End Date" for the license of September 1, 1999, "[a] license request can be met by this license only when the license request date is less than the end date, Sep. 1, 1999." (Meyer, 2:18-20.) Meyer also discloses a technique to ensure that no more than a maximum number (e.g. five) of copies of the software are "checked out" pursuant to the license. (Meyer, 2:8-11.) More details about these checks are discussed in the further limitations of claim 1[d] below.

92. Meyer explains that these attributes are used to "evaluat[e] the deployment of the software product for compliance with the software license contract." In particular, an application program can send a "checkout" request when use of the software is requested. (Meyer, 1:46-49.) "The licensing server **25** and the application **15**, **20** [Fig. 1] cooperate to authenticate the license and to verify that the license is intended to allow the operation of the application in the current configuration, environment, and at the current time." (Meyer, 1:49-53.) If

there has been non-compliance with the software license contract, Meyer notifies the user: "When a license is violated by a client computer system, the license server **760** preferably returns a status message to the requesting computer system that indicates that the computer system would be violating the license policy when using the designated program." (Meyer, 7:62-66 (underlining added).)

93. Meyer and Best Practice therefore disclose the step of "evaluating the deployment of the software product for compliance with the software license contract," as recited in claim 1**[d]**. Claim 1**[d]** also includes sub-steps, which will be discussed below.

> **(i) "connecting and comparing the first model and the second model by comparing the first configuration item with the license certificate and connecting the license certificate with the second configuration item responsive to comparing the first configuration item with the license certificate; and" (Claim 1[d][1])**

94. The comparing and connecting step here requires the following two ordered steps: (1) comparing "**the first configuration item**" with "**the license certificate**"; and then in response, (2) connecting "**the license certificate**" with "**the second configuration item**." I will address the "comparing" and "connecting" steps separately below.

95. **Comparing Step**: Meyer and Best Practice disclose and suggest the claimed comparison between "the first configuration item" and "the license

certificate." Claim 1[b] states that the "first configuration item" stores "information related to the software product." (Meyer, 13:49-50.) This "information related to the software product" in Meyer is provided in the license request issued to the licensing server.

96. More specifically, as noted above, an application in Meyer can ask to "checkout" a particular piece of software by issuing a request to the license server, the request containing a number of "key/value" pairs. (Meyer, 1:46-49, 1:63-66 ("Attributes are key/value data pairs that are included in the licensing request and may originate in the environment of the applications, or be explicitly set by the application.").) The licensing server receives that request and compares "information related to the software product" (the key/value attributes in the licensing request) with the license record attributes ("**license certificate**"):

> FIG. 3 is a block diagram that illustrates examples of (attribute key, attribute value) pairs found in a typical license file. License **200** includes four attributes. The first attribute key **205** is the license count and its associated attribute value **210** is the number five, indicating that five licenses may be checked out. The second attribute key **215** is the feature name, and the associated attribute value **220** is "Spell Checker". A license request can be met by this license only when the feature name in the license request exactly matches the feature name in the license. In the present example, the license request must include a feature name of "Spell Checker". The third attribute indicates an end

> date (225) of Sep. 1, 1999 (230). A license request can be met by this license only when the license request date is less than the end date, Sep. 1, 1999. The fourth attribute indicates a version number (235) of 1.2 (240). A license request can be met by this license only if the requested version number is less than or equal to 1.2.

(Meyer, 2:6-23 (underlining added).)

97. Meyer therefore discloses the step of "comparing" information related to the software product to the license certificate. Meyer discloses at least two other examples of the claimed comparing step, each involving additional attributes for share modes and checkout data. (Meyer, 2:37-54 (discussing Figure 4), 2:63-3:6 (discussing Figure 5).)

98. Meyer therefore discloses comparing information relating to the software product with the license certificate, as recited in the claim.

99. **Combining Meyer and Best Practice for "Comparing" Step**: I note that this claim limitation requires a comparison between the license certificate and "**the** first configuration item," referring back to the first configuration item in the CMDB recited in claim 1**[b]**. That first configuration item, in turn, stores "information related to the software product."

100. As noted previously, Meyer does not disclose a configuration item (CI) or a CMDB. The "information related to the software product" that is compared, in Meyer, originates from a licensing request. But the claimed "first

configuration item" is disclosed in Best Practice, as explained above.

101. As I explained fully in connection with claim 1[c] above, it would have been obvious to one of ordinary skill in the art to add a CMDB to the software licensing system of Meyer. As applied to the "comparing" step, this combination would have predictably resulted in the software licensing system of Meyer the attributes in the license record (the "**license certificate**") are compared against information about the software from the "**first configuration item**" in the CMDB, rather than the licensing request in Meyer. As I explained for claim 1[c], a person of ordinary skill in the art would have found Meyer and Best Practice to be combinable and would have ample motivation for doing so.

102. With respect to the comparing step, one of ordinary skill in the art would have had additional reasons to make this combination. Best Practice explains that a centralized IT database (such as a CMDB) allows an organization to save money by centrally keeping track of how software is used throughout the enterprise, allowing it to purchase a number of licenses commensurate with its actual use of the software. (Best Practice, Ex. 1004, at 26-27.) The proposed combination would further this purpose by allowing coordination between the centrally-managed CMDB and the license server. As Addy explains, this type of coordination is one of the key purposes of a CMDB:

Every single process within the ITSM arena has the potential to touch

> the CMDB to access information or to use it to store information related to specific instances of the process inputs and outputs. In short, the CMDB should be an embedded part of every IT process and should be used by every member of the IT function on a daily, if not hourly basis.

(Addy, Ex. 1005, p. 239, § 19.8.8 (emphasis added).)

103. It would therefore have been obvious to a person of ordinary skill in the art for the license server in Meyer to communicate with a central CMDB to obtain information about the software products in order to perform the comparison. A person of ordinary skill in the art would therefore have recognized the benefit of enhancing the software licensing system of Meyer by storing information about the requested software in a first configuration item in a centralized CMDB.

104. One of ordinary skill in the art would have perceived no obstacle, technical or otherwise, in making this combination. In fact, Meyer specifically discloses that the key/value pairs in the license request could be provided by the "environment of the applications," rather than being explicitly set by the application itself. (Meyer, 1:63-66.) One of ordinary skill in the art would have appreciated that the comparison process in Meyer could perform its function regardless of whether it obtained information about the software product from a license request, a CI in a CMDB, or some other source. For example, this comparison could have been performed as part of the process for determining

whether installation of the software product in the first instance is consistent with its corresponding software license contract.

105. **Connecting Step**: Claim 1[d][1] next requires, responsive to the comparing step above, "connecting the license certificate with the second configuration item." The "second configuration item" refers back to claim 1[b], which specifies that the second configuration item contains "**information related to the software license contract**." ('093, 13:51-53 (claim 1).)

106. Meyer explains that the software licensing system, in response to the comparison, determines which license or licenses are required in order to satisfy the license request. (Meyer, 2:21-23, 2:53-55, 3:4-6.) However, because Meyer does not disclose a CMDB or CIs, it does not disclose the "second configuration item" or the step of connecting the license certificate with that configuration item.

107. But the connecting step would have been obvious over Meyer in view of Best Practice, for many of the same reasons as the "comparing" step described above. One of the attributes of the license record in Meyer (the "license certificate") is a "License Count" **205**, which indicates the number of licenses that may be checked out. (Meyer, 2:8-11; *see also* Meyer, 1:27-29 ("An SLS typically counts the number of authorized licenses in use, and imposes a restriction on the number, or count, of licenses that may be in use contemporaneously.").) Best Practice similarly discloses a software license configuration item (CI) that includes

an attribute for "**Licence status and counts**," which includes the number of current effective licenses that are used or unused. (Best Practice, Ex. 1004, at 119 (under "Licence status and counts").)

108. It would have been obvious to one of ordinary skill in the art to connect the license certificate with the software license information stored in the CMDB of Best Practice (the "second configuration item") by updating information in the CMDB in response to the results of the comparing step. For example, suppose the license server in Meyer determines (in the "comparing" step) that a use of the software is consistent with the license and accordingly modifies the "License Count" attribute **205** of the license record (the "license certificate") to reflect the newly-allowed checkout. It would have been obvious to one of ordinary skill in the art, at that time, to also update the software license configuration item (CI) in Best Practice (the "second configuration item") to reflect the change in the license certificate. For example, the "**Licence status and counts**" information in the software license CI in Best Practice could be updated to maintain consistency with the "License Count" **205** attribute in the license record of Meyer. In either case, regardless of the CI field updated, the system of Meyer would have had the additional ability to join or link ("connect") the license certificate with the second CI by updating the CI in response to the comparing step.

109. This step of updating the CI in the CMDB of Best Practice would

have been obvious to a person of ordinary skill in the art. One of ordinary skill in the art would have understood that, in any database system responsible for information about the state of any system or process (such as a CMDB), it is critically important to update the database whenever the state information changes.

110. Best Practice confirms this understanding and provides a strong and powerful motivation to combine by asserting, in strong terms, that a configuration database must be kept up-to-date to reflect the current state of the IT assets in the enterprise. (Best Practice, Ex. 1004, at 5 ("The basis of any good SAM system is accurate and up-to-date SAM information, together with the processes for control of its accuracy."), *id.* ("It is impossible to implement an effective SAM process without the successful design, development, implementation and maintenance of accurate SAM databases, automatically updated from the live infrastructure."); *id.* 101 (definition of "**SAM database**": "A database set containing all of the necessary information to support the effective operation of all SAM processes and the management of all software assets. It could form part of an overall CMDB.") (underlining added).)

111. It would therefore have been obvious to update the license count attribute of the CI in Best Practice based on the license record in Meyer, in response to the comparison in Meyer.

112. This would have predictably resulted in the claimed "connecting" of

the license record in Meyer ("license certificate") with the software license inventory CI in Best Practice ("second configuration item"). As noted above, Addy confirms that this type of coordination between CMDBs and other databases in an enterprise (such as the license database in Meyer) was essential. (Addy, Ex. 1005, p. 239, § 19.8.8.) Accordingly, Meyer, Best Practice, and Addy disclose or suggest each aspect of claim 1[d][1].

(ii) "generating an exception indication if the act of comparing the first model and the second model indicates non-compliance with the software license contract" (Claim 1[d][2])

113. Meyer and Best Practice disclose the final step of generating an exception indication. As noted above, the "comparing" step in claim 1[d][1] evaluates compliance with the software license contract. As explained in Meyer: "Based on the result of these checks, the licensing server either grants or denies (**50**, **55**) a license request. If the license request is denied, the application **15**, **20** may not use the feature associated with the license request." (Meyer, 1:56-59.) Meyer further explains that if the license is violated, a variety of exception indications can be generated including warning the user or outright blocking use of the software:

> When a license is violated by a client computer system, the license server **760** preferably returns a status message to the requesting computer system that indicates that the computer system would be

> violating the license policy when using the designated program. The client computer system **705**, **710**, **715**, **720**, **725** or program then decides the action to take if a violation has occurred. In some embodiments, the server **760** can decide this action. The action taken depends upon the level of enforcement desired by the provider of the license policy. Different degrees of enforcement to the use of the computer program on the client computer system can be provided depending on the needs of the policy provider. For example, if a lenient enforcement technique is implemented, and if a license request for a designated program is denied, then the violating computer system can display a simple warning to the user while still allowing the designated program to be run on the computer system. If strict enforcement is provided, and a license request is denied, the computer system **700** may immediately cause the designated program to quit on the computer system and not allow the program to be activated and used as long as the license policy is violated. Alternatively, the license server can decide the action to take if a license violation has occurred and can transmit the decided action to the client computer system, which can implement the action.

(Meyer, 7:62-8:19 (underlining added).)

114. Any one of these possible outcomes separately satisfies the step of "generating an exception indication if the act of comparing the first model and the second model indicates non-compliance with the software license contract." Accordingly, in my opinion, all limitations of claim 1 are disclosed or suggested by

Meyer in view of Best Practice and Addy.

C. All Limitations of Claim 5 Are Disclosed or Suggested by Meyer in View of Best Practice and Addy

115. Claim 5 depends from claim 1 and recites that "the act of evaluating is performed on demand." Meyer and Best Practice disclose that the evaluating step is performed on demand. As explained in the discussion of claim 1[d] above, the license server in Meyer receives a license request for a "checkout" issued by an application program. (Meyer, 1:46-59.) The license server performs the evaluation in response to the receipt of this checkout request. (Meyer, 2:6-23; *see also* analysis of claim 1[d] above.) The act of "evaluating" in claim 1[d] is therefore performed "on demand," as recited in claim 5.

D. All Limitations of Claim 10 Are Disclosed or Suggested by Meyer in View of Best Practice and Addy

116. Claim 10 depends from claim 1 and recites that "the act of evaluating compliance further comprises: indicating a suggested action for achieving compliance if the act of comparing the first model and the second model indicates non-compliance with the software license contract."

117. This additional limitation is also disclosed by the prior art. In particular, Meyer discloses a number of actions that can be taken to achieve compliance with a software license:

When a license is violated by a client computer system, the license

> server **760** preferably returns a status message to the requesting computer system that indicates that the computer system would be violating the license policy when using the designated program. . . The action taken depends upon the level of enforcement desired by the provider of the license policy . . . If strict enforcement is provided, and a license request is denied, the computer system **700** may immediately cause the designated program to quit on the computer system and not allow the program to be activated and used as long as the license policy is violated.

(Meyer, 7:62-66, 8:2-4, 8:11-16 (underlining added).)

118. Best Practice further discloses that if a violation of the software license contract exists, the system can signal a suggestion to perform corrective action. In particular, Best Practice discloses use of metering tools that keep track of how many copies of licensed software are concurrently in use. "[W]hen the maximum is exceeded," according to Best Practice, "an exception condition can be generated to initiate corrective action or the purchasing of additional licenses." (Best Practice, Ex.1004, at p. 66, § 7.3 (underlining added).)

119. It would have been obvious to a person of ordinary skill in the art to adapt the software license system of Meyer with the teachings of Best Practice, predictably resulting in the system of Meyer with the ability to indicate a suggested action for achieving compliance – such as an exception informing the user to purchase additional licenses – if the act of comparing the first model and the

second model indicates non-compliance with the software license contract. All limitations of claim 10 are therefore disclosed or suggested by the prior art.

E. Claims 11-13 (the "License Type" dependent claims)

120. Claims 11-13 depend from claim 1 and recite the additional features of receiving a selection of a "license type." All limitations of these claims are disclosed by Meyer, Best Practice and Addy, in further view of Bruchlos.

1. Claim 11

121. Claim 11 depends from claim 1 and recites the step of "generating the second model, comprising: receiving a selection of a license type corresponding to the software license contract; and receiving license contract data corresponding to the selected license type." I note that the claim merely specifies the "receiving" of a selection and does not specify where the selection must come from. The claim does not, for example, require a user selection.

122. As explained previously, Meyer discloses the step of creating the license record ("license certificate") by reading data about the license from an external file or other source. (Meyer, 7:40-61.) "The internal license database **765** stores entries for each license received from the external file." (Meyer, 7:51-52 (underlining added).) Meyer does not appear to disclose, however, multiple different types of licenses and thus does not appear to disclose receipt of a selection of a license type as part of this process.

123. Bruchlos discloses the ability to receive a selection of a license type corresponding to the software license contract. Bruchlos discloses a technique for creating and configuring software licenses in the context of Web services. (Bruchlos, Ex. 1006, ¶ 0001.) Bruchlos explains that a preferred way to provide a license adapted to the needs of a customer "is to offer a plurality of different license types, which is able to cover most of the clients [sic] needs, as experience shows." (Bruchlos, ¶ 0041 (underlining added).)

124. Bruchlos identifies six different license types including a "consumptive" license in which only a specific number of requests are allowed, "concurrent" license type in which only a specific number of simultaneous requests are allowed, among others. (Bruchlos, ¶¶ 0042, 0043.) "Preferably, a plurality of different license types are provided for selection to be used, which may further be combined also, in order to match best the needs of a customer." (Bruchlos, Abstract (underlining added); *see also id*., claim 2 ("The method according to claim 1, wherein said license condition data comprises at least one license type selected from a group of predetermined license types.") (underlining added).)

125. It would have been obvious to one of ordinary skill in the art to combine the teachings of Meyer, Best Practice and Addy, and Bruchlos, with no change in their respective functions, predictably resulting in the software licensing system receiving a selection of a license type corresponding to the software license

contract, and receiving license contract data corresponding to the selected license type. This selection could result, for example, in the selection of a different "external file" that is used to populate the records in the license database.

126. Bruchlos is an analogous reference in the same field as Meyer and Best Practice of software license management. Although the method in Bruchlos addresses the licensing of Web services rather than traditional software assets, Bruchlos acknowledges that there is no meaningful difference between these two fields. (Bruchlos, ¶ 0032 ("There is a remarkable similarity between e-business on demand and prior art software usage: in both fields a client is ready to pay some money in order to use a piece of software the execution of which serves him to get some predefined business value.").)

127. It was also well-known to persons of ordinary skill in the art that there are many different types of software license contracts. Best Practice, for example, provides four pages and two entire sections to describing the many different types of licenses that may exist, including software license contracts based on broad categories including duration of use (e.g. perpetual vs temporary), measure of usage (e.g. per copy, concurrent usage, enterprise-wide), and numerous other categories. (Best Practice, Ex. 1004, at pp. 106-09, §§ B.2, B.3.) This license type information is also reflected in the CMDB.

128. For example, one of the fields of the CI in Best Practice, called

"Where used," describes "Link(s) to where the license is used. This depends on the type of licensing, but could be: ▪ specific PC [. . .] ▪ named site." (*Id.* at p. 120 (underlining added).) The ability to select a license type, and receive license contract data corresponding to that selection, would have been obvious to a person of ordinary skill in the art. All limitations of claim 11 are therefore disclosed or suggested by Meyer, Best Practice and Bruchlos.

2. Claim 12

129. Claim 12 depends from claim 11 and further requires that "the act of generating the second model further comprises: providing a plurality of predetermined license types." As explained above for claim 11, the prior art discloses many different types of predetermined license types, including "Perpetual," "Temporary," "Per Copy," etc. (Best Practice, Ex. 1004, at pp. 106-09, §§ B.2, B.3.) Bruchlos, for example, discloses several different license types. (Bruchlos, ¶ 0041 ("A preferred way to do that is to offer a plurality of different license types, which is able to cover most of the clients [sic] needs, as experience shows.").) All limitations of claim 12 are therefore disclosed or suggested by Meyer, Best Practice and Bruchlos for the same reasons as claim 11.

3. Claim 13

130. Claim 13 also depends from claim 11 and further requires that "the act of generating the second model further comprises: allowing a user to define a

custom license type." It would have been obvious to one of ordinary skill based on the disclosures of Meyer, Best Practice, Addy and Bruchlos to allow the user to define a custom license type. As explained in connection with claim 11 above, Best Practice and Bruchlos confirm that multiple different license types were known in the art to tailor a license to particular customer needs.

131. One of ordinary skill in the art would have appreciated that, from time to time, it may be useful to define a custom license type. For example, a particular license may not cleanly fit within a predetermined license type, necessitating a custom license type. For example, Bruchlos explains that a custom license type can be defined by combining aspects of preexisting (atomic) license types. "Further advantageously said atomic license types can be combined in order to be able to offer a large number of different license models to the customer . . ." (Bruchlos, ¶ 0048; *id.* ("A combined license type can be obtained by associating an individual, predetermined fraction, for instance percentage of each atomic license type to the combined one, which preferably also may be used for billing purposes."); *id.* ¶ 0144 ("The following examples show, how atomic license policies can be combined according to a preferred aspect of the present invention to form a new license.") (underlining added).)

132. Bruchlos explains that combining aspects of existing licenses provides the advantage of allowing the user to tailor the license to the customer's specific

needs. (*Id.*, Abstract ("Preferably, a plurality of different license types are provided for selection to be used, which may further be combined also, in order to match best the needs of a customer.") (underlining added).) It would have been obvious to a person of ordinary skill in the art to adapt the system of Meyer to allow a user to define a custom license type.

F. All Limitations of Claim 16 Are Disclosed or Suggested by Meyer, Best Practice, and Addy

133. Claim 16 is an independent system claim that appears to simply incorporate the steps of claim 1:

> 16. A system, comprising:
> a server computer, comprising:
> a processor;
> a configuration database, coupled to the processor;
> a license database, coupled to the processor; and
> a program store, coupled to the processor, on which is stored instructions for the processor, wherein the instructions cause the processor to perform the method of claim 1.

('093, Ex. 1001, 16:1-9 (claim 16).)

134. In my opinion, claim 16 adds nothing of significance over claim 1. As explained for claim 1 above, Meyer discloses a license server, which contains or is connected to components that satisfy the "**server computer**," "**processor**"

and "**license database coupled to the processor**" limitations in claim 16. (*See* Meyer, Ex. 1003, 7:2-5 ("License server **760** may typically include hardware components for implementing license management processes, such as a microprocessor(s) or central processing unit (CPU) and associated components . . . ."), 7:9-11 ("License server **760** either includes or has access to a database **765** implemented on a storage medium such as memory, disk space, or the like."), 7:40-43, 7:49-50 ("Internal database **765** can be implemented on a standard storage device or memory device coupled to the license server **760**, as is well-known to those skilled in the art, and can be organized as a license file. . . For the purposes of this disclosure, the term 'license database' refers to the internal license database **765**.") (underlining added to all).)

135. One of ordinary skill in the art would also have understood that the license server **760** contains a "**program store**" with "**stored instructions**" that cause the processor to carry out the methods described in Meyer. (Meyer, 4:10-21.)

136. It would have been obvious to one of ordinary skill in the art, moreover, that the techniques described in Meyer would be performed by software (program instructions) executing on the license server. With respect to the claimed "**configuration database**," as explained for claim 1 above, Best Practice discloses this limitation in the form of a CMDB. It would have been obvious to one of

ordinary skill in the art that a CMDB could have been part of, or coupled, to the server computer and coupled to the processor.

137. A person of ordinary skill in the art would have found all limitations of claim 16 disclosed by Meyer and Best Practice for the same reasons as claim 1. Simply placing the method on a generic server computer with generic processor and database components adds nothing of inventiveness to the method claim 1.

138. Finally, my Declaration has applied the teachings of Meyer, Best Practice and Bruchlos to claims 1, 5, 10-13 and 16 of the ’093 patent as explained above. In my opinion, these references provide sufficient detail to enable a person of ordinary skill in the art to practice the limitations of the claims to which they apply without undue experimentation.

139. As explained in **Part IV** above, software license compliance and the use of CMDBs were not an invention of the ’093 patent, and their technological underpinnings were firmly in place by March 2009.

140. The subject matter of these claims could have been implemented by one of ordinary skill in the art based on the description provided in Meyer and Best Practice (for claims 1, 5, 10, and 16), and in Meyer, Best Practice and Bruchlos (for claims 11-13).

VII. CONCLUSION

141. In my opinion, each element of claims 1, 5, 10-13, and 16 of the '093 patent is disclosed or suggested by the prior art references described above. I hereby declare under penalty of perjury that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 28 U.S.C. § 1001.

Dated: July 3, 2015

Respectfully submitted,

Tal Lavian, Ph.D.

EXHIBIT A

Tal Lavian, Ph.D.

http://telecommnet.com
http://cs.berkeley.edu/~tlavian
tlavian@telecommnet.com

1640 Mariani Dr.
Sunnyvale, CA 94087
(408)-209-9112

Research and Consulting: Telecommunications, Network Communications, and Mobile Wireless technologies

- Scientist, educator, and technologist with over 25 years of experience
- Co-author on over 25 scientific publications, journal articles, and peer-reviewed papers
- Named inventor on over 80 issued and filed patents
- Industry fellow and lecturer at UC Berkeley Engineering – Center for Entrepreneurship and Technology (CET)

EDUCATION

- **Ph.D.**, Computer Science specializing in networking and communications, UC Berkeley
- **M.Sc.**, Electrical Engineering, Tel Aviv University
- **B.Sc.**, Mathematics and Computer Science, Tel Aviv University

EXPERTISE

Network communications, telecommunications, Internet protocols and mobile wireless:

- **Communication networks:** Internet Protocols; TCP/IP suite; TCP; UDP; IP; VoIP; Ethernet; network protocols; network software applications; Data Link, Network, and Transport Layers (L2, L3, L4)
- **Internet Software:** Internet software applications; distributed computing; cloud computing; Web applications; FTP; HTTP; Java; C; C++; client server; file transfer; multicast; streaming media
- **Routing/switching:** LAN; WAN; VPN; routing protocols; RIP; BGP; MPLS;OSPF; IS-IS;DNS; QoS; switching; packet switching; network infrastructure; network communication architectures
- **Mobile Wireless:** Wireless LAN; 802.11; cellular systems; mobile devices; smartphone technologies

LITIGATION SUPPORT SERVICES

- Expert witness in numerous USPTO PTAB – Inter Partes Review (IPR) and CBM cases
- Expert witness in Federal courts and the ITC (over 30 cases)
- Expert reports, depositions, and courtroom testimonies
- Skilled articulation of technical material for both technical and non-technical audiences
- Product and technology analysis, patent portfolios, claim charts, patentability research
- Litigation support and technology education in patent disputes
- Past cases involved Cisco, Juniper, HP, Ericsson, Microsoft, Google, Samsung and Apple

ACCOMPLISHMENTS

- Selected as Principal Investigator for three US Department of Defense (DARPA) projects
- Led research project on networking computation for the US Air Force Research Lab (AFRL)
- Led and developed the first network resource scheduling service for grid computing
- Led wireless research project for an undisclosed US federal agency
- Managed and engineered the first demonstrated transatlantic dynamic allocation of 10Gbs Lambdas as a grid service
- Spearheaded the development of the first demonstrated wire-speed active network on commercial hardware
- Invented over 80 patents; over 50 prosecuted *pro se* in front of the USPTO
- Created and chaired Nortel Networks' EDN Patent Committee
- Current IEEE Senior Member

PROFESSIONAL EXPERIENCE

University of California, Berkeley, Berkeley, CA 2000-Present

Berkeley Industry Fellow, Lecturer, Visiting Scientist, Ph.D. Candidate, Nortel's Scientist Liaison

Some positions and projects were concurrent, others sequential

- Serves as an Industry Fellow and Lecturer at the Center for Entrepreneurship and Technology (CET).
- Studied network services, telecommunication systems and software, communications infrastructure, and data centers
- Developed long-term technology for the enterprise market, integrating communication and computing technologies
- Conducted research projects in data centers (RAD Labs), telecommunication infrastructure (SAHARA), and wireless systems (ICEBERG)
- Acted as scientific liaison between Nortel Research Lab and UC Berkeley, providing tangible value in advanced technologies
- Earned a Ph.D. in Computer Science with a specialization in communications and networking

Telecomm Net Consulting, Inc. (Innovations-IP) Sunnyvale, CA 2006-Present

Principal Scientist

- Consulting in the areas of network communications, telecommunications, Internet protocols, and smartphone mobile wireless devices

- Providing architecture and system consultation for software projects relating to computer networks, mobile wireless devices, Internet web technologies
- Acting as an expert witness in network communications patent infringement lawsuits

VisuMenu, Inc. – Sunnyvale, CA 2010-Present

Co- Founder and Chief Technology Officer (CTO)

- Design and develop architecture of visual IVR technologies for smartphones and wireless mobile devices in the area of network communications
- Design crawler/spider system for IVR / PBX using Asterisk, SIP and VoIP
- Deploy the system as cloud networking and cloud computing utilizing Amazon Web Services (EC2, S3, VPC, DNS, and RDS)

Ixia, Santa Clara, CA 2008-2008

Communications Consultant

- Researched and developed advanced network communications testing technologies:
  • IxNetwork/IxN2X — tests IP routing and switching devices and broadband access equipment. Provides traffic generation and emulation for the full range of protocols: routing, MPLS, layer 2/3 VPNs, Carrier Ethernet, broadband access, and data center bridging.
  • IxLoad — quickly and accurately models high-volume video, data, and voice subscribers and servers to test real-world performance of multiservice delivery and security platforms.
  • IxCatapult — emulates a broad range of wireless access and core protocols to test wireless components and systems. When combined with IxLoad, provides an end-to-end solution for testing wireless service quality.
  • IxVeriWave — employs a client-centric model to test Wi-Fi and wireless LAN networks by generating repeatable large-scale, real-world test scenarios that are virtually impossible to create by any other means.
  • Test Automation — provides simple, comprehensive lab automation to help test engineering teams create, organize, catalog, and schedule execution of tests.

Nortel Networks, Santa Clara, CA 1996 - 2007

Originally employed by Bay Networks, which was acquired by Nortel Networks

Principal Scientist, Principal Architect, Principal Engineer, Senior Software Engineer

- Held scientific and research roles at Nortel Labs, Bay Architecture Labs, and in the office of the CTO

Principal Investigator for US Department of Defense (DARPA) Projects

- Conceived, proposed, and completed three research projects: Active Networks, DWDM-RAM, and a networking computation project for Air Force Research Lab (AFRL)
- Led a wireless research project for an undisclosed US federal agency

Academic and Industrial Researcher

- Analyzed new technologies to reduce risks associated with R&D investment
- Spearheaded research collaboration with leading universities and professors at UC Berkeley, Northwestern University, University of Amsterdam, and University of Technology, Sydney
- Evaluated competitive products relative to Nortel's products and technology
- Proactively identified prospective business ideas, which led to new networking products
- Predicted technological trends through researching the technological horizon and academic sphere
- Developed software for switches, routers and network communications devices
- Developed systems and architectures for switches, routers, and network management
- Researched and developed the following projects:
  - Data-Center Communications: network and server orchestration 2006-2007
  - DRAC: SOA-facilitated L1/L2/L3 network dynamic controller 2003-2007
  - Omega: classified wireless project for undisclosed US Federal Agency 2006
  - Open Platform: project for the US Air Force Research Laboratory (AFRL) 2005
  - Network Resource Orchestration for Web Services Workflows 2004-2005
  - Proxy Study between Web/Grids Services and Network Services 2004
  - Streaming Content Replication: real-time A/V media multicast at edge 2003-2004
  - DWDM-RAM: US DARPA-funded program on agile optical transport 2003-2004
  - Packet Capturing and Forwarding Service on IP and Ethernet traffic 2002-2003
  - CO2: content-aware agile networking 2001-2003
  - Active Networks: US DARPA-funded research program 1999-2002
  - ORE: programmable network service platform 1998-2002
  - JVM Platform: Java on network devices 1998-2001
  - Web-Based Device Management: network device management 1996-1997

Technology Innovator and Patent Leader

- Created and chaired Nortel Networks' EDN Patent Committee
- Facilitated continuous stream of innovative ideas and their conversion into intellectual property rights
- Developed intellectual property assets through invention and analysis of existing technology portfolios

Aptel Communications, Netanya, Israel 1994-1995

Software Engineer, Team Leader

Start-up company focused on mobile wireless CDMA spread spectrum PCN/PCS

- Developed a mobile wireless device using an unlicensed band [Direct Sequence Spread Spectrum (DSSS)]
- Designed and managed a personal communication network (PCN) and personal communication system (PCS), the precursors of short text messages (SMS)
- Designed and developed network communications software products (mainly in C/C++)
- Brought a two-way paging product from concept to development

Scitex Ltd., Herzeliya, Israel 1990-1993

Software Engineer, Team Leader

Software and hardware company acquired by Hewlett Packard (HP)

- Developed system and network communications (mainly in C/C++)
- Invented Parallel SIMD Architecture
- Participated in the Technology Innovation group

Shalev, Ramat-HaSharon, Israel 1987-1990

Start-up company

Software Engineer

- Developed real-time software and algorithms (mainly in C/C++ and Pascal)

PROFESSIONAL ASSOCIATIONS

- IEEE Senior Member
- IEEE CNSV co-chair Intellectual Property SIG (2013)
- President Next Step Toastmasters (an advanced TM club in the Silicon Valley) (2013)
- Technical Co-Chair, IEEE Hot Interconnects 2005 at Stanford University
- Member, IEEE Communications Society (COMMSOC)
- Member, IEEE Computer Society
- Member, IEEE Systems, Man, and Cybernetics Society
- Member, IEEE-USA Intellectual Property Committee
- Member, ACM, ACM Special Interest Group on Data Communication (SIGCOM)
- Member, ACM Special Interest Group on Hypertext, Hypermedia and Web (SIGWEB)
- Member, IEEE Consultants' Network (CNSV)
- Global Member, Internet Society (ISOC)
- President Java Users Group – Silicon Valley Mountain View, CA,1999-2000
- Toastmasters International

ADVISORY BOARDS

- Quixey (present) – search engine for wireless mobile apps
- Mytopia – mobile social games
- iLeverage – Israeli Innovations

PROFESSIONAL AWARDS

- Top Talent Award – Nortel
- Top Inventors Award – Nortel EDN
- Certified IEEE-WCET - Wireless Communications Engineering Technologies
- Toastmasters International - Competent Communicator (twice)
- Toastmasters International - Advanced Communicator Bronze

Patents and Publications

(Not an exhaustive list)

Patents Issued:

- **US 8,688,796** Rating system for determining whether to accept or reject objection raised by user in social network
- **US 8,572,303** Portable universal communication device
- **US 8,553,859** Device and method for providing enhanced telephony
- **US 8,548,131** Systems and methods for communicating with an interactive voice response system
- **US 8,537,989** Device and method for providing enhanced telephony
- **US 8,341,257** Grid proxy architecture for network resources
- **US8,161,139** Method and apparatus for intelligent management of a network element
- **US 8,146,090** Time-value curves to provide dynamic QoS for time sensitive file transfer
- **US 8,078,708** Grid proxy architecture for network resources
- **US 7,944,827** Content-aware dynamic network resource allocation
- **US7,860,999** Distributed computation in network devices
- **US 7,734,748** Method and apparatus for intelligent management of a network element
- **US 7,710,871** Dynamic assignment of traffic classes to a priority queue in a packet forwarding device
- **US 7,580,349** Content-aware dynamic network resource allocation
- **US 7,433,941** Method and apparatus for accessing network information on a network device
- **US 7,359,993** Method and apparatus for interfacing external resources with a network element
- **US 7,313,608** Method and apparatus for using documents written in a markup language to access and configure network elements
- **US 7,260,621** Object-oriented network management interface

- **US 7,237,012** Method and apparatus for classifying Java remote method invocation transport traffic
- **US 7,127,526** Method and apparatus for dynamically loading and managing software services on a network device
- **US7,047,536** Method and apparatus for classifying remote procedure call transport traffic
- **US7,039,724** Programmable command-line interface API for managing operation of a network device
- **US6,976,054** Method and system for accessing low-level resources in a network device
- **US6,970,943** Routing architecture including a compute plane configured for high-speed processing of packets to provide application layer support
- **US6,950,932** Security association mediator for Java-enabled devices
- **US6,850,989** Method and apparatus for automatically configuring a network switch
- **US6,845,397** Interface method and system for accessing inner layers of a network protocol
- **US6,842,781** Download and processing of a network management application on a network device
- **US6,772,205** Executing applications on a target network device using a proxy network device
- **US6,564,325** Method of and apparatus for providing multi-level security access to system
- **US6,175,868** Method and apparatus for automatically configuring a network switch
- **US6,170,015** Network apparatus with Java co-processor
- **US 8,619,793** Dynamic assignment of traffic classes to a priority queue in a packet forwarding device
- **US 8687,777** Systems and methods for visual presentation and selection of IVR menu
- **US 8,681,951** Systems and methods for visual presentation and selection of IVR menu

- **US 8,625,756** Systems and methods for visual presentation and selection of IVR menu
- **US 8,594,280** Systems and methods for visual presentation and selection of IVR menu
- **US 8,548,135** Systems and methods for visual presentation and selection of IVR menu
- **US 8,406,388** Systems and methods for visual presentation and selection of IVR menu
- **US 8,345,835** Systems and methods for visual presentation and selection of IVR menu
- **US 8,223,931** Systems and methods for visual presentation and selection of IVR menu
- **US 8,160,215** Systems and methods for visual presentation and selection of IVR menu
- **US 8,155,280** Systems and methods for visual presentation and selection of IVR menu
- **US 8,054,952** Systems and methods for visual presentation and selection of IVR menu
- **US 8,000,454** Systems and methods for visual presentation and selection of IVR menu
- **EP 1,905,211** Technique for authenticating network users
- **EP 1,142,213** Dynamic assignment of traffic classes to a priority queue in a packet forwarding device
- **EP 1,671,460** Method and apparatus for scheduling resources on a switched underlay network
- **CA 2,358,525** Dynamic assignment of traffic classes to a priority queue in a packet forwarding device

Patent Applications Published and Pending:
(Not an exhaustive list)

- **US 20140105025** Dynamic Assignment of Traffic Classes to a Priority Queue in a Packet Forwarding Device
- **US 20140105012** Dynamic Assignment of Traffic Classes to a Priority Queue in a Packet Forwarding Device
- **US 20140012991** Grid Proxy Architecture for Network Resources
- **US 20130080898** Systems and Methods for Electronic Communications
- **US 20130022191** Systems and Methods for Visual Presentation and Selection of IVR Menu
- **US 20130022183** Systems and Methods for Visual Presentation and Selection of IVR Menu
- **US 20130022181** Systems and Methods for Visual Presentation and Selection of IVR Menu
- **US 20120180059** Time-Value Curves to Provide Dynamic QOS for Time Sensitive File Transfers
- **US 20120063574** Systems and Methods for Visual Presentation and Selection of IVR Menu
- **US 20110225330** Portable Universal Communication Device
- **US 20100220616** Optimizing Network Connections
- **US 20100217854** Method and Apparatus for Intelligent Management of a Network Element
- **US 20100146492** Translation of Programming Code
- **US 20100146112** Efficient Communication Techniques
- **US 20100146111** Efficient Communication in a Network
- **US 20090313613** Methods and Apparatus for Automatic Translation of a Computer Program Language Code
- **US 20090313004** Platform-Independent Application Development Framework
- **US 20090279562** Content-aware dynamic network resource allocation
- **US 20080040630** Time-Value Curves to Provide Dynamic QoS for Time Sensitive File Transfers
- **US 20070169171** Technique for authenticating network users
- **US 20060123481** Method and apparatus for network immunization
- **US 20060075042** Extensible Resource Messaging Between User Applications and Network Elements in a Communication Network

- **US 20050083960** Method and Apparatus for Transporting Parcels of Data Using Network Elements with Network Element Storage
- **US 20050076339** Method and Apparatus for Automated Negotiation for Resources on a Switched Underlay Network
- **US 20050076336** Method and Apparatus for Scheduling Resources on a Switched Underlay Network
- **US 20050076173** Method And Apparatus for Preconditioning Data to Be Transferred on a Switched Underlay Network
- **US 20050076099** Method and Apparatus for Live Streaming Media Replication in a Communication Network
- **US 20050074529** Method and apparatus for transporting visualization information on a switched underlay network
- **US 20040076161** Dynamic Assignment of Traffic Classes to a Priority Queue in a Packet Forwarding Device
- **US 20020021701** Dynamic Assignment of Traffic Classes to a Priority Queue in a Packet Forwarding Device
- **WO 2007/008976** Technique for Authenticating Network Users
- **WO 2006/063052** Method and apparatus for network immunization
- **WO2000/0054460** Method and apparatus for accessing network information on a network device

Publications

(Not an exhaustive list)

- "R&D Models for Advanced Development & Corporate Research" Understanding Six Models of Advanced R&D - Ikhlaq Sidhu, Tal Lavian, Victoria Howell - University of California, Berkeley. Accepted paper for 2015 ASEE Annual Conference and Exposition- June 2015
- "Communications Architecture in Support of Grid Computing",Tal Lavian,Scholar's Press 2013ISBN 978-3-639-51098-0.
- "Applications Drive Secure Lightpath Creation across Heterogeneous Domains, Feature Topic Optical Control Planes for Grid Networks: Opportunities, Challenges and the Vision." Gommans L.; Van Oudenaarde B.; Dijkstra F.; De Laat C.; Lavian T.; Monga I.; Taal A.; Travostino F.; Wan A.; IEEE *Communications Magazine*, vol. 44, no. 3, March 2006, pp. 100-106.
- *Lambda Data Grid: Communications Architecture in Support of Grid Computing*. Tal I. Lavian, Randy H. Katz; Doctoral Thesis, University of California at Berkeley. January 2006.
- "Information Switching Networks." Hoang D.B.; T. Lavian;*The 4th Workshop on the Internet, Telecommunications and Signal Processing, WITSP2005*, December 19-21, 2005, Sunshine Coast, Australia.
- "Impact of Grid Computing on Network Operators and HW Vendors." Allcock B.; Arnaud B.; Lavian T.; Papadopoulos P.B.; Hasan M.Z.; Kaplow W.; *IEEE Hot Interconnects at Stanford University 2005*, pp.89-90.
- *DWDM-RAM: AData Intensive Grid Service Architecture Enabled by Dynamic Optical Networks*. Lavian T.; Mambretti J.; Cutrell D.; Cohen H.J; Merrill S.; Durairaj R.; Daspit P.; Monga I.; Naiksatam S.; Figueira S.; Gutierrez D.; Hoang D.B., Travostino F.; CCGRID 2004, pp. 762-764.
- *DWDM-RAM: AnArchitecture for Data Intensive Service Enabled by Next Generation Dynamic Optical Networks*. Hoang D.B.; Cohen H.; Cutrell D.; Figueira S.; Lavian T.; Mambretti J.; Monga I.; Naiksatam S.; Travostino F.; Proceedings IEEE Globecom 2004, Workshop on High-Performance Global Grid Networks, Houston, 29 Nov. to 3 Dec. 2004, pp.400-409.
- *Implementation of a Quality of Service Feedback Control Loop on Programmable Routers*. Nguyen C.; Hoang D.B.; Zhao, I.L.; Lavian, T.; Proceedings, 12th IEEE International Conference on Networks 2004. (ICON 2004) Singapore, Volume 2, 16-19 Nov. 2004, pp.578-582.
- *A Platform for Large-Scale Grid Data Service on Dynamic High-Performance Networks*. Lavian T.; Hoang D.B.; Mambretti J.; Figueira S.; Naiksatam S.; Kaushil N.; Monga I.; Durairaj R.; Cutrell D.; Merrill S.; Cohen H.; Daspit P.; Travostino F; GridNets 2004, San Jose, CA., October 2004.
- *DWDM-RAM: Enabling Grid Services with Dynamic Optical Networks*. Figueira S.; Naiksatam S.; Cohen H.; Cutrell D.; Daspit, P.; Gutierrez D.; Hoang D. B.; Lavian T.; Mambretti J.; Merrill S.; Travostino F; Proceedings, 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, Chicago, USA, April 2004, pp. 707-714.
- *DWDM-RAM: Enabling Grid Services with Dynamic Optical Networks*. Figueira S.; Naiksatam S.; Cohen H.; Cutrell D.; Gutierrez D.; Hoang D.B.; Lavian T.; Mambretti J.; Merrill S.; Travostino F.; 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, Chicago, USA, April 2004.

- *An Extensible, Programmable, Commercial-Grade Platform for Internet Service Architecture*. Lavian T.; Hoang D.B.; Travostino F.; Wang P.Y.; Subramanian S.; Monga I.; IEEE Transactions on Systems, Man, and Cybernetics on Technologies Promoting Computational Intelligence, Openness and Programmability in Networks and Internet Services Volume 34, Issue 1, Feb. 2004, pp.58-68.
- *DWDM-RAM: An Architecture for Data Intensive Service Enabled by Next Generation Dynamic Optical Networks*. Lavian T.; Cutrell D.; Mambretti J.; Weinberger J.; Gutierrez D.; Naiksatam S.; Figueira S.; Hoang D. B.; Supercomputing Conference, SC2003 Igniting Innovation, Phoenix, November 2003.
- *Edge Device Multi-Unicasting for Video Streaming*. Lavian T.; Wang P.; Durairaj R.; Hoang D.; Travostino F.; Telecommunications, 2003. ICT 2003. 10th International Conference on Telecommunications, Tahiti, Volume 2, 23 Feb.-1 March, 2003 pp. 1441-1447.
- The SAHARA Model for Service Composition Across Multiple Providers. Raman B.; Agarwal S.; Chen Y.; Caesar M.; Cui W.; Lai K.; Lavian T.; Machiraju S.; Mao Z. M.; Porter G.; Roscoe T.; Subramanian L.; Suzuki T.; Zhuang S.; Joseph A. D.; Katz Y.H.; Stoica I.; Proceedings of the First International Conference on Pervasive Computing. ACM Pervasive 2002, pp. 1-14.
- *Enabling Active Flow Manipulation in Silicon-Based Network Forwarding Engines*. Lavian T.; Wang P.; Travostino F.; Subramanian S.; Duraraj R.; Hoang D.B.; Sethaput V.; Culler D.; Proceeding of the Active Networks Conference and Exposition, 2002.(DANCE) 29-30 May 2002, pp. 65-76.
- *Practical Active Network Services within Content-Aware Gateways*. Subramanian S.; Wang P.; Durairaj R.; Rasimas J.; Travostino F.; Lavian T.; Hoang D.B.; Proceeding of the DARPA Active Networks Conference and Exposition, 2002.(DANCE) 29-30 May 2002, pp. 344-354.
- *Active Networking on a Programmable Network Platform.* Wang P.Y.; Lavian T.; Duncan R.; Jaeger R.; Fourth IEEE Conference on Open Architectures and Network Programming (OPENARCH), Anchorage, April 2002.
- *Intelligent Network Services through Active Flow Manipulation*. Lavian T.; Wang P.; Travostino F.; Subramanian S.; Hoang D.B.; Sethaput V.; IEEE Intelligent Networks 2001 Workshop (IN2001), Boston, May 2001.
- Intelligent Network Services through Active Flow Manipulation. Lavian T.; Wang P.; Travostino F.; Subramanian S.; Hoang D.B.; Sethaput V.; Intelligent Network Workshop, 2001 IEEE 6-9 May 2001, pp.73 - 82.
- *Enabling Active Flow Manipulation in Silicon-based Network Forwarding Engine*. Lavian, T.; Wang, P.; Travostino, F.; Subramanian S.; Hoang D.B.; Sethaput V.; Culler D.; Journal of Communications and Networks, March 2001, pp.78-87.
- *Active Networking on a Programmable Networking Platform*. Lavian T.; Wang P.Y.; IEEE Open Architectures and Network Programming, 2001, pp. 95-103.
- *Enabling Active Networks Services on a Gigabit Routing Switch*. Wang P.; Jaeger R.; Duncan R.; Lavian T.; Travostino F.; 2nd Workshop on Active Middleware Services, 2000.

- *Dynamic Classification in Silicon-Based Forwarding Engine Environments*. Jaeger R.; Duncan R.; Travostino F.; Lavian T.; Hollingsworth J.; Selected Papers. 10th IEEE Workshop on Metropolitan Area and Local Networks, 1999. 21-24 Nov. 1999, pp.103-109.
- *Open Programmable Architecture for Java-Enabled Network Devices*. Lavian, T.; Jaeger, R. F.; Hollingsworth, J. K.; IEEE Hot Interconnects Stanford University, August 1999, pp. 265-277.
- *Open Java SNMP MIB API*. Rob Duncan, Tal Lavian, Roy Lee, Jason Zhou, Bay Architecture Lab Technical Report TR98-038, December 1998.
- *Java-Based Open Service Interface Architecture*. Lavian T.; Lau S.; BAL TR98-010 Bay Architecture Lab Technical Report, March 1998.
- *Parallel SIMD Architecture for Color Image Processing*. Lavian T. Tel – Aviv University, Tel – Aviv, Israel, November 1995.
- *Grid Network Services, Draft-ggf-ghpn-netservices-1.0*.George Clapp, Tiziana Ferrari, Doan B. Hoang, Gigi Karmous-Edwards, Tal Lavian, Mark J. Leese, Paul Mealor, Inder Monga, Volker Sander, Franco Travostino, Global Grid Forum(GGF).
- *Project DRAC: Creating an applications-aware network.*Travostino F.; Keates R.; Lavian T.; Monga I.; Schofield B.; Nortel Technical Journal, February 2005, pp. 23-26.
- *Optical Network Infrastructure for Grid, Draft-ggf-ghpn-opticalnets-1.*Dimitra Simeonidou, Reza Nejabati, Bill St. Arnaud, Micah Beck, Peter Clarke, Doan B. Hoang, David Hutchison, Gigi Karmous-Edwards, Tal Lavian, Jason Leigh, Joe Mambretti, Volker Sander, John Strand, Franco Travostino, Global Grid Forum(GGF) GHPN Standard GFD-I.036 August 2004.
- *Popeye - Using Fine-grained Network Access Control to Support Mobile Users and Protect Intranet Hosts.*Mike Chen, Barbara Hohlt, Tal Lavian, December 2000.

Presentations and Talks

(Not an exhaustive list)

- Lambda Data Grid: An Agile Optical Platform for Grid Computing and Data-intensive Applications.
- Web Services and OGSA
- WINER Workflow Integrated Network Resource Orchestration.
- Technology & Society.
- Abundant Bandwidth and how it affects us?
- Active Content Networking(ACN).
- DWDM-RAM:Enabling Grid Services with Dynamic Optical Networks .
- Application-engaged Dynamic Orchestration of Optical Network Resources .
- A Platform for Data Intensive Services Enabled by Next Generation Dynamic Optical Networks .
- Optical Networks.
- Grid Optical Network Service Architecture for Data Intensive Applications.
- Optical Networking & DWDM.
- OptiCal Inc.
- OptiCal & LUMOS Networks.
- Optical Networking Services.
- Business Models for Dynamically Provisioned Optical Networks.
- Business Model Concepts for Dynamically Provisioned Optical Networks.
- Optical Networks Infrastructure.
- Research Challenges in agile optical networks.
- Services and Applications' infrastructure for agile optical networks.
- Impact on Society.
- TeraGrid Communication and Computation.
- Unified Device Management via Java-enabled Network Devices.
- Active Network Node in Silicon-Based L3 Gigabit Routing Switch.
- Active Nets Technology Transfer through High-Performance Network Devices.
- Programmable Network Node: Applications.
- Open Innovation via Java-enabled Network Devices.
- Practical Considerations for Deploying a Java Active Networking Platform.
- Open Java-Based Intelligent Agent Architecture for Adaptive Networking Devices.
- Java SNMP Oplet.
- Open Distributed Networking Intelligence: A New Java Paradigm.
- Open Programmability.
- Active Networking On A Programmable Networking Platform.
- Open Networking through Programmability.
- Open Programmable Architecture for Java-enabled Network Devices.

- Integrating Active Networking and Commercial-Grade Routing Platforms.
- Programmable Network Devices.
- To be smart or not to be?