

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of: Ranganath G. Iyengar
U.S. Patent No.: 7,904,686 Attorney Docket No.: 39898-0050IP1
Issue Date: March 8, 2011
Appl. Serial No.: 12/152,595
Filing Date: May 13, 2008
Title: DATA SECURITY FOR USE WITH A FILE SYSTEM

DECLARATION OF DR. NENAD MEDVIDOVIC

TABLE OF CONTENTS

I.	INTRODUCTION	20
II.	QUALIFICATIONS	20
III.	LEGAL PRINCIPLES	29
	A. Person of Ordinary Skill in the Art	29
	B. Obviousness.....	30
	C. Claim Interpretation	32
IV.	MATERIALS CONSIDERED	32
V.	THE '686 PATENT	35
	A. Pertinent Disclosure	35
VI.	PERSON OF ORDINARY SKILL IN THE ART	36
	A. A POSITA's Ordinary Level of Skill.....	36
	B. A POSITA's Background Knowledge and Understanding of the Pertinent Technology	37
	1. File Systems	37
	2. Data Security	39
VII.	OVERVIEW OF CONCLUSIONS FORMED	42
VIII.	GROUND 1A: MADANY AND LOCASTO	43
	A. Madany (EX1004).....	43
	B. Locasto (EX1005)	47
	C. The Madany-Locasto Combination.....	52
	1. Reasons to Combine	58
	2. Reasonable Expectation of Success	63

D.	Claim Element Analysis.....	64
1.	Claim 1	64
	(i) Element [1.pre]: A method of providing data security for use with a file system, the method comprising:	64
	(ii) Element [1.1.1]: applying a mapping function to data block numbers that are associated with a file,.....	65
	(iii) Element [1.1.2]: wherein the data block numbers are contained in an index node associated with said file; and	68
	(iv) Element [1.2.1]: obtaining mapped data block numbers after applying the mapping function,.....	73
	(v) Element [1.2.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.	73
2.	Claim 2	76
	(i) Claim [2]: The method of claim 1, further comprising: storing a mapping of the data block numbers and the mapped data block numbers.	76
3.	Claim 5	78
	(i) Claim [5]: The method of claim 1, wherein the mapping function is an equation that is applied to each data block number of the file.....	78
4.	Claim 6	80
	(i) Claim [6]: The method of claim 1, further comprising: selecting a complexity level of the mapping function in order to set a security level and efficiency of the mapping function.	80
5.	Claim 7	82

(i)	Claim [7]: The method of claim 1, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.....	82
6.	Claim 8.....	83
(i)	Claim [8]: The method of claim 1, wherein the mapping function prevents copying of data of the file in the storage device.....	83
7.	Claim 9.....	84
(i)	Element [9.pre]: An apparatus for providing data security for use with a file system, the apparatus comprising:	84
(ii)	Element [9.1.1]: a block distribution engine configured to apply a mapping function to data block numbers that are associated with a file,.....	85
(iii)	Element [9.1.2]: wherein the data block numbers are contained in an index node associated with said file, and	86
(iv)	Element [9.1.3]: obtain mapped data block numbers after applying the mapping function;.....	86
(v)	Element [9.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.	86
8.	Claim 10.....	86
(i)	Claim [10]: The apparatus of claim 9, wherein the block distribution engine is configured to store a mapping of the data block numbers and the mapped data block numbers.	86
9.	Claim 13.....	86
(i)	Claim [13] The apparatus of claim 9, wherein the mapping function is an equation that is applied to each data block number of the file.....	86

10.	Claim 14.....	87
	(i) Claim [14] The apparatus of claim 9, wherein a complexity level of the mapping function is selected in order to set a security level and efficiency of the mapping function.	87
11.	Claim 15.....	87
	(i) Claim [15] The apparatus of claim 9, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.	87
12.	Claim 16.....	87
	(i) Claim [16] The apparatus of claim 9, wherein the mapping function prevents copying of data of the file in the storage device.	87
13.	Claim 17.....	87
	(i) Element [17.pre] An apparatus for providing data security for use with a file system, the apparatus comprising:	88
	(ii) Element [17.1.1] means for applying a mapping function to data block numbers that are associated with a file,	88
	(iii) Element [17.1.2] wherein the data block numbers are contained in an index node associated with said file; and	89
	(iv) Element [17.2.1] means for obtaining mapped data block numbers after applying the mapping function,	90
	(v) Element [17.2.2] wherein the mapped data block numbers are addresses of data of the file in a storage device	90
14.	Claim 18.....	91

(i)	Element [18.pre] Article of manufacture, comprising: Element [18.1] a machine-readable medium having stored thereon instructions to:	91
(ii)	Element [18.2.1] apply a mapping function to data block numbers that are Element associated with a file,	91
(iii)	Element [18.2.2] wherein the data block numbers are contained in an index node associated with said file; and	91
(iv)	Element [18.3.1] obtain mapped data block numbers after applying the mapping function,	91
(v)	Element [18.3.2] wherein the mapped data block numbers are addresses of data of the file in a storage device.	92
IX.	GROUND 1B: MADANY AND LOCASTO	92
A.	Madany (EX1004).....	92
B.	Locasto (EX1005)	92
C.	The Madany-Locasto Combination.....	92
1.	Reasons to Combine and Reasonable Expectation of Success.	95
D.	Claim Element Analysis.....	95
1.	Claim 1	95
(i)	Element [1.pre]: A method of providing data security for use with a file system, the method comprising:	95
(ii)	Element [1.1.1]: applying a mapping function to data block numbers that are associated with a file,	95
(iii)	Element [1.1.2]: wherein the data block numbers are contained in an index node associated with said file; and	98
(iv)	Element [1.2.1]: obtaining mapped data block numbers after applying the mapping function,.....	102

(v)	Element [1.2.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.	102
2.	Claim 2	103
(i)	Claim [2]: The method of claim 1, further comprising: storing a mapping of the data block numbers and the mapped data block numbers.	103
3.	Claim 5	106
(i)	Claim [5]: The method of claim 1, wherein the mapping function is an equation that is applied to each data block number of the file.....	106
4.	Claim 6	107
(i)	Claim [6]: The method of claim 1, further comprising: selecting a complexity level of the mapping function in order to set a security level and efficiency of the mapping function.	107
5.	Claim 7	108
(i)	Claim [7]: The method of claim 1, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.....	108
6.	Claim 8	108
(i)	Claim [8]: The method of claim 1, wherein the mapping function prevents copying of data of the file in the storage device.	108
7.	Claim 9	108
(i)	Element [9.pre]: An apparatus for providing data security for use with a file system, the apparatus comprising: ..	108

(ii)	Element [9.1.1]: a block distribution engine configured to apply a mapping function to data block numbers that are associated with a file,.....	109
(iii)	Element [9.1.2]: wherein the data block numbers are contained in an index node associated with said file, and	109
(iv)	Element [9.1.3]: obtain mapped data block numbers after applying the mapping function;.....	109
(v)	Element [9.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.	109
8.	Claim 10.....	110
(i)	Claim [10]: The apparatus of claim 9, wherein the block distribution engine is configured to store a mapping of the data block numbers and the mapped data block numbers.	110
9.	Claim 13.....	110
(i)	Claim [13] The apparatus of claim 9, wherein the mapping function is an equation that is applied to each data block number of the file.....	110
10.	Claim 14.....	110
(i)	Claim [14] The apparatus of claim 9, wherein a complexity level of the mapping function is selected in order to set a security level and efficiency of the mapping function.	110
11.	Claim 15.....	110
(i)	Claim [15] The apparatus of claim 9, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.	110
12.	Claim 16.....	111

(i)	Claim [16] The apparatus of claim 9, wherein the mapping function prevents copying of data of the file in the storage device.	111
13.	Claim 17.....	111
(i)	Element [17.pre] An apparatus for providing data security for use with a file system, the apparatus comprising: ..	111
(ii)	Element [17.1.1] means for applying a mapping function to data block numbers that are associated with a file, ..	111
(iii)	Element [17.1.2] wherein the data block numbers are contained in an index node associated with said file; and	111
(iv)	Element [17.2.1] means for obtaining mapped data block numbers after applying the mapping function,	112
(v)	Element [17.2.2] wherein the mapped data block numbers are addresses of data of the file in a storage device.	112
14.	Claim 18.....	112
(i)	Element [18.pre] Article of manufacture, comprising: Element [18.1] a machine-readable medium having stored thereon instructions to:	112
(ii)	Element [18.2.1] apply a mapping function to data block numbers that are associated with a file,	113
(iii)	Element [18.2.2] wherein the data block numbers are contained in an index node associated with said file; and	113
(iv)	Element [18.3.1] obtain mapped data block numbers after applying the mapping function,	113
(v)	Element [18.3.2] wherein the mapped data block numbers are addresses of data of the file in a storage device.	113
X.	GROUND 2: MADANY-LOCASTO-LEFFLER.....	113

A.	Leffler (EX1006).....	113
B.	Claim Element Analysis.....	118
1.	Claims 1, 9, 17-18.....	118
	(i) Element [1.1.2]: wherein the data block numbers are contained in an index node associated with said file; and	118
	(ii) Element [9.1.2]: wherein the data block numbers are contained in an index node associated with said file, and	118
	(iii) Element [17.1.2] wherein the data block numbers are contained in an index node associated with said file; and	118
	(iv) Element [18.2.2] wherein the data block numbers are contained in an index node associated with said file; and	118
2.	Claim 3.....	122
	(i) Element [3.1]: The method of claim 1, wherein a read operation for the file comprises: obtaining the data block numbers of the file;.....	122
	(ii) Element [3.2]: obtaining the mapped data block numbers of the file based on the mapping function that is applied to the data block numbers; and	122
	(iii) Element [3.3]: reading the data in the mapped data block numbers in the storage device.	122
3.	Claim 4.....	125
	(i) Element [4.1]: The method of claim 1, wherein a write operation for the file comprises: allocating data block numbers of free data blocks in the storage device;	125

(ii)	Element [4.2]: linking the data block numbers to a file name of the file;	127
(iii)	Element [4.3]: applying the mapping function to the data block numbers in order to obtain the mapped data block numbers; and.....	129
(iv)	Element [4.4]: writing the data of the file into the mapped data block numbers in the storage device.....	129
4.	Claim 11	130
(i)	Element [11.1] The apparatus of claim 9, wherein for a read operation for the file, the block distribution engine obtains the data block numbers of the file from a file system, and	130
(ii)	Element [11.2] obtains the mapped data block numbers of the file based on the mapping function that is applied to the data block numbers;	130
(iii)	Element [11.3] and wherein a block device driver is configured to read the data in the mapped data block numbers in the storage device.	130
5.	Claim 12	130
(i)	Element [12.1] The apparatus of claim 9, wherein for a write operation for the file, the block distribution engine is configured to receive allocated data block numbers of free data blocks in the storage device from a file system, ...	130
(ii)	Element [12.2] link the data block numbers to a file name of the file, and	131
(iii)	Element [12.3] apply the mapping function to the data block numbers in order to obtain the mapped data block numbers;	131
(iv)	Element [12.4] and wherein a block device driver is configured to write the data of the file into the mapped data block numbers in the storage device.	131

XI.	GROUND 3: LEFFLER AND LOCASTO	131
A.	Leffler (EX1006).....	131
B.	Locasto (EX1005)	131
C.	The Leffler-Locasto Combination.....	131
1.	Reasons to Combine	136
2.	Reasonable Expectation of Success	139
D.	Claim Element Analysis.....	140
1.	Claim 1	140
(i)	Element [1.pre]: A method of providing data security for use with a file system, the method comprising:	140
(ii)	Element [1.1.1]: applying a mapping function to data block numbers that are associated with a file,.....	141
(iii)	Element [1.1.2]: wherein the data block numbers are contained in an index node associated with said file; and	143
(iv)	Element [1.2.1]: obtaining mapped data block numbers after applying the mapping function,.....	145
(v)	Element [1.2.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.	146
2.	Claim 2	147
(i)	Claim [2]: The method of claim 1, further comprising: storing a mapping of the data block numbers and the mapped data block numbers.	147
3.	Claim 3	150
(i)	Element [3.1]: The method of claim 1, wherein a read operation for the file comprises: obtaining the data block numbers of the file;.....	150

(ii)	Element [3.2]: obtaining the mapped data block numbers of the file based on the mapping function that is applied to the data block numbers; and	150
(iii)	Element [3.3]: reading the data in the mapped data block numbers in the storage device.	150
4.	Claim 4	151
(i)	Element [4.1]: The method of claim 1, wherein a write operation for the file comprises: allocating data block numbers of free data blocks in the storage device;	151
(ii)	Element [4.2]: linking the data block numbers to a file name of the file;	153
(iii)	Element [4.3]: applying the mapping function to the data block numbers in order to obtain the mapped data block numbers; and.....	156
(iv)	Element [4.4]: writing the data of the file into the mapped data block numbers in the storage device.....	156
5.	Claim 5	157
(i)	Claim [5]: The method of claim 1, wherein the mapping function is an equation that is applied to each data block number of the file.....	157
6.	Claim 6	158
(i)	Claim [6]: The method of claim 1, further comprising: selecting a complexity level of the mapping function in order to set a security level and efficiency of the mapping function.	158
7.	Claim 7	158
(i)	Claim [7]: The method of claim 1, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.....	158

8.	Claim 8.....	159
	(i) Claim [8]: The method of claim 1, wherein the mapping function prevents copying of data of the file in the storage device.....	159
XII.	CONCLUSION.....	159

LISTING OF CLAIMS

Claim 1	
[1.pre]	A method of providing data security for use with a file system, the method comprising:
[1.1.1]	applying a mapping function to data block numbers that are associated with a file,
[1.1.2]	wherein the data block numbers are contained in an index node associated with said file; and
[1.2.1]	obtaining mapped data block numbers after applying the mapping function,
[1.2.2]	wherein the mapped data block numbers are addresses of data of the file in a storage device.
Claim 2	
[2]	The method of claim 1, further comprising: storing a mapping of the data block numbers and the mapped data block numbers.
Claim 3	
[3.1]	The method of claim 1, wherein a read operation for the file comprises: obtaining the data block numbers of the file;
[3.2]	obtaining the mapped data block numbers of the file based on the mapping function that is applied to the data block numbers; and
[3.3]	reading the data in the mapped data block numbers in the storage device.
Claim 4	
[4.1]	The method of claim 1, wherein a write operation for the file comprises: allocating data block numbers of free data blocks in the storage device;

[4.2]	linking the data block numbers to a file name of the file;
[4.3]	applying the mapping function to the data block numbers in order to obtain the mapped data block numbers; and
[4.4]	writing the data of the file into the mapped data block numbers in the storage device.
Claim 5	
[5]	The method of claim 1, wherein the mapping function is an equation that is applied to each data block number of the file.
Claim 6	
[6]	The method of claim 1, further comprising: selecting a complexity level of the mapping function in order to set a security level and efficiency of the mapping function.
Claim 7	
[7]	The method of claim 1, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.
Claim 8	
[8]	The method of claim 1, wherein the mapping function prevents copying of data of the file in the storage device.
Claim 9	
[9.pre]	An apparatus for providing data security for use with a file system, the apparatus comprising:
[9.1.1]	a block distribution engine configured to apply a mapping function to data block numbers that are associated with a file,
[9.1.2]	wherein the data block numbers are contained in an index node associated with said file, and

[9.1.3]	obtain mapped data block numbers after applying the mapping function;
[9.2]	wherein the mapped data block numbers are addresses of data of the file in a storage device.
Claim 10	
[10]	The apparatus of claim 9, wherein the block distribution engine is configured to store a mapping of the data block numbers and the mapped data block numbers.
Claim 11	
[11.1]	The apparatus of claim 9, wherein for a read operation for the file, the block distribution engine obtains the data block numbers of the file from a file system, and
[11.2]	obtains the mapped data block numbers of the file based on the mapping function that is applied to the data block numbers;
[11.3]	and wherein a block device driver is configured to read the data in the mapped data block numbers in the storage device.
Claim 12	
[12.1]	The apparatus of claim 9, wherein for a write operation for the file, the block distribution engine is configured to receive allocated data block numbers of free data blocks in the storage device from a file system,
[12.2]	link the data block numbers to a file name of the file, and
[12.3]	apply the mapping function to the data block numbers in order to obtain the mapped data block numbers;
[12.4]	and wherein a block device driver is configured to write the data of the file into the mapped data block numbers in the storage device.

Claim 13	
[13]	The apparatus of claim 9, wherein the mapping function is an equation that is applied to each data block number of the file.
Claim 14	
[14]	The apparatus of claim 9, wherein a complexity level of the mapping function is selected in order to set a security level and efficiency of the mapping function.
Claim 15	
[15]	The apparatus of claim 9, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.
Claim 16	
[16]	The apparatus of claim 9, wherein the mapping function prevents copying of data of the file in the storage device.
Claim 17	
[17.pre]	An apparatus for providing data security for use with a file system, the apparatus comprising:
[17.1.1]	means for applying a mapping function to data block numbers that are associated with a file,
[17.1.2]	wherein the data block numbers are contained in an index node associated with said file; and
[17.2.1]	means for obtaining mapped data block numbers after applying the mapping function,
[17.2.2]	wherein the mapped data block numbers are addresses of data of the file in a storage device.
Claim 18	
[18.pre]	Article of manufacture, comprising:

[18.1]	a machine-readable medium having stored thereon instructions to:
[18.2.1]	apply a mapping function to data block numbers that are associated with a file,
[18.2.2]	wherein the data block numbers are contained in an index node associated with said file; and
[18.3.1]	obtain mapped data block numbers after applying the mapping function,
[18.3.2]	wherein the mapped data block numbers are addresses of data of the file in a storage device.

I, Nenad Medvidovic, PhD, declare as follows:

I. INTRODUCTION

1. I have been retained as a technical expert by counsel on behalf of FedEx Corporation and FedEx Office and Print Services, Inc. (“FedEx”/“Petitioner”). I understand that FedEx is requesting that the Patent Trial and Appeal Board institute an *inter partes* review (“IPR”) proceeding with respect to U.S. Patent No. 7,904,686 (“the ’686 patent”) (EX1001).

2. I have been asked to provide my independent analysis of the ’686 patent in light of certain prior art publications, and I have done so based on my personal knowledge and experience.

3. I am not, and never have been, an employee of FedEx. I received no compensation for this Declaration beyond my normal hourly compensation based on my time actually spent analyzing the ’686 patent, the prior art publications cited below, and various issues related thereto. I will not receive any added compensation based on the outcome of any IPR or other proceeding involving the ’686 patent.

II. QUALIFICATIONS

4. In formulating my opinions, I have relied on my knowledge, training, and experience in the relevant field. My education and experience are described

more fully in the attached curriculum vitae (Appendix A). For ease of reference, I have highlighted certain information below.

5. I received a Bachelor of Science degree, summa cum laude, from Arizona State University's Computer Science and Engineering department.

6. I received a Master of Science degree from the University of California at Irvine's Information and Computer Science department.

7. I received a Doctor of Philosophy degree from the University of California at Irvine's Information and Computer Science department. My dissertation was entitled "Architecture-Based Specification-Time Software Evolution."

8. For over thirty years, my research has focused on software systems; their design, development, maintenance, and evolution; their individual building blocks and overall architectures; and mechanisms for their implementation and deployment. My Ph.D. dissertation yielded a new design style, called C2, for the development and evolution of highly distributed software systems built out of independent components that exchange different types of information encoded in messages to accomplish specific goals. This research focused on the design principles behind those systems, the appropriate support for modeling the systems (both at the level of individual modules as well as their integrations into subsystems

and entire systems), and the infrastructure for their implementation, deployment, optimization, and long-term evolution.

9. My foundational work on the C2 architectural style focused specifically on flexibly dividing complex distributed applications into three classes of components: user interface (UI), processing (a.k.a. “business logic”), and data storage. In the context of the data storage components, I worked extensively to build abstractions that will allow for leveraging local file storage capabilities to support remote storage, manipulation, sharing, and processing of data, along with a range of schemes for ensuring system efficiency and data consistency. This work was applied across a wide range of domains, from traditional distributed systems, to mobile devices and applications, to computational and data grid as well as cloud-based systems, to embedded systems and autonomous robots.

10. This work was subsequently applied to and expanded for the needs of data grid systems, such as OODT, the data grid built in the early 2000s in collaboration with NASA’s Jet Propulsion Laboratory (JPL). Specifically, this work was part of the dissertation research by my PhD student Christian Mattmann, who was a JPL engineer at that time, and who was propelled by the success of this work to the position of JPL’s Chief Technology and Innovation Officer. In the context of the OODT project, we developed a novel architecture for efficient data storage and

access in highly distributed settings and showed how the highly popular published “anatomy and physiology” of grid systems (i.e., their prescriptive architectures) were flawed and, in fact, not adhered to by any of 18 data and computational grid systems in wide use at that time. My work proposed an alternative architecture that was more realistic and suited for grids, and I demonstrated its plausibility and efficiency in a project called DREAM, which was a redesign and reimplementation of NASA/JPLs data grid system OODT targeting mobile and resource constrained platforms.

11. Subsequent extensions of this work involved manually recovering the architecture of an early version of the Hadoop MapReduce architecture, including its distributed file system. This effort was in collaboration with two of Hadoop’s chief architects, and it served as the foundation for a large body of subsequent research on automating the recovery of and visualizing key software architectural facets of a number of large-scale software systems, including subsequent versions of Hadoop.

12. My research has been funded by U.S. government agencies (e.g., National Science Foundation, National Institutes of Health, Defense Advanced Research Projects Agency, NASA, Department of Homeland Security) and

industrial organizations (e.g., Boeing, IBM, Northrop Grumman, Xerox, Google, Bosch).

13. I am employed by the University of Southern California (“USC”) as a faculty member in the Computer Science Department and have been since January 1999. I am a tenured professor. Since July 2022, I have served as my Department’s Chair. Between January 2009 and January 2013, I served as the Director of USC’s Center for Systems and Software Engineering (USC-CSSE), a large research center that involved the work of a number of USC faculty, students, and staff as well as several dozen industry partners. Under my direction, USC-CSSE was involved in a range of cutting-edge projects for our industry partners, ranging from the Aerospace Corporation to Xerox. Between July 2011 and July 2015, I served as my Department’s Associate Chair for Ph.D. Affairs. Between July 2021 and June 2022, I served as my Department’s Vice Chair for Tenure Affairs.

14. In 2007 I was offered the position of Chief Software Engineer of the Transformational Satellite Communications System (TSAT) program. This was a \$4B U.S. Department of Defense (DOD) program, sponsored by the U.S. Air Force. The program’s charge was to build the satellite grid of the future. With my university’s agreement, I was prepared to go on a multi-year leave-of-absence

assignment until the TSAT project was operational. Unfortunately, the program's funding was cut before my involvement could begin.

15. I have created and/or taught undergraduate-, Master's-, and Ph.D.-level courses in software engineering, software requirements, software architecture, and embedded systems. As part of these courses, I have developed and taught modules, and created and guided projects that exposed students to the interplay between software and traditional computing devices on one hand, and a wide range of physical equipment on the other hand, including sensors, RFID tags, cameras, mobile robots, and wearable computing devices.

16. A significant portion of my professional activity centers on conducting research through advising Ph.D. students. I have graduated 21 Ph.D. students to date, and I advise seven students currently pursuing their Ph.D. My students have gone on to successful careers in top software companies (e.g., Google, NVIDIA, Intel, Snap), research institutes (e.g., NASA's Jet Propulsion Laboratory, Information Sciences Institute, Quandary Peak Research), and universities (e.g., University of Massachusetts, University of California, University of Edinburgh).

17. I served as Program Co-Chair for the flagship conference in my field—International Conference on Software Engineering (“ICSE”)—held in May 2011. I have served as Chair or Co-Chair for various other conferences in the Software

Engineering field, including: the Fifth Working Institute of Electrical and Electronics Engineers (“IEEE”)/IFIP Conference on Software Architecture, the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, the Fifteenth International Association for Computing Machinery’s (“ACM”) Special Interest Group on Software Engineering (“SIGSOFT”) Symposium on Component Based Software Engineering, the IEEE/CSSE/ISE Workshop on Software Architecture Challenges for the 21st Century, and the Doctoral Symposia at the Sixteenth ACM SIGSOFT International Symposium on the Foundations of Software Engineering and at the 44th International Conference on Software Engineering.

18. I have served as an editor of several peer-reviewed journals, including IEEE Transactions on Software Engineering, ACM Transactions on Software Engineering and Methodology, Journal of Software Engineering for Robotics, Elsevier Information and Software Technology Journal, Journal of Systems and Software, Journal of Software Engineering Research and Development, and Springer Computing Journal. I have also served as a guest editor of several special issues for other journals. Between January 2018 and December 2022, I served as Editor-in-Chief of IEEE Transactions on Software Engineering (“TSE”), the flagship software engineering research journal.

19. Between September 2013 and September 2015, I served as Chair of the ICSE Steering Committee. Both before that (2008-2013) and subsequent to that (2015-present) I have been a member of the ICSE Steering committee. I am currently a member of the Steering Committee of the joint European Conference on Software Engineering and the Symposium on the Foundations of Software Engineering (“ESEC/FSE”). From July 2015 to September 2017, I chaired the Steering Committee of FSE, while it was a standalone symposium. I have served as a member of the Steering Committees of the Working IEEE/IFIP Conference on Software Architecture on two separate occasions, including presently.

20. Between July 2015 and June 2018, I served as Chair of the ACM SIGSOFT, the largest professional organization in my field. During the past year, I have chaired the committee in charge of ACM’s Software System Award, one of the most important technical awards in my field.

21. I co-authored Software Architecture: Foundations, Theory, and Practice, a widely used textbook in software systems architecture.

22. I have served as editor of various books in the Software Engineering field, including Proceedings of the 3rd International Conference on Self-Adaptive and Self-Organizing Systems, Proceedings of the Warm-Up Workshop for the 32nd International Conference on Software Engineering, Proceedings of the 5th Working

IEEE/IFIP Conference on Software Architecture, and 40 Editions of ICSE—the Ruby Anniversary Celebration.

23. I have authored or co-authored nearly 300 papers in the Software Engineering field. My work has been cited over 21,000 times. My most cited paper alone has been cited nearly 3,200 times; that paper recently received a decade’s Most Influential Paper Award from IEEE TSE. A paper I co-authored in the 1998 ICSE, my field’s flagship conference, was given ten years later, in 2008, that conference’s Most Influential Paper Award. Another of my publications received a Most Influential Paper Award in 2020. Several of my other publications have received “best paper” or “most cited” awards.

24. My research frequently results in the development of software tools that accompany and demonstrate the published ideas. A paper describing a security-vulnerability detection and remediation tool for Android devices built in my lab, won the Best Tool Paper award at the 2017 Automated Software Engineering Conference. That idea was subsequently awarded a U.S. patent. I am a co-inventor on two additional issued patents in distributed systems design, implementation, deployment, and computer security.

25. I have been named a Distinguished Scientist of the ACM. I have been elected a Fellow of the IEEE. Fellow of the IEEE is IEEE's highest grade; less than 0.1% of its membership earns this grade each year.

26. A detailed list of my other professional activities, memberships, and speaking engagements is included in my CV.

III. LEGAL PRINCIPLES

27. In forming my analysis and conclusions expressed in this Declaration, I have applied the legal principles described in the following paragraphs, which were provided to me by counsel for the Petitioner.

A. Person of Ordinary Skill in the Art

28. I understand that the factors considered in determining the ordinary level of skill in a field of art include: the level of education and experience of persons working in the field; the types of problems encountered in the field; the teachings of the prior art regarding solutions to such problems; and the sophistication of the technology at the time of the alleged invention. I understand that a person of ordinary skill in the art ("POSITA") is not a specific real individual, but rather is a hypothetical individual having the qualities reflected by the factors above and knowledge of all relevant prior art references, including the references I cite below.

B. Obviousness

29. I have been informed that a patent claim is invalid as “obvious” in light of one or more prior art references if it would have been obvious to a POSITA at the time of the alleged invention, taking into account (1) the scope and content of the prior art, (2) the differences between the prior art and the claims, (3) the level of ordinary skill in the art, and (4) any so called “secondary considerations” of non-obviousness, which include: (i) “long-felt but unresolved need” for the claimed invention, (ii) commercial success attributable to the claimed invention, (iii) unexpected results of the claimed invention, and (iv) “copying” of the claimed invention by others.

30. While I do not know the exact date that the alleged invention of the ’686 patent was made, I do know that the ’686 patent’s earliest claimed priority date is July 31, 2007. For purposes of my obviousness analysis, I have applied a date of July 31, 2007 as the date of the alleged invention (“Critical Date”), although in many cases the same analysis would hold true even if the date of the alleged invention were earlier or later.¹

¹ Note, however, that I do not concede that the ’686 patent is entitled to its earliest claimed priority date.

31. I have been informed that a claim can be obvious in light of a single prior art reference or multiple prior art references. To be obvious in light of a single prior art reference or multiple prior art references, there must be a reason that would have prompted a POSITA to modify or supplement the single prior art reference, or to combine two or more references, in a manner that provides the elements of the claimed invention. This reason may come from a teaching, suggestion, or motivation to combine, or may come from the reference(s) themselves, the knowledge or “common sense” of a POSITA, or from the nature of the problem to be solved, and this reason may be explicit or implicit from the prior art as a whole. I have been informed that, under the law, the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results. I also understand it is improper to rely on hindsight in making the obviousness determination.

32. I understand that an obviousness determination also requires that a POSITA would have had a reasonable expectation of success. This concept has been explained to me as relating to the relative likelihood or predictability—from the perspective of a POSITA—of successfully modifying the prior art in a manner that would meet the claimed limitations of the patent being challenged (here the ’686 patent). I understand that the expectation of success need only be “reasonable” and,

thus, does not require the absolute certainty gleaned from physically creating the proposed prior art modification.

C. Claim Interpretation

33. I understand that, for purposes of my analysis in this *inter partes* review proceeding, the terms appearing in the patent claims should be interpreted according to their “ordinary and customary meaning.” In determining the ordinary and customary meaning, I understand that the words of a claim are first given their plain meaning that those words would have had to a person of ordinary skill in the art at the time of the alleged invention. I also understand that the structure of the claims, the specification and file history may be used to better construe a claim. Moreover, I understand that even treatises and dictionaries may be used, albeit under limited circumstances, to determine the meaning attributed by a person of ordinary skill in the art to a claim term. Additionally, with the exception of my analysis of claim 17, I did not construe any claim terms at least because it is evident that the prior art renders the claims obvious without defining the outer bounds of each claim term.

IV. MATERIALS CONSIDERED

34. My analysis in this Declaration is based on my knowledge and experience. Based on my above-described qualifications in Section II, the Patent Trial and Appeal Board should consider me to be an expert in the field. Also, based

on my experiences, I understand and know of the capabilities of persons of ordinary skill in this field.

35. As part of my independent analysis for this Declaration, I have considered the following: the '686 patent (EX1001) and its prosecution history (EX1002); the background knowledge/technologies that were commonly known to persons of ordinary skill as of the Critical Date; my own knowledge and experience gained from my work in the field; my experience in teaching and advising others in this field; and my experience working with others involved in this field. In addition, I have analyzed the following publications and materials:

EX1004	U.S. Patent No. 5,774,715 ("Madany")
EX1005	Locasto, et al., 2004. "CamouflageFS: Increasing the effective key length in cryptographic filesystems on the cheap." In Applied Cryptography and Network Security: Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004. Proceedings 2 (pp. 1-15). Springer Berlin Heidelberg ("Locasto")
EX1006	Leffler, et al., "The Design and Implementation of the 4.3 BSD UNIX Operating System," Addison-Wesley, 1989-90 ("Leffler")
EX1007	Downing, et. al., "Dictionary of Computer and Internet Terms," Ninth Edition, Barron's Educational Series, Inc., 2006 ("Downing").
EX1008	U.S. Pat. App. Pub. No. 2006/0184673 ("Liebman")
EX1009	U.S. Pat. App. Pub. No. 2012/0124121 ("Pope")
EX1010	Custer, "Inside the Windows NT File System," Microsoft Press, 1994 ("Custer")
EX1011	Garfinkel, "Practical Unix Security," O'Reilly & Associates, 1992 ("Garfinkel")

EX1012	U.S. Pat. App. Pub. No. 2005/0091487 (“Cross”)
EX1013	U.S. Pat. App. Pub. No. 2007/0168678 (“Meenakshisundaram”)
EX1014	Rivest, et al, 1998, August. “The RC6™ block cipher.” In First advanced encryption standard (AES) conference (p. 16) (“Rivest”)
EX1015	Arnold, “Unix Security: a practical tutorial” McGraw-Hill, 1992 (“Arnold”)
EX1016	Tanenbaum, et. al, “Modern Operating Systems, Second Edition,” Prentice Hall, 2001 (“Tanenbaum”)
EX1017	Kelsey, 2002, February, “Compression and information leakage of plaintext,” in International Workshop on Fast Software Encryption (pp. 263-276). Berlin, Heidelberg: Springer Berlin Heidelberg (“Kelsey”)

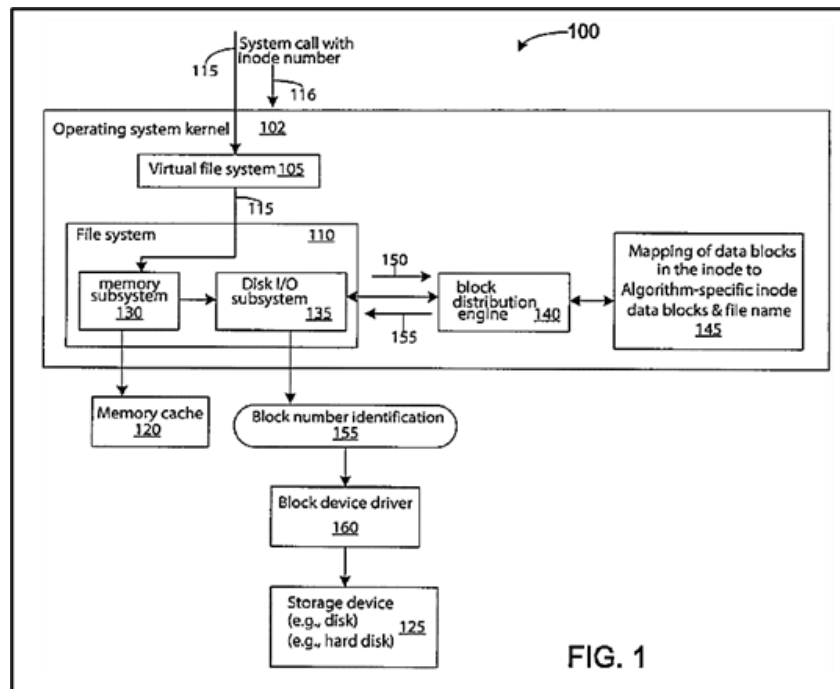
36. Note that my citations to non-patent literature throughout this Declaration reference the absolute page number added to the exhibit (as opposed to the original pagination of the document), except for Leffler (EX1006) which refers to the document’s original pagination. For patent literature, I’ve used the column/line numbers or paragraph numbers. Note also that, unless I’ve indicated otherwise, all emphasis in any quoted text are ones I have added.

37. Although I cite and quote to selected portions of various references in this Declaration, the reader should understand that these citations are representative examples. A POSITA would have viewed each reference in its entirety and in combination with other references. Accordingly, I intend the references identified in this Declaration to be viewed as incorporated in their entireties.

V. THE '686 PATENT

A. Pertinent Disclosure

38. The '686 patent "relate[s] generally to data security for use with a file system." EX1001, 1:6-7. Figure 1 of the '686 patent shows "a block diagram of an apparatus (system)...implemented in, for example, a computer." EX1001, 2:1-14. The block diagram shows an operating system 102 including a file system 110 (e.g., one or more "Unix file systems") that "will assign a unique inode for each file." EX1001, 2:15-28. "As known to those skilled in the art, an inode (index node) is a data structure that contains information about a file," including "the addresses of the file's data blocks on the disk." *Id.* Where, "[i]n previous systems, the data blocks of a file are linearly placed on a storage device and can be obtained by accessing the inode of the file," the '686 patent uses a block distribution engine 140 to perform "a block distribution algorithm...[that] will vary the distribution of the data blocks on the storage device [] in order to provide security to the data blocks." EX1001, 2:62-3:8. Accordingly, the '686 patent's independent claim 1 recites "applying a mapping function to data block numbers" and "obtaining mapped data block numbers" that "are addresses of data of the file in a storage device." EX1001, 6:12-20.



EX1001, FIG. 1.

VI. PERSON OF ORDINARY SKILL IN THE ART

A. A POSITA's Ordinary Level of Skill

39. Based on my knowledge and experience in the field and my review of the '686 patent and its file history, I believe that a POSITA would have had (1) at least a master's degree in computer science, computer engineering, or a related field, and (2) at least two years of professional experience involving the study and/or development of operating systems, file systems, and/or storage devices. Such experience could be obtained through research and study in a graduate program or through comparable exposure through industry employment. Additional education could substitute for professional experience, and professional experience for education.

40. My analysis and conclusions expressed in this Declaration are based on the perspective of a POSITA having this level of knowledge and skill. My education, technical expertise and personal knowledge discussed in Section II shows that I meet the qualifications of a POSITA.

B. A POSITA's Background Knowledge and Understanding of the Pertinent Technology

1. File Systems

41. As explained by well-known computer scientist Andrew Tanenbaum in his 2001 textbook called Modern Operating Systems, “computers are equipped with a layer of software called the operating system, whose job is to manage all these devices and provide user programs with a simpler interface to the hardware.” EX1016 (“Tanenbaum”), 27. A “key concept supported by virtually all operating systems is the file system” that “hide[s] the peculiarities of the disks and other I/O devices and present[s] the programmer with a nice, clean abstract model of device-independent files.” EX1016, 70. Files are necessary since “many applications” must retain information for “weeks, months, or even forever,” and such information “must not go away when a computer crash” occurs. EX1016, 467.

42. “System calls are obviously needed to create files, remove files, read files, and write files.” EX1016, 70. Accordingly, Tanenbaum identifies some of the more common file operations as: create, delete, open, close, read, write, append, seek

(specify from where to access a file's data), rename, etc. EX1016, 477-478. For example, “[b]efore a file can be read, it must be located on the disk and opened, and after it has been read it should be closed, so calls are provided to do these things.” EX1016, 70.

43. To allow a file to be accessed non-sequentially (i.e., at any position in the file) and efficiently, and to avoid a disk becoming fragmented (where some segments on the disk cannot be used), many file systems (e.g., “Unix V7 File System) use a data structure called an “**i-node (index node)**” which includes “disk addresses of the files blocks.” EX1016, 491-494, 496 (emphasis in the original), 545. Tanenbaum illustrates an i-node in FIG. 6-15, reproduced below.

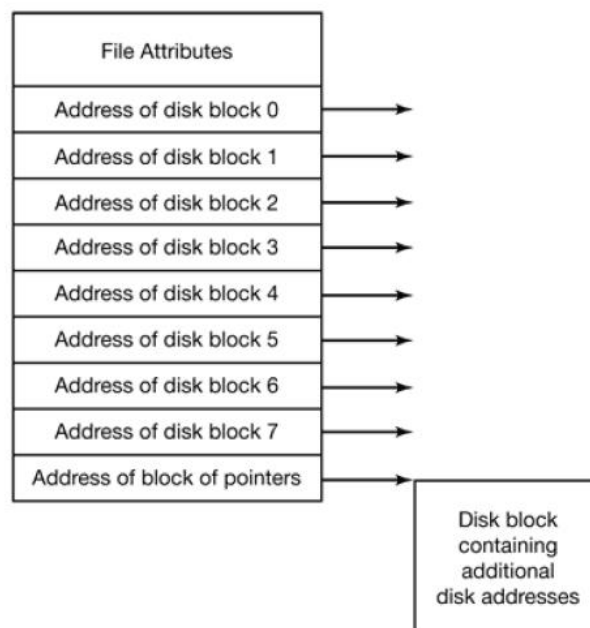


Figure 6-15. An example i-node.

EX1016, FIG. 6-15.

2. Data Security

44. The need for data security was well-known as of the Critical Date in 2007. For example, Garfinkel² states that “you should consider using encryption for every piece of sensitive information on your computer system” and “[y]ou simply can’t assure that the data on your computer will *never* fall into the wrong hands.” EX1011, 5 (emphasis in the original). Custer³ similarly explains that “[d]ata security is of primary importance to businesses and government agencies that process private or sensitive information...Such customers need guarantees that their data will be secure from unauthorized access.” EX1010, 5. Arnold⁴ echoes Custer’s statements, explaining that that “Government purchases are the largest market for computers. The government requires highly trusted systems.” EX1015, 6.

45. To provide data security, encryption techniques were introduced into existing file systems, often called an “encrypted file system.” For example, Cross⁵ discloses an encrypted file system. EX1012, Abstract, *generally*. Additionally,

² “Practical Unix Security,” O’Reilly & Associates, 1992

³ “Inside the Windows NT File System,” Microsoft Press, 1994

⁴ “Unix Security: a practical tutorial” McGraw-Hill, 1992

⁵ U.S. Pat. App. Pub. No. 2005/0091487

Meenakshisundaram's⁶ "database system provides an encrypted file system that is optimized for database storage security." EX1013, [0088].

46. Passwords can also be used to protect files. For example, Tanenbaum writes that file "attributes relate to the file's protection and tell who may access it and who may not," e.g., "[i]n some systems the user must present a password to access a file, in which case the password must be one of the attributes.: EX1016, 475.

47. Access Control Lists (ACLs) provide another mechanism for restricting access to a file. An ACL associates each object with "an (ordered) list containing all the domains that may access the object, and how." EX1016, 781. For example, "[e]ach file has an ACL associated with it." *Id.* Tanenbaum provides an example in which an ACL has a "first entry [that] says that any process owned by user A may read and write the file," and a "second entry says that any process owned by user B may read the file," while "[a]ll other accesses by these users and all accesses by other users are forbidden." *Id.* This example is illustrated in FIG. 9-25, reproduced below, where "R" indicates read access, "W" indicates writes access, and "X" indicates execute access. For example, "RW" indicates read and write access; "RWX" indicates read, write and execute access.

⁶ U.S. Pat. App. Pub. No. 2007/0168678

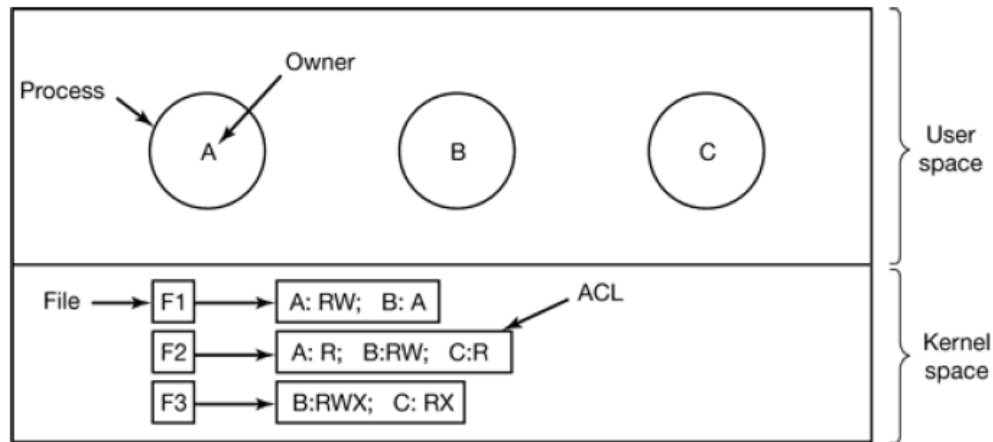


Figure 9-25. Use of access control lists to manage file access.

EX1016, FIG. 9-25.

48. More generally, Tanenbaum devotes a chapter to security since “[m]any companies possess valuable information that they guard closely.” EX1016, 706. Tanenbaum’s chapter discusses cryptography, user authentication, potential attacks, etc. EX1016, 707-807.

49. Note that while both compression and encryption transform one set of bits into an altered set of bits, the purpose of each differs. For example, Kelsey⁷ explains that “[t]he real reason for using compression algorithms isn’t to increase security, but rather to save on bandwidth and storage.” EX1017, 5. Kelsey further explains that the “goal of any compression algorithm [] is to reduce the redundancy of some block of data, so that an input that required R bits to encode can be written

⁷ Kelsey, 2002, February, “Compression and information leakage of plaintext,” in International Workshop on Fast Software Encryption (pp. 263-276). Berlin, Heidelberg: Springer Berlin Heidelberg.

as an output with fewer than R bits.” EX1017, 3. In contrast, “[c]ryptosystems like AES and triple-DES are designed to encrypt a sequence of input bytes (the plaintext) into a sequence of output bytes (the ciphertext) in such a way that the output carries no information about that plaintext except its length.” EX1017, 1.

VII. OVERVIEW OF CONCLUSIONS FORMED

50. This Declaration explains the conclusions that I have formed based on my independent analysis. To summarize those conclusions:

- Based upon my knowledge and experience, and my review of the materials listed above, I believe that Claims 1-2, 5-10, and 13-18 of the ’686 patent are obvious in view of Madany (EX1004) and Locasto (EX1005). *See* Grounds 1A and 1B at Sections VIII and IX.
- Based upon my knowledge and experience, and my review of the materials listed above, I believe that Claims 1-18 of the ’686 patent are obvious in view of Madany, Locasto, and Leffler (EX1006). *See* Ground 2 at Section X.
- Based upon my knowledge and experience, and my review of the materials listed above, I believe that Claims 1-8 of the ’686 patent are obvious in view of Leffler and Locasto. *See* Ground 3 at Section XI.

VIII. GROUND 1A: MADANY AND LOCASTO

A. Madany (EX1004)

51. Madany describes a “method, apparatus, and computer-usable medium for compressing data in a file system utilizing the concept of ‘holes’.” EX1004, Abstract. Madany’s “FIG. 1 shows in general a structure for a file system 200” that contains “a hierarchy of directory structures 201, a file table 301 and a mapping table 305...for each entry in the file table 301.” 5:29-35. The directory contains “entries of file names within the file system and corresponding file numbers that allow files to be indexed to an entry in the file table.” EX1004, 5:32-34. In Madany, “[e]ach entry 302 in the file table 301 identifies a file of the file system and is commonly known as an Inode.” EX1004, 5:35-37.

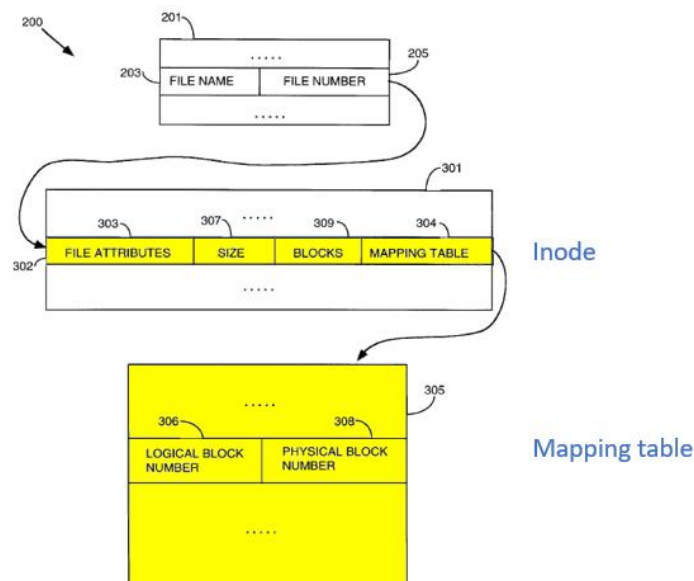


Figure 1

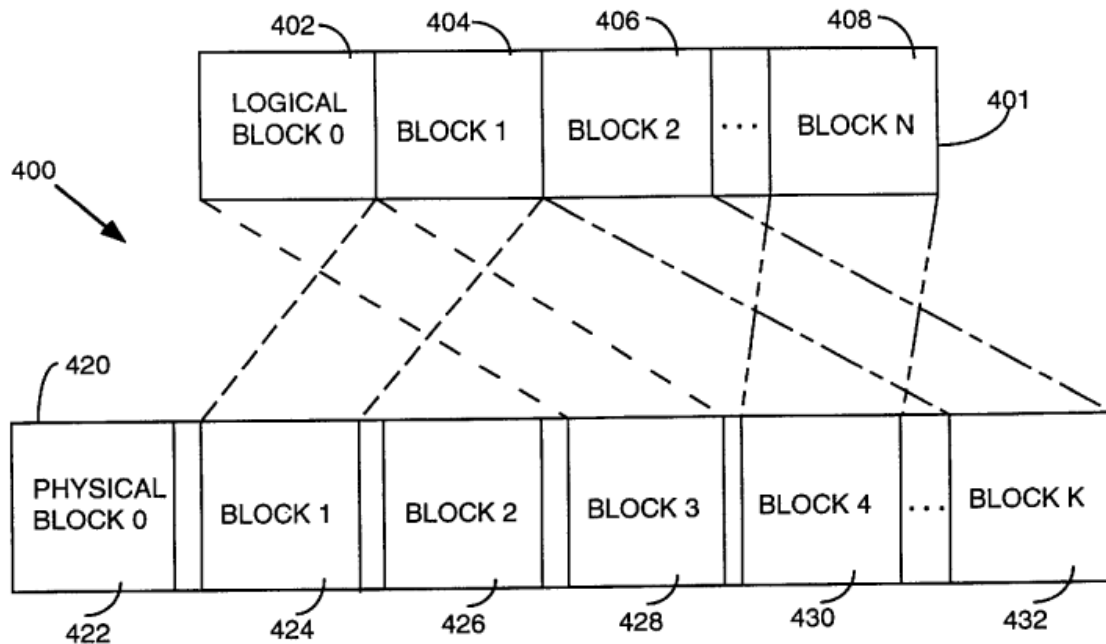
EX1004, FIG. 1 (annotated)

52. Madany explains that the “mapping table 305 contains information on how the logical blocks within a file are mapped to physical blocks on a disk” and the “physical block number identifies the location on the disk where the logical block is to be stored.” EX1004, 5:52-54, 5:63-64. For example, Madany explains that the first entry in the mapping table 305⁸ “maps to the physical block number 308 for logical block number 0 of the file.” EX1004, 5:58-60.

53. Madany further describes the mapping process in reference to Figure 2, which “shows symbolically how the logical blocks of the file may be mapped to physical blocks on a hard disk.” EX1004, 6:3-4. As shown below, Figure 2 depicts a logical file 401 and a hard disk 420. EX1004, 6:5-7. The “hard disk is typically separated into physical blocks, but it is not always true that the logical blocks of the file are stored in contiguous physical blocks on a hard disk,” e.g., “the logical blocks of logical file 400 are shown being mapped to noncontiguous physical blocks of the hard disk 420,” so “when the logical file 401 is written to the hard disk the logical blocks will be written to noncontiguous physical blocks.” EX1004, 6:7-15. “In this example, logical block zero 402 is written to physical block three 428;” “[l]ogical

⁸ Madany contains a clear typographical error that a POSITA would have identified and easily resolved—i.e., identifying the mapping table 305 as “file table 305.” EX1004, 5:58-59. The file table is reference number 301 and does not include the “[t]he first entry 306.” EX-1004, 5:31.

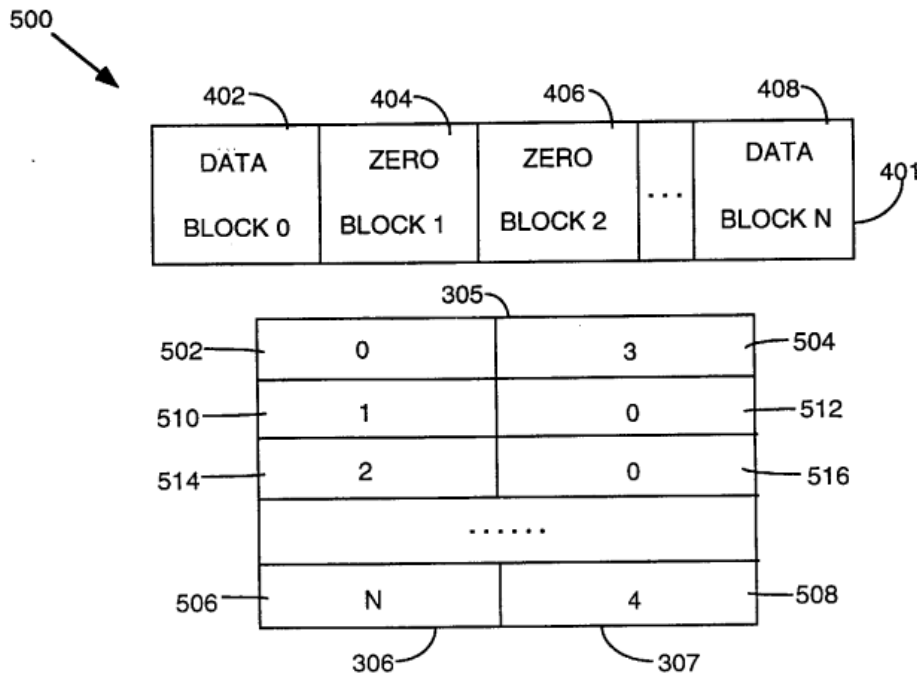
block one 404 is written to physical block one 424;” etc. EX1004, 4:15-16. Therefore, Madany’s “mapping table 305 of Figure 1 is used to indicate to which physical blocks the logical blocks are written.” EX1004, 6:25-29.



EX1004, FIG. 2.

54. Madany also explains that “a logical block within a file [can] contain[] very little data or may in fact be completely empty, that is, the logical block is full of zeroes,” e.g., “if logical block one 404 was completely full of zeroes then this zero block would be written in its entirety to physical block one 424.” EX1004, 6:33-35. This approach “is not desirable as a physical block completely full of zeroes is a waste of space.” EX1004, 6:35-37. Madany calls blocks containing only zeroes “holes.” EX1004, 6:38-39.

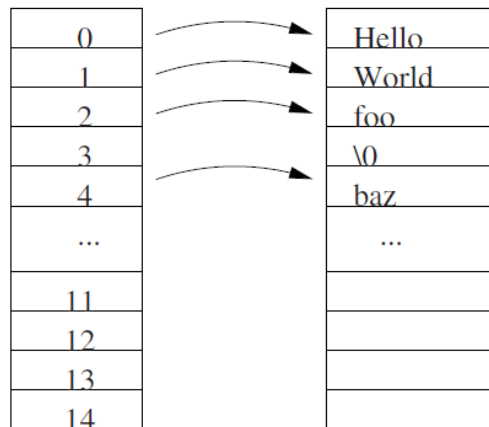
55. To avoid such waste, in the mapping table, Madany represents holes using a special address, which is “not an actual physical block address where data will be written but a single address that indicates that the corresponding logical blocks are in fact full of zeroes.” EX1004, 6:59-61. Madany’s Figure 3 shows an example in which “logical block zero 402 and logical block N 408 each contain non-zero data,” but “logical blocks one and two both contain nothing but zeroes.” EX1004, 6:43-46. In this example, “a hole 512 or 516 is represented by the number zero.” EX1004, 6:57-58.



EX1004, FIG. 3.

B. Locasto (EX1005)

56. Locasto is a technical paper titled “ComouflageFS: Increasing the Effective Key Length in Cryptographic Filesystems on the Cheap.” EX1005, 1. Locasto explains that file systems can use indexed allocation in which “the operating system maintains an index of entries per file that can quickly address any given block of that file.” EX1005, 3. Locasto’s Figure 2 shows “a single-level direct-mapped index,” where “[t]he file index points directly to blocks with plaintext data.” EX1005, 3.

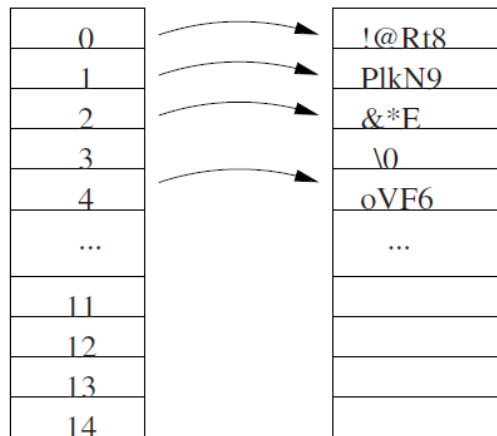


EX1005, FIG. 2.

57. Like Madany, Locasto explains that “[h]oles in the file may exist; reading data from such holes returns zeroed-out blocks. EX1005, 3. The index of Figure 2 “has a hole at the third block position.” EX1005, 4.

58. Locasto further explains that “[c]ryptographic file systems encrypt the stored data, which leaves the index structure identical but protects the contents of

the data blocks, as shown in Figure 3,” reproduced below. EX1005, 3. In this example, “[t]he indexing has not changed, merely the contents of the data blocks,” which are encrypted. EX1005, 3, 5. As with FIG. 1, “the file hole at block three is present.” EX1005, 5.



EX1005, FIG. 3.

59. To improve security “without incurring the usual performance penalty,” Locasto introduces a “spreading schedule” that “randomly distribute[s] the real file data throughout a large address space so that an attacker would have to first guess the spreading schedule before he attempts a brute force search on the rest of the key.” EX1005, 2 (altered emphasis), 4.

60. Locasto explains that in conventional systems, “the number of the index entry is calculated by taking the floor of the current file position ‘pos’ divided by the block size,” such that “ $index = pos / blocksize$.” EX1005, 4. The computed “index number is then used to derive the logical block number (the block on disk) where

the data at ‘pos’ resides,” such that “*lbn = get from index(index)*,” where “lbn” is the logical block number. EX1005, 4. Consider an example where a block is 128 bytes⁹ and the file is referenced at byte position (pos) 10. In that case, the index would be 0 (10 divided by 128 and truncated), which is the first table position.¹⁰ The block number is then obtained by using the index (0) to retrieve the block number at that position in the table. Continuing the example, if the file is referenced at byte position 130, the index would be 1, the second table position, and the block number is retrieved from that position in the table.

61. To be clear, when Locasto refers to a “logical block number” as “the block on the disk,” a POSITA would understand this to actually mean the number of the physical disk block where the data resides since a physical block is the block on the disk. Additionally, a POSITA would readily understand that this is a mere choice of terminology made by Locasto, and that “logical” in the term is intended to modify “number” rather than “block” – a logical number of the physical block.

62. To avoid confusion with respect to Locasto’s terminology and Madany’s use of the terms “logical block number” and “physical block number,”

⁹ Actual block sizes would be much larger. I use this small number to simplify the example.

¹⁰ Arrays are typically indexed from 0 to N-1 instead of from 1 to N, so the first position in the array is 0, the second position is 1, and so on.

this Declaration will use the generic term “block number” when referencing and quoting from Locasto. Further, to be clear, my conclusions in this Declaration would not materially change regardless of Locasto’s block number terminology. As discussed below, my analysis adopts Madany’s block number terminology.

63. Moving on, Locasto alters the conventional two-step approach by introducing a spreading schedule such that the “initial calculation of the index is performed, but before the [] block number is derived, a pseudo-random permutation (PRP) function takes the calculated index and the bits of the spreading seed to return a new index value,” which Locasto calls a “map” function in the second equation below. EX1005, 4-5. Thus, Locasto “calculat[es] a new entry in the file index, rather than using the strictly sequential ordering” of the blocks. EX1005, 5. Locasto calls the mapped virtual data blocks “an implicit *virtual index*” which is used “to spread the file data blocks throughout the file’s address space.” EX1005, 6.

$$index = pos / blocksize$$

$$index = map(index, spread_seed)$$

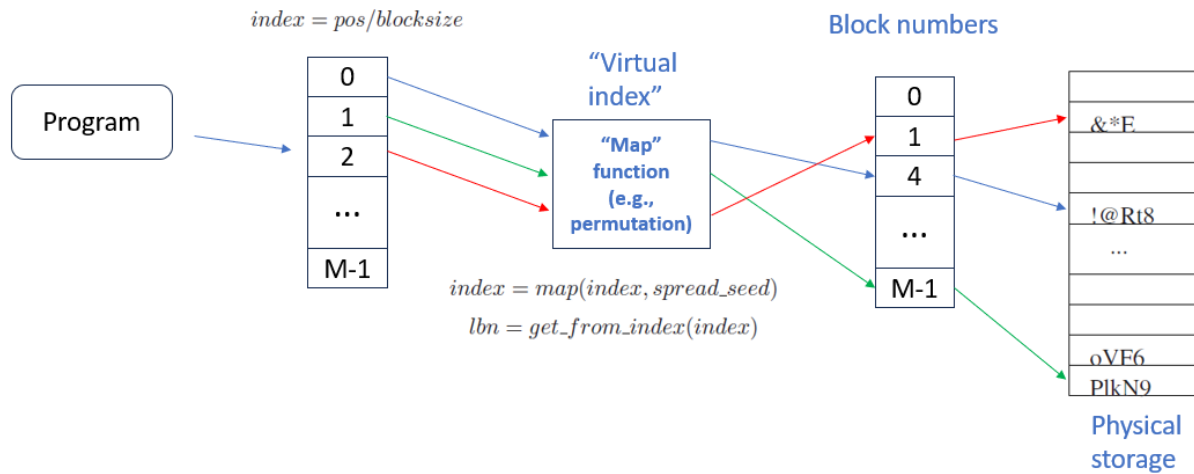
$$lbn = get_from_index(index)$$

EX1005, 5.¹¹

¹¹ While Locasto uses “spread_seed” as a variable in this function, Locasto describes it as a “spreading seed.” This Declaration uses “spreading seed” in the formulas.

64. Locasto further explains that a “variable length block cipher is utilized as a pseudo-random permutation (PRP) to map sequential block indices to ostensibly random indices.” EX1005, 10. Moreover, Locasto states that “the variable length block cipher can be built upon any existing block cipher and stream cipher.” EX1005, 10. “RC6 was chosen for this implementation because its construction makes it applicable to small block sizes and RC4 was utilized due to its simplicity.” EX1005, 10. (RC6 is described in EX1014.)

65. Locasto’s process is illustrated in the visual aid below. In the first step, an “index entry is calculated by taking the floor of the current file position ‘pos’ divided by the block size.” EX1005, 4. In the second step, an index (e.g., 0) is “spread” to a block number (e.g., 4, shown with the blue arrow) using a “map” function, e.g., a pseudo-random permutation (e.g., $index = map(index, spread\ seed)$) followed by an identification of the block number (e.g., $lbn = get_from_index(index)$). EX1005, 5. Using the same process, index number 1 is mapped (spread) to block number M-1 (green arrow) and block number 2 is mapped to mapped block number 1 (red arrow).



Locasto: Visual Aid

C. The Madany-Locasto Combination

66. As discussed, Madany describes a “method, apparatus, and computer-usable medium for compressing data in a file system utilizing the concept of ‘holes’.” EX1004, Abstract. Madany explains that “mapping table 305 contains information on how the logical blocks within a file are mapped to physical blocks on a disk” and the “physical block number identifies the location on the disk where the logical block is to be stored.” EX1004, 5:52-54, 5:63-64; §VIII.A, *supra*. Similarly to Madany, Locasto describes a file system in which “indices” are mapped to block numbers. EX1005, 3-4; §VIII.B, *supra*. Locasto introduces two security features: [i] a cryptographic file system that encrypts the stored data and a mapping function that uses a permutation function and [ii] a spreading seed to randomly

distribute real file data throughout a large address space. EX1005, 3-6, FIGS. 3-4; §VIII.B, *supra*.

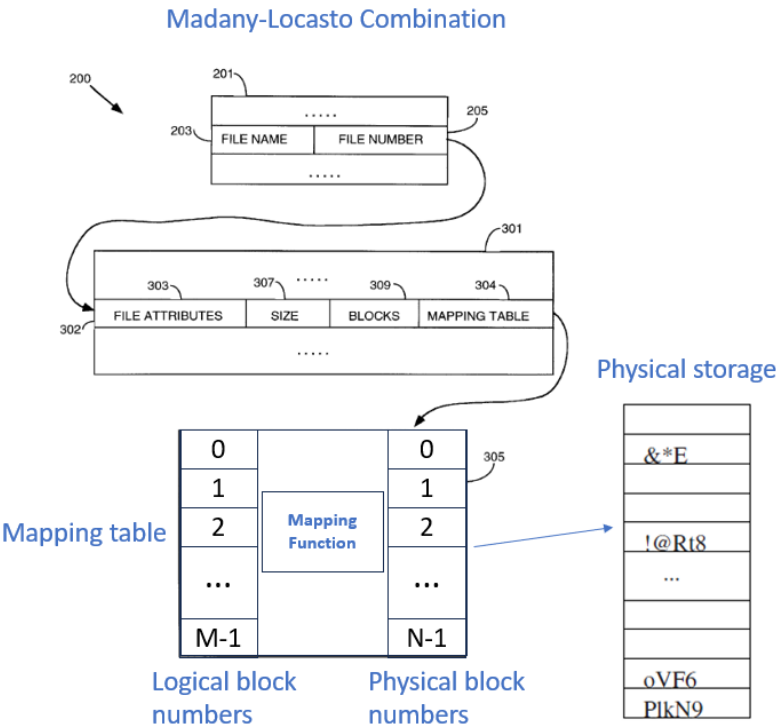
67. In the Madany-Locasto Combination, the file system Madany teaches is enhanced with security features like those taught in Locasto, (e.g., adding a mapping function and encrypting the stored data, as illustrated below). As described by Locasto, such “[c]ryptographic file systems provide data confidentiality by employing encryption to protect files against unauthorized access.” EX1005, 1. Thus, as explained further below in §VIII.C.1, a POSITA would have appreciated the predictable benefits of incorporating Locasto’s security features with Madany’s efficient file system and associated data structures. For example, as I noted above at §VI.B.2, Custer explains that “[d]ata security is of primary importance to businesses and government agencies that process private or sensitive information...Such customers need guarantees that their data will be secure from unauthorized access.” EX1010, 5. Additionally, Garfinkel states that “you should consider using encryption for every piece of sensitive information on your computer system” and “[y]ou simply can’t assure that the data on your computer will *never* fall into the wrong hands.” EX1011, 5 (emphasis in the original).

68. Moreover, the POSITA would have understood the compatibility of Loasto’s teachings with Madany, such that Locasto’s technical improvements could

be implemented while preserving the fundamental aspects of Madany's disclosure. For example, Madany describes Unix file systems and Locasto describes the "Linux Ext2fs" file system. EX1004, 1:45-50, 3:14-18, 5:41-42; EX1005, 3, 8. A POSITA would have understood that Linux is a Unix-like operating system. For example, Liebman (U.S. Pat. App. Pub. No. 2006/0184673), a contemporaneous patent publication to the '686 patent, explains that the "solution of the present invention is built upon a unique usage of an underlying feature of the Linux Operating System" and is "also found in other Unix or Unix-like Operating Systems." EX1008, [0027]. Pope (U.S. Pat. App. Pub. No. 2012/0124121), a second contemporaneous patent publication to the '686 patent, similarly describes "Unix or Unix-like environments such as Linux." EX1009, [0400].

69. A POSITA would have understood that Madany's file system architecture presents two apparent options for incorporating Locasto's mapping function. First, it would have been understood or obvious to the POSITA that Locasto's mapping function would be deployed effectively as a mechanism to convert Madany's logical block numbers to physical block numbers, as shown in the visual aid below. This first version of the Madany-Locasto Combination is the subject of Ground 1A. (Section VIII.) Second, it would have been understood by the POSITA that Locasto's mapping function also would be deployed effectively as a mechanism to map the physical block numbers in Madany's mapping table to

addresses on the physical storage disk. This second version of the Madany-Locasto Combination is the subject of Ground 1B. (Section IX.) Both of these options for combining Madany and Locasto are straightforward, and neither would require undue experimentation, as will be demonstrated in the detailed elaboration of Ground 1A and Ground 1B below.



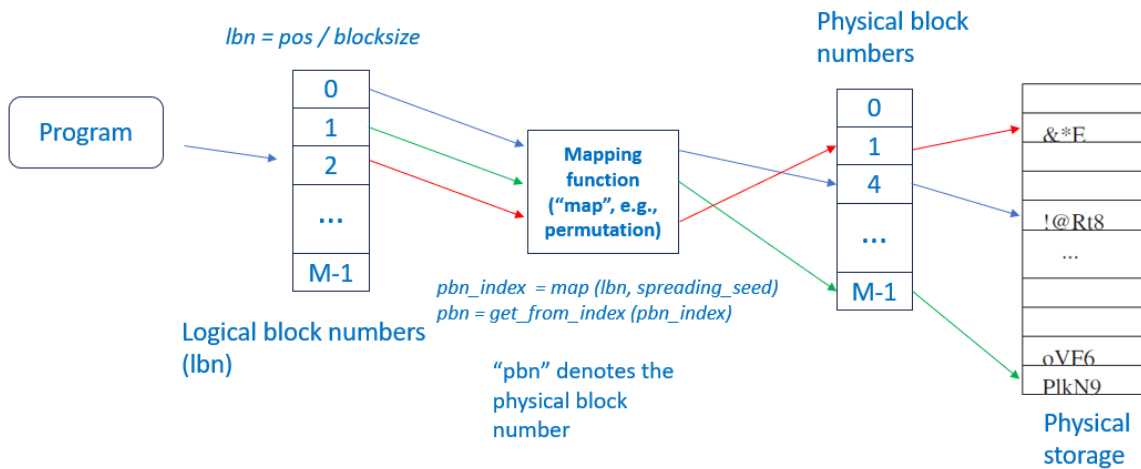
Madany-Locasto Combination: Visual Aid.¹²

¹² This visual aid, adapted from Madany and Locasto, is not proposing a physical or real-world integration of these reference. Instead, my intent is to merely show an example of how the Madany-Locasto Combination integrates the conceptual teachings of these references for the Board’s convenience. The same is true of all visual aids in this Declaration.

70. In operation, when a program accesses a file at a given position, the Madany-Locasto Combination of Ground 1A determines the inode (e.g., Inode 301,¹³ per Madany), and uses a mapping table pointer in the inode (e.g., mapping table pointer 304, per Madany) to determine the mapping table (e.g., mapping table 305, per Madany). EX1004, 5:38-40, 48-49. Then, as shown in the visual aid above, the Combination employs a process like Locasto's (*see* §VIII.B) to facilitate the conversion of logical block numbers in Madany's mapping table to physical block numbers (*see* §VIII.A). More specifically, logical block numbers are "spread" to physical block numbers using Locasto's "map" function, e.g., a pseudo-random permutation function and a "spreading seed." EX1005, 4-5. The mapping function "randomly distribute[s] the real file data...throughout a large address space so that an attacker would have to first guess the spreading schedule before he attempts a brute force search on the rest of the key." EX1005, 4.

¹³ While Madany capitalizes "Inode," this Declaration adopts the more common spelling as "inode," except when quoting from Madany.

Madany-Locasto Combination



Madany-Locasto Combination: Visual Aid

71. As shown in the visual aid above, the Madany-Locasto Combination applies Locasto's mapping function (*map*) to the logical block numbers (*lbn*) in Madany's mapping table and the bits of a *spreading seed* to determine the physical block number (*pbn*):

$$pbn_index = map(lbn, spreading_seed)$$

$$pbn = get_from_index(pbn_index)$$

EX1005, 5-6. These two functions can be expressed equivalently as the single (composite) mapping function:

$$pbn = get_from_index(map(lbn, spreading_seed))$$

1. Reasons to Combine

72. The following reasons, taken separately or collectively, support the above-described Madany-Locasto Combination.

73. First, a POSITA would have been motivated to pursue the Madany-Locasto Combination for the purpose of maintaining confidentiality of the data in Madany's file system. EX1004, Abstract; EX1005, 3-6. As Locasto explains, "[c]ryptographic file systems provide data confidentiality by employing encryption to protect files against unauthorized access." EX1005, 1. The POSITA would have appreciated the importance of protecting sensitive data against unauthorized access and, thus, would have been prompted to bolster Madany's file system disclosure with data encryption teachings from a supplemental reference like Locasto. As I explained in §VI.B.2, many contemporaneous sources explicitly acknowledged the need for enhanced data protection. This became especially important as computer systems became increasingly accessible remotely over public networks, creating additional opportunities for bad actors to gain access to a computer. For example, Garfinkel states that "you should consider using encryption for every piece of sensitive information on your computer system" and "[y]ou simply can't assure that the data on your computer will *never* fall into the wrong hands." EX1011, 5 (emphasis in the original). Above, at Section VI.B.2, I provide further explanation

and supporting evidence regarding the POSITA's knowledge and motivation to pursue solutions for improving data security.

74. Second, a POSITA would have been motivated to pursue the Combination to achieve in Madany the specific benefits espoused in Locasto. For example, Locasto's approach to encryption "reduc[es] the performance penalty associated with using larger key sizes without decreasing the level of security." EX1005, 2. As Locasto explains, by "using only part of the key for cryptographic operations" and "[t]he rest of the key...to unpredictably spread the data into the file's address space," "a user may decrease the performance penalty they pay for employing a cryptographic file system." *Id.*; see also 1 ("improved performance relative to a cipher with a larger key-schedule"), 12 ("calculation of a new index has a negligible impact on performance") ("we have increased the effective cryptographic key length by 28 bits, correspondingly increasing an attacker's work factor by 2^{28} ").

75. From the POSITA's perspective, Locasto's balance of security and performance would have been an advantageous addition to Madany's file system, which pursues efficiency along an orthogonal dimension—storage size—that is compatible with that pursued by Locasto. For example, Locasto explains that "[s]ince encryption is an expensive operation, there is a trade-off between

performance and security that a system designer must take into consideration.” EX1005, 1. In response, Locasto, “reduc[es] the performance penalty associated with using larger key sizes without decreasing the level of security.” EX1005, 2. Locasto’s solution “can increase an attacker’s work factor by 2^{28} without noticeable performance loss for legitimate users.” *Id.* To a POSITA, achieving enhanced security comparable to robust encryption algorithms without sacrificing performance would have been a predictable and highly desirable outcome of the Madany-Locasto Combination.

76. Third, a POSITA would have been prompted to pursue a combination of Madany and Locasto based on contemporaneous trends in file system design and marketing. For example, a POSITA would have known that data security solutions were incorporated into “popular file systems,” such as the “WINDOWS NT file system” that Madany expressly identified. EX1004, 1:44-52. As explained in Custer, the WINDOWS NT reference book that I discussed above at Section VI.B.2 and that Madany also cited, “Data security is of primary importance to businesses and government agencies that process private or sensitive information...Such customers need guarantees that their data will be secure from unauthorized access.” EX1010, 5. Madany also references and applies aspects of the UNIX operation system. EX1004, 1:44-52, 5:29-51. And it was well known that skilled artisans and market participants were actively pursuing secure variations of the UNIX operating

system, some of which included built-in encryption programs. For example, as I explained at Section VI.B.2, Arnold explains that “Government purchases are the largest market for computers. The government requires highly trusted systems. The UNIX operating system has been augmented by many firms in order to gain better ratings.” EX1015, 6. Additionally, Garfinkel writes that “[a]lthough there is no standard UNIX program that performs encryption using the DES [encryption algorithm], Sun Microsystems’ operating system does include such a program[.]” EX1011, 14. Accordingly, the POSITA would have sought data security improvements to Madany’s file system in an effort to compete and distinguish over competitor products in the marketplace.

77. Fourth, by the time of the ’686 patent in 2007, the scientific and patent literature was replete with proposals to incorporate data encryption techniques into existing file systems, often called an “encrypted file system.” For example, as I explained above at Section VI.B.2, Cross discloses an encrypted file system. EX1012, Abstract, *generally*. Meenakshisundaram’s “database system provides an encrypted file system that is optimized for database storage security.” EX1013, [0088]. Moreover, Locasto provides a “Related Work” section that details numerous different types of secure file systems and related technologies for encrypting, hiding, and otherwise protecting confidential data. EX1005, 13. A POSITA would have been motivated by this widely and growing recognized need, contemporaneous

design, and research trends to incorporate Locasto's encryption-focused teachings into a file system like Madany's.

78. Fifth, a POSITA would have been led to the Madany-Locasto Combination because it merely involves the application of a known technique (Locasto's data encryption) to a known system (Madany's file system) to achieve predictable results (a file system with improved data security). Each element in combination merely performs the same function as it does separately. For example, Locasto uses the known "Blowfish encryption algorithm [15] to encrypt each block of data." EX1005, 9 (citing Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," in Fast Software Encryption, Cambridge Security Workshop Proceedings, pages 191–204. Springer-Verlag, December 1993). Further, Madany's provides an overview of known file systems technology (e.g., EX1004, 5:29-6:37, FIGS. 1, 2), including "Inodes [which] are described in 'The Design and Implementation of the 4.3BSD UNIX Operating System', Chapter 7." Additionally, both Madany and Locasto describe similar file systems running in Unix and Unix-like operating systems, and thus could be predictably combined. EX1004, 1:47-50, 5:41-44; EX1005, 1. Thus, Locasto's mapping function (prototyped in Linux, a Unix-like operating system), which is applied to block numbers, can easily be applied to Madany's block number produced in a similar

operating system (Unix). (EX1008, [0027], EX1009, [0400], both describing Linux as Unix-like.)

2. Reasonable Expectation of Success

79. A POSITA would have had a reasonable expectation of success in creating a combination like the Madany-Locasto Combination, which would have required only routine programming knowledge well within the skill of a POSITA. For example, the Madany-Locasto Combination includes components of a file system and known encryption techniques to enhance the security of such file systems. *Supra*, §§VIII.A–C.

80. Integrating these aspects of Madany and Locasto would have been a routine programming task for a POSITA. For example, both Unix and the Linux kernel were largely implemented in C, a programming language a POSITA would have known well. Further, Linux is an open-source software project to which many POSITA's contributed. My own students worked under my guidance on specialized Linux drivers in the context of my "Software Engineering for Embedded Systems" graduate course in the early 2000s; additionally, my PhD students and I worked on Linux modifications in the context of an Android security research project in mid-2010s. Additionally, Locasto states that "CamouflageFS is a rather straightforward extension to the standard Ext2 file system for the Linux 2.4.19 kernel," and "[t]he

current implementation can coexist with normal file operations and does not require any extra work to use regular Ext2 files.” EX1005, 8. Since the extension is straightforward, applying its techniques to Madany would have been well within the ability of a POSITA.

81. Additionally, the Madany-Locasto Combination preserves Madany’s core benefits, e.g., using “holes” to avoid using physical storage for blocks that have never been written. EX1004, 2:60-67, As discussed, through the lens of a POSITA, this Combination would have amounted to nothing more than the use of a known technique to improve similar file systems in a similar way and combining prior art elements according to known methods to yield the predictable and beneficial results described above.

D. Claim Element Analysis

1. Claim 1

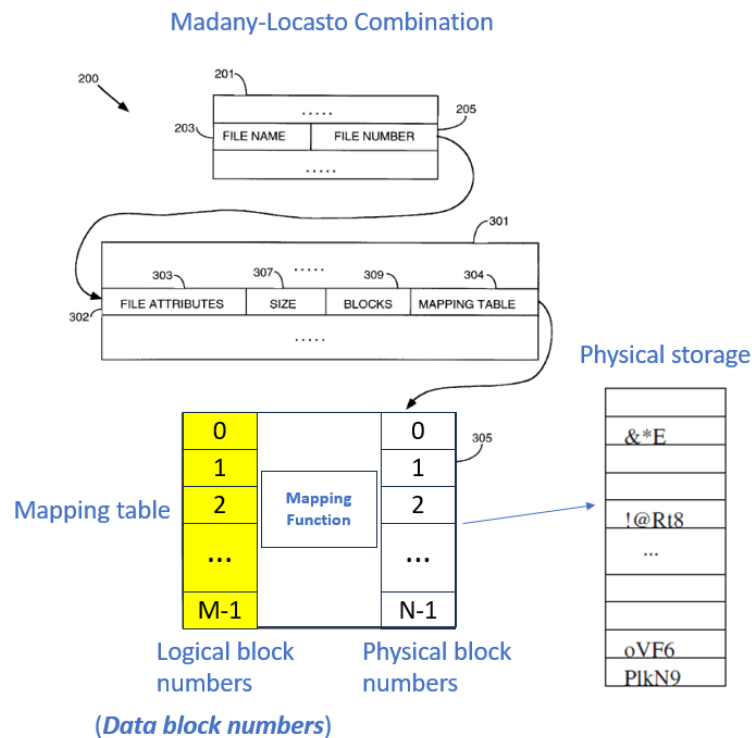
(i) *Element [1.pre]: A method of providing data security for use with a file system, the method comprising:*

82. The Madany-Locasto Combination renders obvious Element [1.pre] and Claim 1 as a whole. For example, the Combination includes Madany’s description of a “**method**, apparatus, and computer-usable medium for compressing data in a **file system** utilizing the concept of ‘holes’.” EX1004, Abstract. The Combination additionally includes Locasto’s explanation that “[c]ryptographic file

systems provide **data** confidentiality by employing encryption to protect files against unauthorized access.” EX1005, 1; *see also supra* §§VIII.A-C (discussing the combination). Locasto then describes a cryptographic file system (Camouflage FS) that “offers improved performance relative to a cipher with a larger key-schedule, while providing the same **security** properties.” EX1005, 1.

(ii) **Element [1.1.1]: applying a mapping function to data block numbers that are associated with a file,**

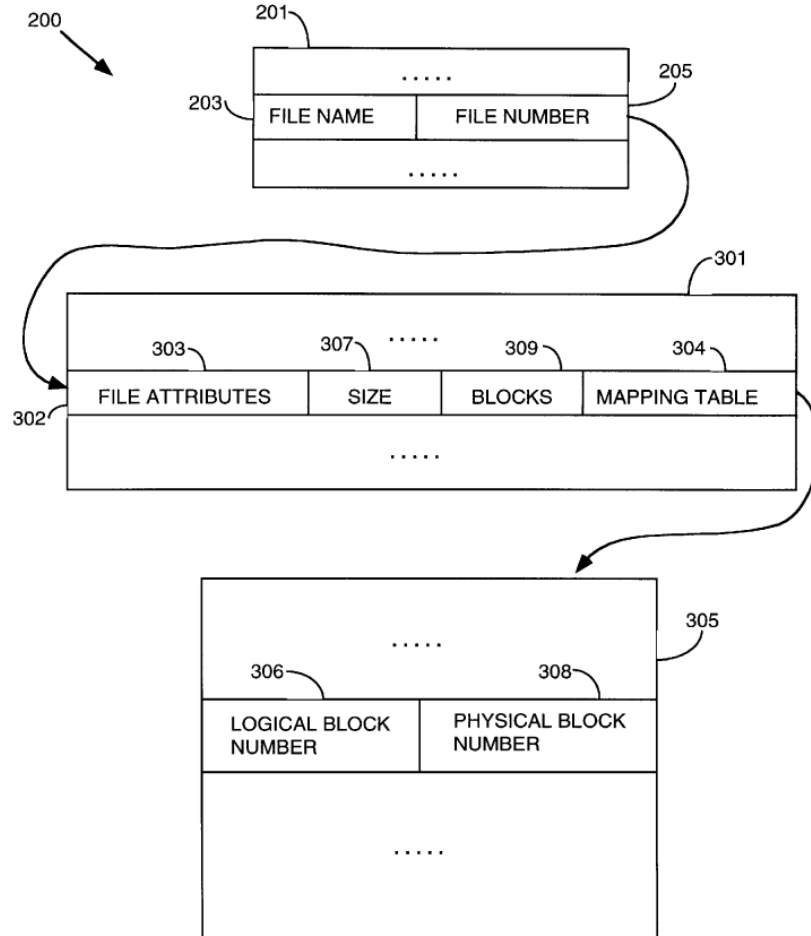
83. The Madany-Locasto Combination renders obvious Element [1.1.1]. For example, the Combination includes **data block numbers** in the form of Madany’s logical block numbers.



Madany-Locasto Combination: Visual Aid

84. A POSITA would have understood or found obvious that logical block numbers are ***data block numbers*** at least since logical block numbers correspond to data managed by a file system and stored on a physical disk. In other words, as would have been clear to a POSITA, Madany's logical ***block numbers*** are used by the file system to store and access ***data***. For example, as illustrated in the visual aid above, to determine physical block numbers, the Combination applies a mapping function to logical block numbers (***data block numbers***). The physical block numbers are used to read/write data on physical storage. Thus, a POSITA would have understood that logical block numbers correspond to the physical data stored on a physical disk.

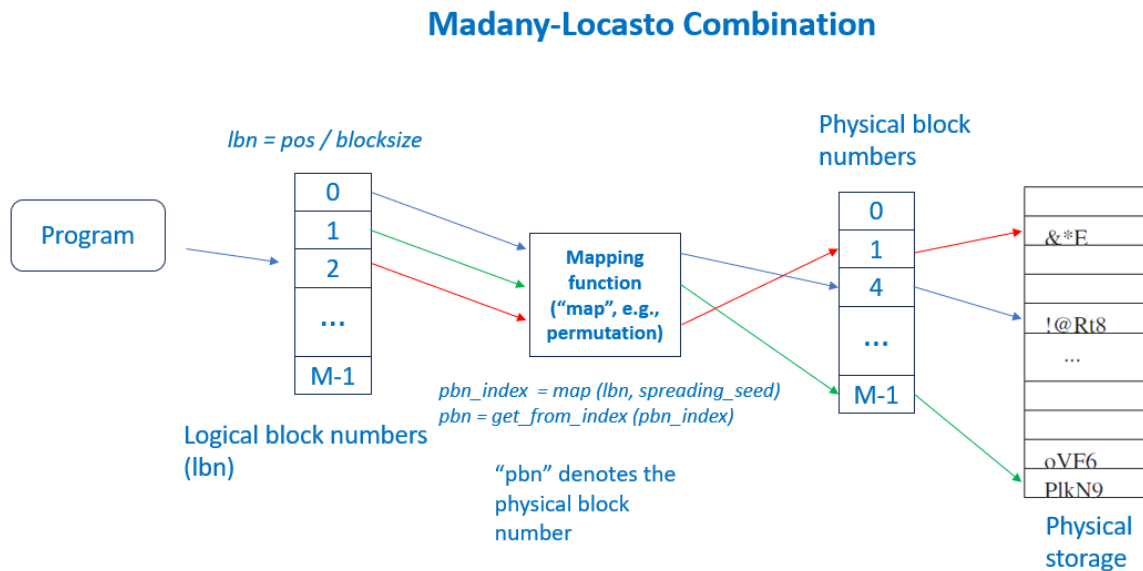
85. The Combination further teaches or renders obvious that the logical block numbers (***block numbers***) are ***associated with a file***. For example, Madany explains that "[t]o effectively support block I/O, a file system typically divides the array of bytes in a given file into a set of fixed size logical blocks." EX1004, 2:33-35. Thus, it would have been understood by or obvious to a POSITA that the logical ***block numbers are associated with a file*** since the file is assigned to the logical blocks corresponding to those logical block numbers. Madany illustrates the association in Figure 1 (below), which shows information about the file (e.g., file name 203, file number 205) linked to the file's file table 301, which is linked to the file's mapping table 305 containing logical block numbers 306.



EX1004, Figure 1.

86. As discussed at Sections VIII.A-C, the Madany-Locasto Combination ***applies a mapping function*** to logical block numbers (***data block numbers***). For example, Locasto explains that “a pseudo-random permutation (PRP) function takes the calculated” logical block number “and the bits of the spreading seed to return a” physical block number. EX1005, 4-5. Locasto provides the formula in which the physical block number is computed by applying two functions, *map* and *get_from_index* such that the results of *map(logical_block_number,*

spreading_seed) are used in the function *get_from_index()*. EX1005, 5. The process is illustrated in the visual aid below.

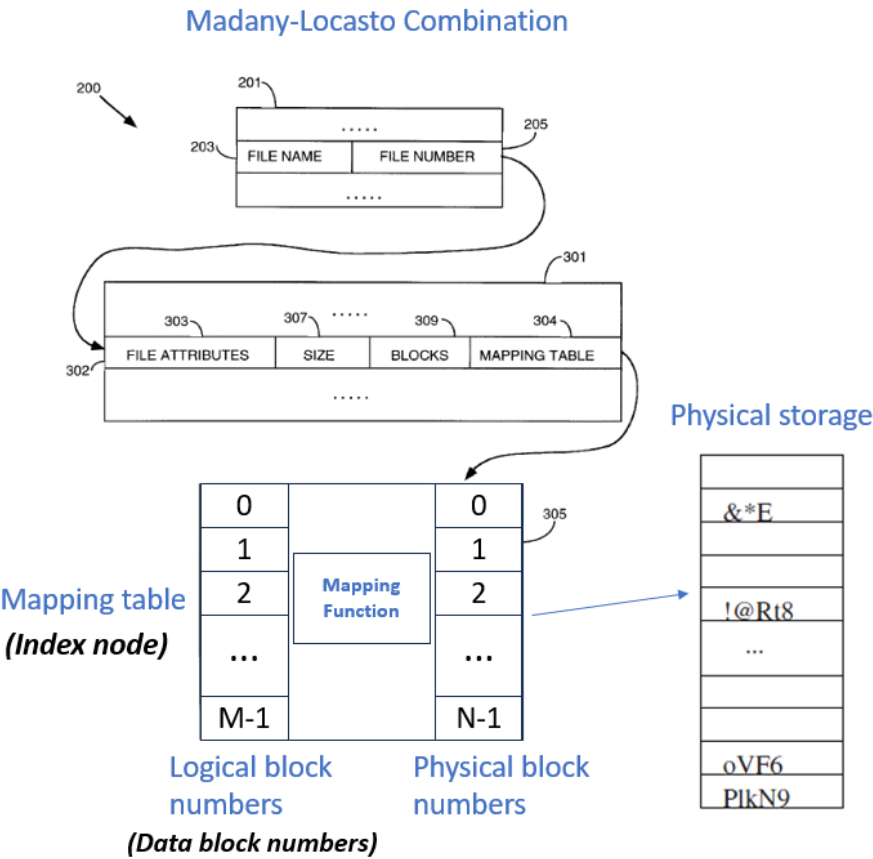


Madany-Locasto Combination: Visual Aid

(iii) ***Element [1.1.2]: wherein the data block numbers are contained in an index node associated with said file; and***

87. The Madany-Locasto Combination renders obvious Element [1.1.2]. As discussed at Element [1.1.1], the Combination includes logical block numbers (***data block numbers***) ***contained*** in a mapping table. "The mapping table ***contains*** information on how the ***logical blocks*** within a file are mapped to physical blocks on a disk." EX1004, 5:52-54 (reference numbers omitted); *see also generally* 5:29-7:65, Figures 1-4. For example, Madany explains, the first entry in the mapping

table “maps to the physical block number for logical block number 0 of the file,” and so on. EX1004, 5:58-66 (reference numbers omitted).



Madany-Locasto Combination: Visual Aid

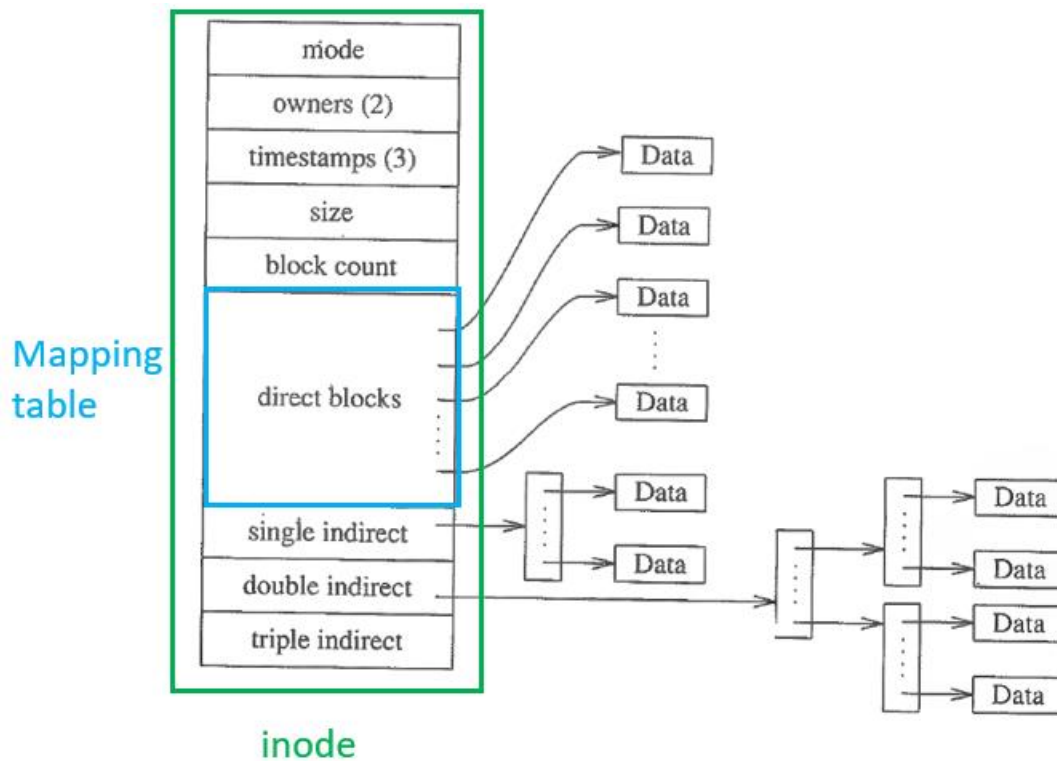
88. As would have been understood by or obvious to a POSITA from Madany’s teachings, the mapping table serves as *an index node* by containing a mapping from a first index (logical block numbers) to a second index (physical block

numbers). For example, Leffler¹⁴ explains that an index node “contains an array of pointers to the blocks in the file,” allowing the system to “convert from a logical block number to a physical sector number by indexing into the array using the logical block number.” EX1004, 194. Since Madany’s mapping table also contains an array of pointers to blocks in a file and allows the system to convert from logical block numbers to physical block numbers, a POSITA would have understood and found obvious that Madany’s mapping table is a type of index node.

89. Alternatively, while Madany explains that the inode 302 contains a pointer 304 to the mapping table 305, it would have nevertheless been understood by or obvious to a POSITA that the inode can include the mapping table itself (e.g., as implemented in the 4.3BSD Unix operating system), increasing efficiency by eliminating the need to follow a pointer to access the mapping table.

90. For example, describing the 4.3BSD Unix operating system, Leffler’s FIG. 6 clearly shows the mapping (labeled “direct blocks”) is present in the inode itself.

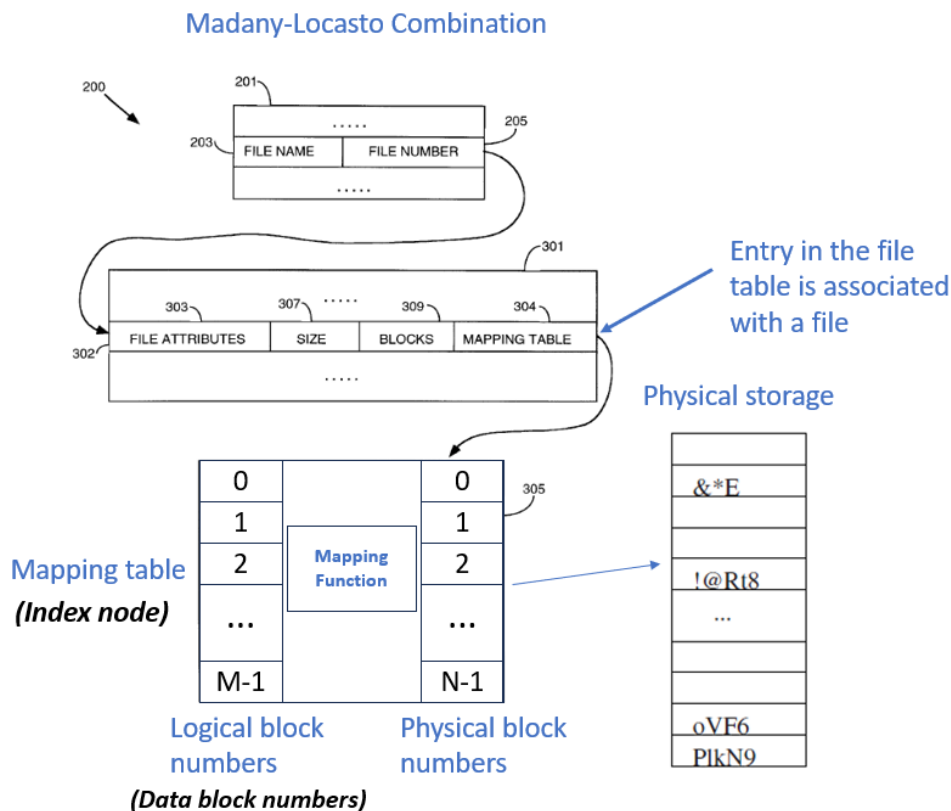
¹⁴ Leffler is the textbook, “The Design and Implementation of the 4.3 BSD UNIX Operating System,” Addison-Wesley, 1989-90 (EX1006). I describe it in more detail in §X.A.



EX1006, FIG. 7.6 (annotated)

91. This predictable variation is consistent with Madany’s explanation that “[i]t should be appreciated that other structures are possible for a file system, and that many file systems may be suitable for use with the present invention.” EX1004, 5:47-51. In this alternative, the inode (***index node***) contains the logical block numbers (***data block numbers***) since the inode contains the mapping table 305. Such a variation would have been known to a POSITA, e.g., from Tanenbaum’s discussion of a “method for keeping track of which blocks belong to which file” by “associat[ing] with each file a data structure called an i-node (index-node), which lists the attributes and disk addresses of the files blocks.” EX1016, 496.

92. Further, as would have been understood by or obvious to a POSITA from Madany's teachings, the mapping table (as well as the inode) is **associated with a file**. For example, Madany explains that "[e]ach entry 302 in the file table 301 identifies a **file** of the file system and is commonly known as an Inode." EX1004, 5:35-37; *see also generally* 5:29–51 (explaining that a file system 200 includes a directory 201 including file names 203 and file numbers 205, with each file number 205 pointing to a particular inode 302 in the file table 301). Madany further explains that "[t]here is a mapping table 305 for each entry [Inode 302] in the file table 301." EX1004, 5:31-32.



Madany-Locasto Combination: Visual Aid

(iv) ***Element [1.2.1]: obtaining mapped data block numbers after applying the mapping function,***

93. The Madany-Locasto Combination renders obvious Element [1.2.1]. As discussed at Sections VIII.B-C the Madany-Locasto Combination applies Locasto's ***mapping function***, e.g., "a pseudo-random permutation (PRP) function [that] takes [Madany's logical block number] and the bits of the spreading seed to return a new index value." EX1005, 4-5.

$$pbn_index = map(lbn, spreading_seed)$$

94. The physical "block number is then derived from this new index." EX1005, 5.

$$pbn = get_from_index(pbn_index)$$

Id.

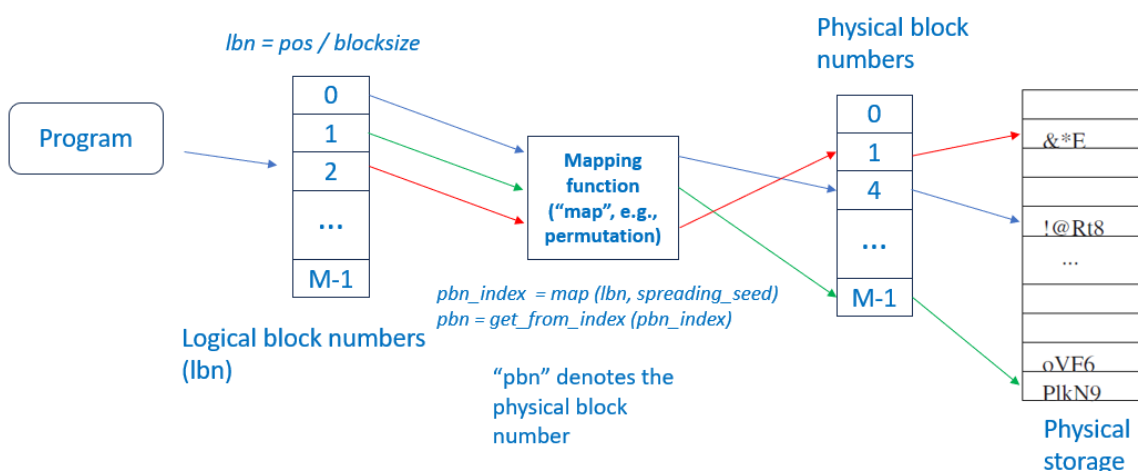
95. Thus, in the Madany-Locasto Combination, the physical block numbers (***mapped data block numbers***) ***are obtained after applying*** the "map" (***mapping***) ***function***.

(v) ***Element [1.2.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.***

96. The Madany-Locasto Combination renders obvious Element [1.2.2]. For example, as Madany teaches, "the physical block number [(***mapped data block***

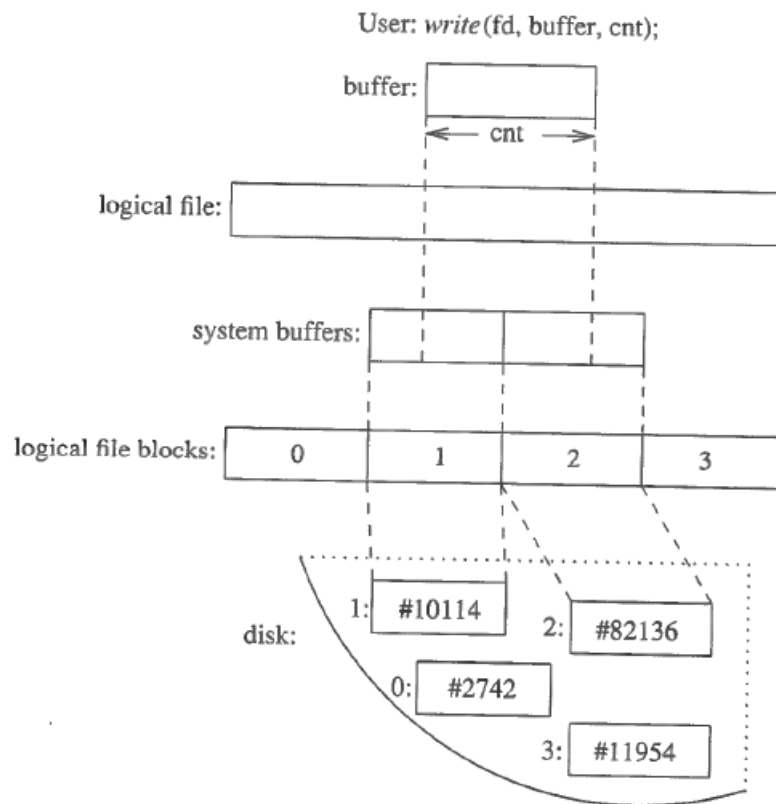
numbers]] identifies the location [(**addresses of data of the file**)] on the disk [(**storage device**)] where the logical block is to be stored.” EX1004, 5:63-64; see also 2:34–39 (explaining how a file system “divides the array of bytes in a given **file** into a set of fixed size logical blocks), 2:40–46 (explaining that “[t]he **data** in each logical block is stored in a physical block on the disk” and “[a] physical block is the location on the disk to which the file system maps a logical block”), 2:47–51 (explaining that “logical to physical block mappings” provide “the physical block **address** for [the corresponding] logical block”). Likewise, Locasto teaches that the mapping function is applied to derive **mapped data block numbers** that correspond to “the block[s] on the disk” where the data resides (**addresses of the data of the file**). EX1005, 4-5.

Madany-Locasto Combination



Madany-Locasto Combination: Visual Aid

97. Additionally, a POSITA would have understood that ***mapped data block numbers are addresses of data*** contained in ***the file*** in a storage device. For example, Leffler explains that “each logical block is stored in a *physical block* on the disk.” EX1006, 193 (emphasis in the original). Leffler’s FIG. 7.5 (reproduced below) illustrates this concept.



EX1006, FIG. 7.5.

2. Claim 2

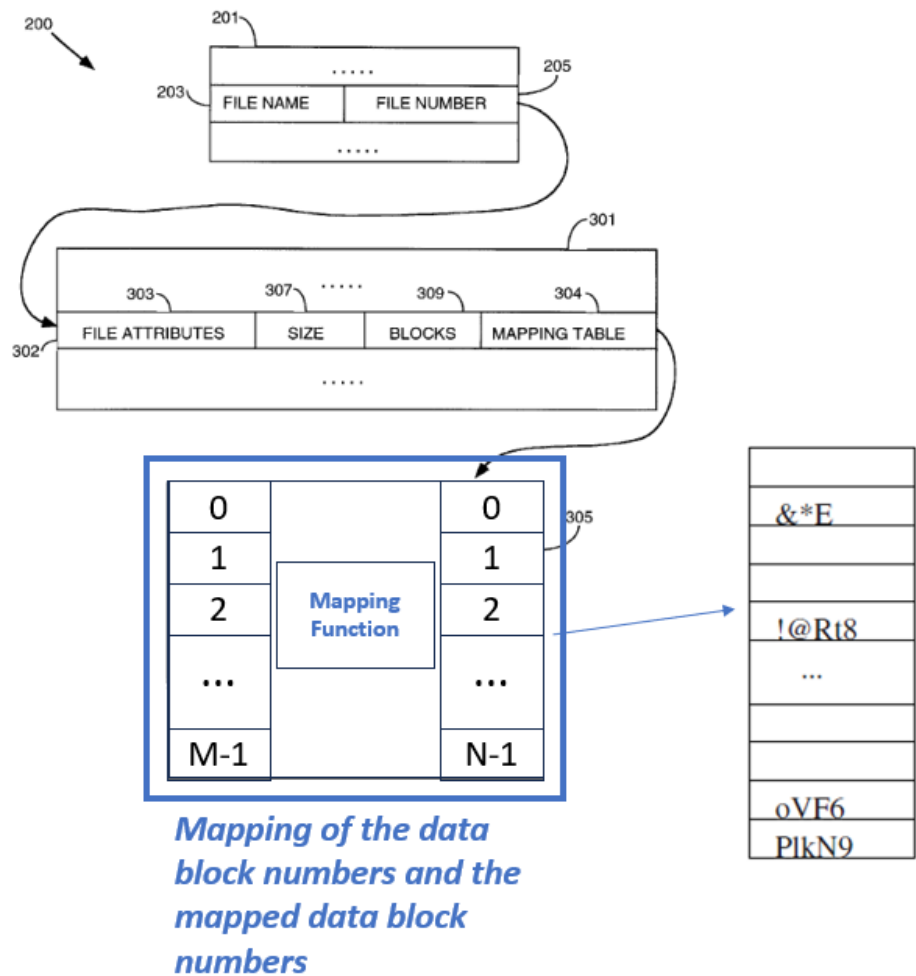
- (i) ***Claim [2]: The method of claim 1, further comprising: storing a mapping of the data block numbers and the mapped data block numbers.***

98. The Madany-Locasto Combination renders obvious Claim [2]. For example, the notion of ***storing a mapping*** would have been obvious to a POSITA in view of Madany's teachings pertaining to an updatable "***mapping*** table" where the Madany-Locasto Combination maintains (***stores***) a ***mapping*** between logical block numbers (***data block numbers***) and physical block numbers (***mapped data block numbers***). EX1004, Abstract ("[a] mapping table in a file system maps the logical blocks of a file to actual physical blocks on disk where the data is stored"), 5:52-60, FIG. 1.

99. Moreover, as discussed, in the Madany-Locasto Combination, the conversion from logical to physical block numbers involves applying Locasto's ***mapping*** function (e.g., a pseudo-random permutation), which a POSITA would have understood to be ***stored*** as coded instructions for performing the spreading algorithm. EX1005, 4-5. For example, Locasto explains that "either SHA-1 or an adaptation of RC6" can be used "during the calculation of the spread index entries." EX1005, 9. A POSITA would have understood that SHA-1 refers to the Secure Hash Algorithm 1 which would be stored as coded instruction representing that

algorithm. Similarly, RC6 is the “Rivest Cipher 6,” which is a block cipher that would also be stored as coded instructions. (RC6 is described in EX1014, “The RC6 Block Cipher.”)

Madany-Locasto Combination



Madany-Locasto Combination: Visual Aid

100. Additionally, the Combination’s use of Locasto’s mapping function is analogous to the use of a function in the ’686 patent. The ’686 patent used an example function of $f(x)=5x$, further noting that “ $f(x)$ can be programmed to be other

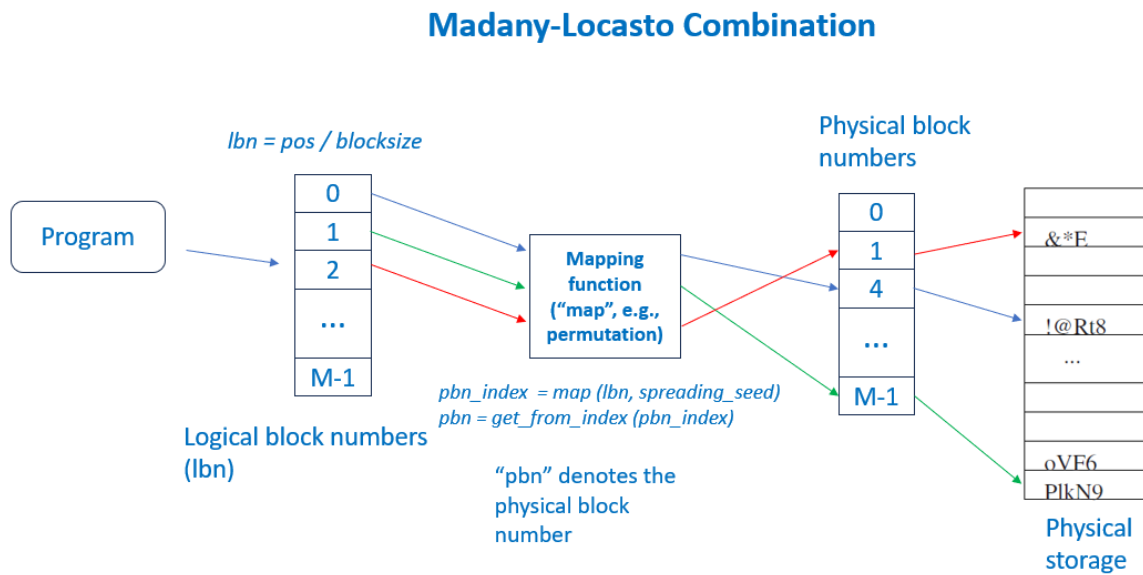
functions” and “[t]he function $f(x)$ is only an example of an algorithm that the algorithm designer can program for use by the block distribution engine 140” and “the block distribution engine 140 can apply other algorithms to the data block numbers (x).” EX1001, 3:65-4:5. Additionally, “[t]he algorithm designer has the choice to design different functions $f(x)$ that differ in complexity level, based on, for example, considerations of efficiency and security level of the mapping 145.” EX1001, 4:7-10. A POSITA would have understood that Locasto’s functions are such examples. Thus, the use of Locasto’s algorithms is analogous to the use of “5x” in the ’686 patent, but Locasto’s algorithms provide even better security.

3. Claim 5

- (i) ***Claim [5]: The method of claim 1, wherein the mapping function is an equation that is applied to each data block number of the file.***

101. The Madany-Locasto Combination renders obvious Claim [5]. For example, as discussed at Sections VIII.C, VIII.D.1 (Element [1.1.1]), the Madany-Locasto Combination ***applies a mapping function*** to logical block numbers (***data block numbers***) ***of a file***. EX1005, 4-5. The concept of performing this technique at ***each*** logical block number is rudimentary and would have been understood by or obvious to a POSITA. First, when each data block of a file is written, the mapping is performed on the data block number of that data block. EX1005, 4 (“The purpose of the spreading schedule is to randomly distribute the real file data throughout a

large address space”). Therefore, since all of the data blocks are written, the mapping function is applied for each data block number of the file because applying the mapping function is a necessary step when writing the file using the Combination. Second, the mapping function is applied to each data block number for the purpose of retrieving the complete set of data blocks for the file and, at least in some instances, entire files will be read.¹⁵



Madany-Locasto Combination: Visual Aid

¹⁵ To the extent Patent Owner argues Madany does not map each logical block number to a physical block number when a file is compressed, such an argument would fail because Madany's teaches that some files are not compressed. EX1004, 10:57-11:4 ("compression would not be useful and will not be used").

102. As discussed, the **mapping function** corresponds to the following **equation**:

$$pbn = get_from_index(map(lbn, spreading_seed))$$

§VIII.C, *supra*.

103. Moreover, Locasto teaches that the mapping function includes a pseudo-random permutation function based on the well-known cipher algorithms “RC6” and “RC4,” both of which include multiple mathematical operations. EX1005, 4-5, 10. As I noted in §VIII.B, RC6 is described in EX1014.

4. Claim 6

- (i) ***Claim [6]: The method of claim 1, further comprising: selecting a complexity level of the mapping function in order to set a security level and efficiency of the mapping function.***

104. The Madany-Locasto Combination renders obvious Claim [6]. For example, regarding the **mapping function** Locasto explains that a “variable length block cipher is utilized as a pseudo-random permutation (PRP) to map sequential block indices to ostensibly random indices.” EX1005, 10. By “variable length,” Locasto refers to a “variable-key length,” where “larger key lengths imply higher resistance against specific types of attacks” (higher **security level**) and a higher **complexity level** by “requiring more rounds of processing to spread the influence of the key.” EX1005, 1. At the same time, Locasto says, variations that increase the

security level often present a “trade-off [with] performance” (**efficiency**). EX1005, 1, 11 (explaining that “performance” is “measured by the amount of time to read or write a file”). A POSITA would have readily understood that speed of an operation in a computer system is equated with efficiency. For example, in the case of a storage device, performance (e.g., as measured by read/write time) provides an indication of its efficiency since (on the same hardware) faster reads and writes indicate more efficient resource use and slow reads and writes indicate less efficient use.

105. For the CamouflageFS prototype, Locasto **selected** a “28-bit” key and noted that “RC6 was chosen [(**selected**)]...because its construction makes it applicable to small block sizes and RC4 was utilized due to its simplicity.” EX1005, 10. Locasto also indicates the further design choice of **selecting** different key lengths—e.g., “128 bit keys” or “56 bit keys”—and different cipher algorithms. EX1005, 10 (“the variable length block cipher can be built upon any existing block cipher and stream cipher”).

106. Thus, Locasto’s teachings make clear that key length and cipher algorithms are **selectable** variables that define **the complexity level** of the **mapping function** and thus **set the security level and efficiency of the mapping function**.

5. Claim 7

- (i) ***Claim [7]: The method of claim 1, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.***

107. The Madany-Locasto Combination renders obvious Claim [7]. For example, Locasto makes clear that its data-security spreading and encryption techniques “can coexist with normal file operations and does not require any extra work to use regular Ext2 files.” EX1005, 8. Thus, it would have been understood by or obvious to a POSITA that some operations proceed normally—i.e., without spreading or encryption (***if the mapping function is not applied to the data block numbers***). EX1005, 8.

108. The POSITA would also have understood that, when a computer is first used, a file will be assigned to contiguous data blocks. For example, Leffler explains that “[f]or mass-storage devices such as disks, the new filesystem tries to allocate a file’s new blocks on the same cylinder and rotationally well positioned. The distance between rotationally optimal blocks varies greatly; optimal blocks can be consecutive or rotationally delayed, depending on system characteristics. For disks attached to a dedicated I/O processor, two consecutive disk blocks often can be accessed without time lost because of an intervening disk revolution.” EX1006, 200.

109. In this predictable and obvious scenario, a file will occupy the first physical blocks on the disk, typically numbered 0 to N-1, where the file occupies N physical blocks, and the logical block numbers 0 to N-1 will be the same as the physical block numbers (also 0 to N-1). Thus, in this scenario, the data block numbers are addresses of data of the file in the storage device, both numbered 0 to N-1.

6. Claim 8

- (i) *Claim [8]: The method of claim 1, wherein the mapping function prevents copying of data of the file in the storage device.***

110. The Madany-Locasto Combination renders obvious Claim [8]. As background, the '686 patent states that “[a]n unauthorized user will not be able to obtain the data for a file 205 by simply accessing the inode 210 of the file 205 and copying the data blocks 212, because the actual data of the file 205 are stored in the mapped data blocks 215.” EX1001, 4:34-38.

111. As discussed, the Madany-Locasto Combination likewise prevents copying because the file data are stored in mapped data blocks. *Supra*, §VIII.C. For example, per Locasto, the Combination uses “the extra bits of the cryptographic key to spread the file data throughout its address space and use the primary key material to encrypt that data.” EX1005, 3. “By combining this spreading function with random data for unallocated blocks,” the Combination “prevent[s] an attacker from

knowing which blocks to perform a brute force search on.” *Id.* In addition, “[t]o maintain this illusion of a larger file without actually allocating it on disk, [the Combination] return[s] consistently random data on read() operations that are not accompanied by the proper cryptographic key.” *Id.* Therefore, the Combination’s ***mapping function prevents copying of data of the file in the storage device.*** EX1001, 4:34-38. That is, even if the physical bits on the disk (which are encrypted) could somehow be copied, the copied bits cannot be decrypted, and even if they could be decrypted (which they cannot), the decrypted bits cannot be associated with the correct file (because of the spreading). Therefore, a POSITA would have understood that the Combination has prevented copying of the “data of the file” using two layers of protection, encryption and spreading.

7. Claim 9

(i) ***Element [9.pre]: An apparatus for providing data security for use with a file system, the apparatus comprising:***

112. The Madany-Locasto Combination renders obvious Element [9.pre]. *Supra*, §VIII.D.1, Element [1.pre] (***data security for use with a file system***). For example, Madany describes “a method, ***apparatus***, and computer-usable medium.” EX1004, Abstract. Madany further describes “a typical computer system” including a CPU, ROM, RAM, etc., which Madany illustrates in FIG. 9, reproduced below. 16:66-17:30.

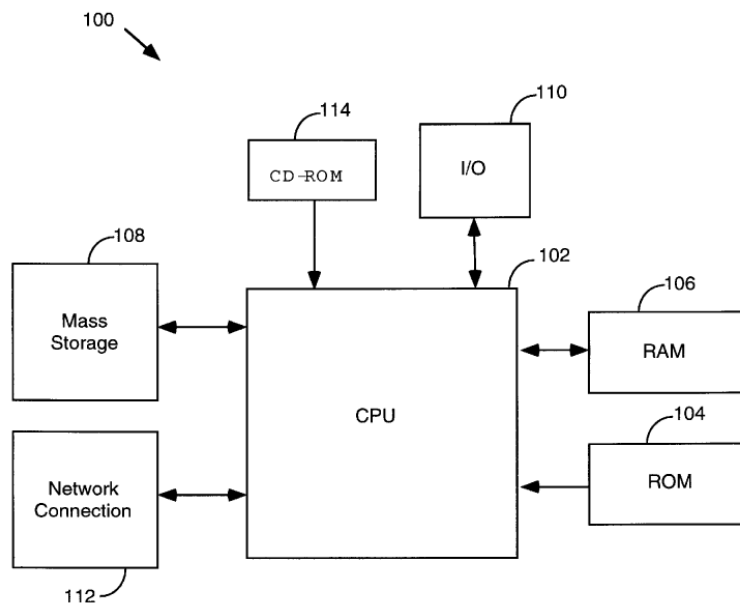


Figure 9

EX1004, FIG. 9

- (ii) ***Element [9.1.1]: a block distribution engine configured to apply a mapping function to data block numbers that are associated with a file,***

113. The Madany-Locasto Combination renders obvious Element [9.1.1]. See *supra*, §VIII.D.1, Element [1.1.1] (discussing ***apply a mapping function to data block numbers***). Additionally, it would have been understood by or obvious to a POSITA that the Madany-Locasto Combination provides ***a block distribution engine***. Indeed, an “engine” in this context merely connotes “the part of a program that implements a special technique.” EX1007, 3. Here, the special technique is ***block distribution***, which the Madany-Locasto Combination provides via Locasto’s mapping function, which spreads (***distributes***) blocks across an address space. *Supra*, §VIII.D.1, Elements [1.1.1]-[1.2.2].

- (iii) ***Element [9.1.2]: wherein the data block numbers are contained in an index node associated with said file, and***

114. See §VIII.D.1, Element [1.1.2], *supra*.

- (iv) ***Element [9.1.3]: obtain mapped data block numbers after applying the mapping function;***

115. See §VIII.D.1, Element [1.2.1], *supra*.

- (v) ***Element [9.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.***

116. See §VIII.D.1, Element [1.2.2], *supra*.

8. Claim 10

- (i) ***Claim [10]: The apparatus of claim 9, wherein the block distribution engine is configured to store a mapping of the data block numbers and the mapped data block numbers.***

117. See §VIII.D.2, Claim [2], *supra*.

9. Claim 13

- (i) ***Claim [13] The apparatus of claim 9, wherein the mapping function is an equation that is applied to each data block number of the file.***

118. See §VIII.D.3, Claim [5], *supra*.

10. Claim 14

- (i) ***Claim [14] The apparatus of claim 9, wherein a complexity level of the mapping function is selected in order to set a security level and efficiency of the mapping function.***

119. See §VIII.D.4, Claim [6], *supra*.

11. Claim 15

- (i) ***Claim [15] The apparatus of claim 9, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.***

120. See §VIII.D.5, Claim [7], *supra*.

12. Claim 16

- (i) ***Claim [16] The apparatus of claim 9, wherein the mapping function prevents copying of data of the file in the storage device.***

121. See §VIII.D.6, Claim [8], *supra*.

13. Claim 17

122. I have been asked to analyze the '686 patent's claim 17 based on interpretations of certain claim terms that use the word "means" as a generic noun followed by a verb that begins the articulation of a functional limitation (e.g., "means for applying..."). I have been informed that such terms require not only the recited function in the claim, but also the corresponding structure disclosed in the '686 patent's specification (or equivalents of such disclosed structure). Below, for each

of these “means” claim terms, I identify the claimed function and the corresponding structure.

- (i) ***Element [17.pre] An apparatus for providing data security for use with a file system, the apparatus comprising:***

123. See §VIII.A.7, Element [9.pre], *supra*.

- (ii) ***Element [17.1.1] means for applying a mapping function to data block numbers that are associated with a file,***

124. The Madany-Locasto Combination renders obvious Element [17.1.1].

125. To start, the structure provided in the '686 patent's specification and described as corresponding to the claimed function of ***applying a mapping function to data block numbers that are associated with a file*** is a “a computer” executing an “operating system” that includes a “block distribution engine” performing a “distribution algorithm,” such as “ $f(x)=5x$,” to block numbers. EX1001, 2:1-14, 2:62-4:3, Figures 1-2.

126. As discussed at Element [1.1.1], the Madany-Locasto Combination describes and/or renders obvious the function of ***applying a mapping function to data block numbers that are associated with a file***. And, as discussed at Element [9.1.1], the Madany-Locasto Combination describes and/or renders obvious the aspect of the corresponding structure that involves a “block distribution engine.” Moreover, like the '686 patent, Madany and Locasto each describe implementations

in the context of a computer operating system. *E.g.*, EX1004, 10:52, 5:38-47; EX1005, 8 (“CamouflageFS is a rather straightforward extension to the standard Ext2 file system for the Linux 2.4.19 kernel”), 11 (“[e]xecution time was measured with the Unix *time*(1) utility”).

127. As for the “distribution algorithm,” Locasto describes “a pseudo-random permutation (PRP) function” based on the well-known cipher algorithms “RC6” and “RC4,” while the ’686 patent describes an example using the simple algebraic expression “ $f(x)=5x$.” *Compare* EX1005, 4-5, 10, with EX1001, 3:50-4:3. Nevertheless, the differences between Locasto’s algorithm and the ’686 patent’s algorithm are insubstantial. In fact, as I described in reference to Claim 2 (§VIII.D.2), Locasto’s algorithm is superior to the algorithm in the ’686 patent. Regardless, what matters is that Locasto’s algorithm (within the Combination) and the ’686 patent’s algorithm work at least substantially the same way—i.e., by converting block numbers derived from an inode into mapped block numbers spread throughout an address space—to achieve at least substantially the same result—i.e., hiding the location of data on the storage disk from anyone unable to repeat the algorithm.

(iii) *Element [17.1.2] wherein the data block numbers are contained in an index node associated with said file; and*

128. See §VIII.D.1, Element [1.1.2], *supra*.

- (iv) ***Element [17.2.1] means for obtaining mapped data block numbers after applying the mapping function,***

129. The Madany-Locasto Combination renders obvious Element [17.2.1]. To start, the structure provided in the '686 patent's specification and described corresponding to the claimed function of ***obtaining mapped data block numbers after applying the mapping function*** is "a computer" executing an "operating system" performing that function. EX1001, 2:1-14, 2:62-4:3, Figures 1-2.

130. As discussed at Element [1.2.1], the Madany-Locasto Combination describes and/or renders obvious the function of ***obtaining mapped data block numbers after applying the mapping function***. And, as discussed at Elements [9.1.pre] and [17.1.1], the Madany-Locasto Combination describes and/or renders obvious the aspect of the corresponding structure that involves a computer with a processor executing an operating system.

- (v) ***Element [17.2.2] wherein the mapped data block numbers are addresses of data of the file in a storage device***

131. See §VIII.D.1, Element [1.2.2], *supra*.

14. Claim 18

- (i) ***Element [18.pre] Article of manufacture, comprising: Element [18.1] a machine-readable medium having stored thereon instructions to:***

132. The Madany-Locasto Combination renders obvious Elements [18.pre]-[18.1]. For example, the Combination includes Madany's description of a "method, apparatus (***[article of manufacture]***), and computer-usable (***[machine-readable]***) medium for compressing data in a file system utilizing the concept of 'holes'." EX1004, Abstract. A POSITA would have understood that Madany's computer-usable medium is ***machine-readable*** at least because the machine (computer) must read the medium for the information on the medium to be useable.

- (ii) ***Element [18.2.1] apply a mapping function to data block numbers that are Element associated with a file,***

133. See [1.1.1], *supra*.

- (iii) ***Element [18.2.2] wherein the data block numbers are contained in an index node associated with said file; and***

134. See [1.1.2], *supra*.

- (iv) ***Element [18.3.1] obtain mapped data block numbers after applying the mapping function,***

135. See [1.2.1], *supra*.

- (v) *Element [18.3.2] wherein the mapped data block numbers are addresses of data of the file in a storage device.*

136. See [1.2.2], *supra*.

IX. GROUND 1B: MADANY AND LOCASTO

A. Madany (EX1004)

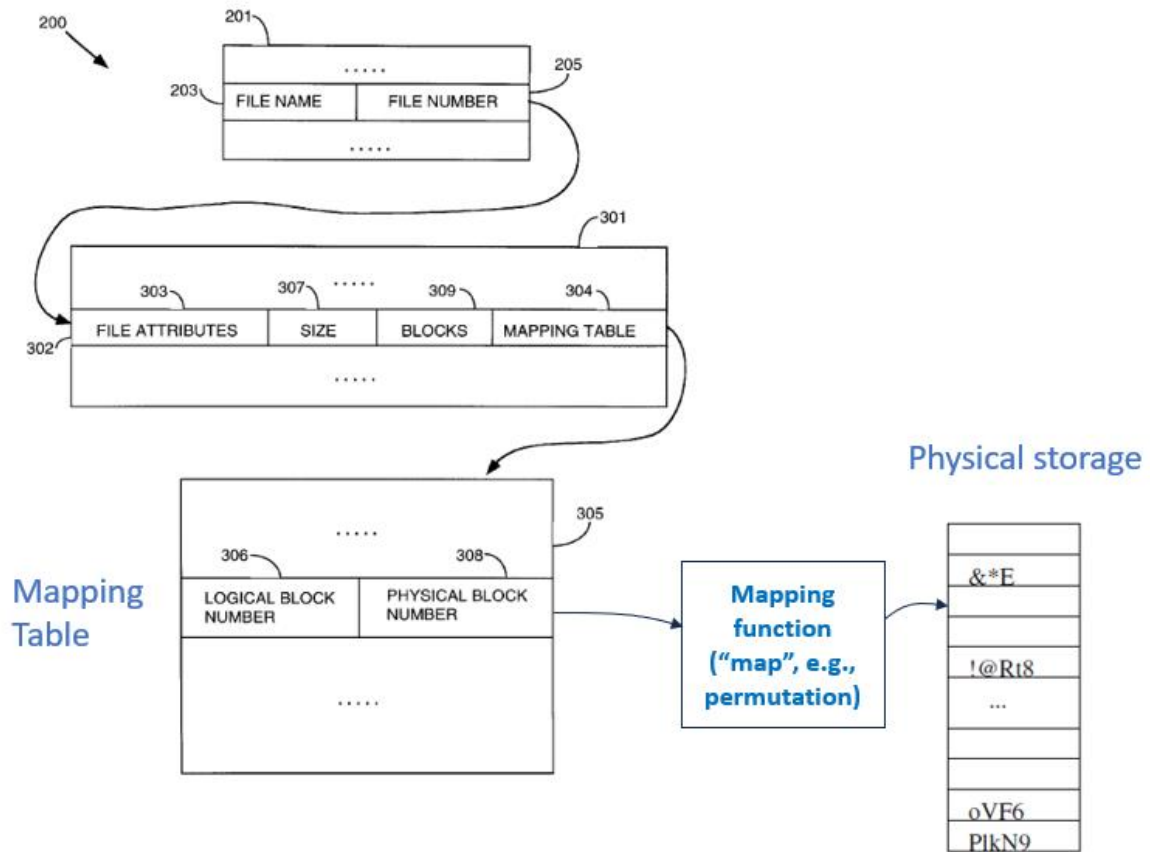
137. See §VIII.A, *supra*.

B. Locasto (EX1005)

138. See §VIII.B, *supra*.

C. The Madany-Locasto Combination

139. As discussed at Section §VIII.C, a POSITA would have understood that Madany's file system architecture presents two apparent options for incorporating Locasto's mapping function, including using Locasto's techniques to map the physical block numbers stored in Madany's mapping table to addresses on the physical storage disk, e.g., as illustrated in the visual aid below and as described here in Ground 1B.



Madany-Locasto Combination: Visual Aid

140. In operation, when a program accesses a file at a given position, the Madany-Locasto Combination of Ground 1B determines the inode (e.g., Inode 301, per Madany), and uses a mapping table pointer in the inode (e.g., mapping table pointer 304, per Madany) to determine the mapping table (e.g., mapping table 305, per Madany). EX1004, 5:38-40, 48-49. (As Leffler indicates, such use of inodes was conventional long before the Critical Date. EX1006, 194, FIG. 7.6.)

141. Per Madany, the Combination maps logical block numbers to physical block numbers. EX1004, 5:58-64. Then, as shown in the visual aid above, the Combination converts the physical block numbers in Madany's mapping table to physical block numbers that are spread, thus providing security. EX1005, 5-6. More specifically, Madany's physical block numbers are spread using a mapping function, e.g., Locasto's "map" function which can be a pseudo-random permutation function and a "spreading seed" (*spreading_seed*). EX1005, 4-5. The function is shown below:

$$\text{spread_pbn_index} = \text{map}(\text{pbn}, \text{spreading_seed})$$
$$\text{spread_pbn} = \text{get_from_index}(\text{spread_pbn_index})$$

EX1005, 5.

142. These two functions can be expressed equivalently as the single (composite) mapping function:

$$\text{spread_pbn} = \text{get_from_index}(\text{map}(\text{pbn}, \text{spreading_seed}))$$

143. Thus, the mapping function "randomly distribute[s] the real file data...throughout a large address space so that an attacker would have to first guess the spreading schedule before he attempts a brute force search on the rest of the key."

EX1005, 4

1. Reasons to Combine and Reasonable Expectation of Success

144. As discussed above at Ground 1A (§VIII.C), a POSITA would have had multiple reasons (taken independently or collectively) to pursue the Madany-Locasto Combination (e.g., the importance of the security techniques taught by Locasto, as I discuss in §§VI.B.2, VIII.C) with a reasonable expectation of success. All of those motivation rationales apply whether Locasto's mapping function is employed to convert Madany's logical block numbers to physical block numbers, as above in Ground 1A, or to map the physical block numbers in Madany's mapping table to addresses on the physical storage disk, as here in Ground 1B. In both cases, Locasto's mapping function provides security by spreading the data. EX1005, 2-5.

D. Claim Element Analysis

1. Claim 1

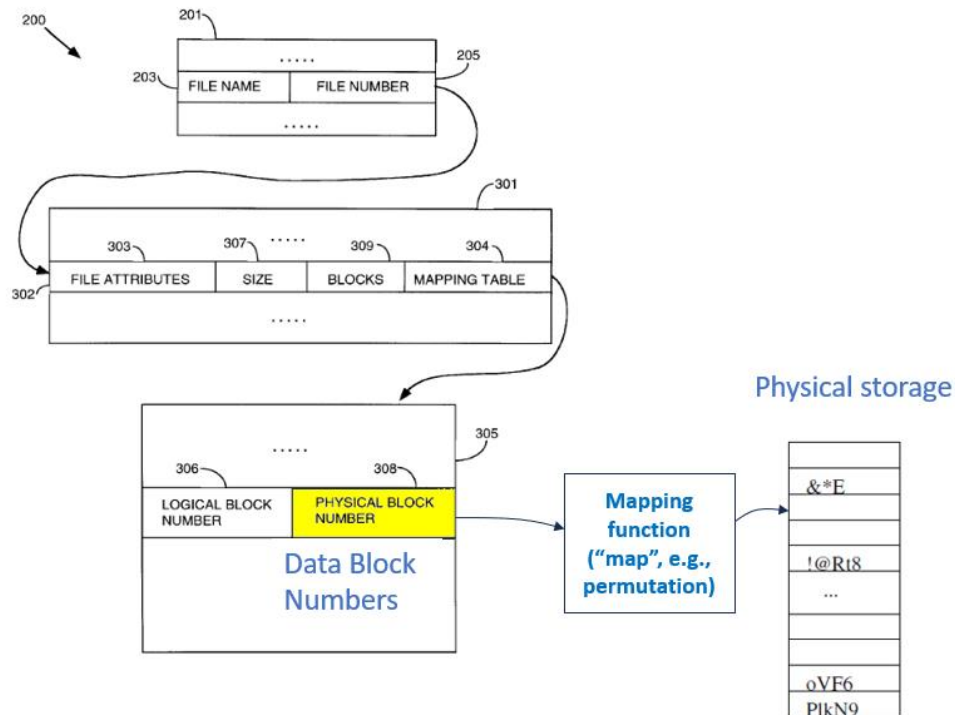
- (i) ***Element [1.pre]: A method of providing data security for use with a file system, the method comprising:***

145. See §VIII.D.1, Element [1.pre], *supra*.

- (ii) ***Element [1.1.1]: applying a mapping function to data block numbers that are associated with a file,***

146. The Madany-Locasto Combination renders obvious Element [1.1.1]. For example, the Madany-Locasto Combination ***applies a mapping function*** to the

physical block numbers (***data block numbers***) in the mapping table to obtain the physical addresses of the data blocks on the physical storage disk, as illustrated in the visual aid below. §IX.C, *supra*.



Madany-Locasto Combination: Visual Aid

147. The Combination further teaches or renders obvious that the physical block numbers (***block numbers***) are ***associated with a file***. For example, Madany explains that the “mapping table 305 contains information on how the logical blocks within a file are mapped to physical blocks on a disk,” and that the “physical block number identifies the location on the disk where the logical block [of the file] is to be stored.” EX1004, 5:52-54, 5:63-64. Thus, a POSITA would have understood

that the physical **block numbers are associated with a file** since they are associated with the file's logical blocks (via the mapping table) and with the file's data stored on physical storage. Madany illustrates the association in Figure 1, which shows information about the file (e.g., file name 203, file number 205) linked to the file's file table 301, which is linked to the file's mapping table 305.

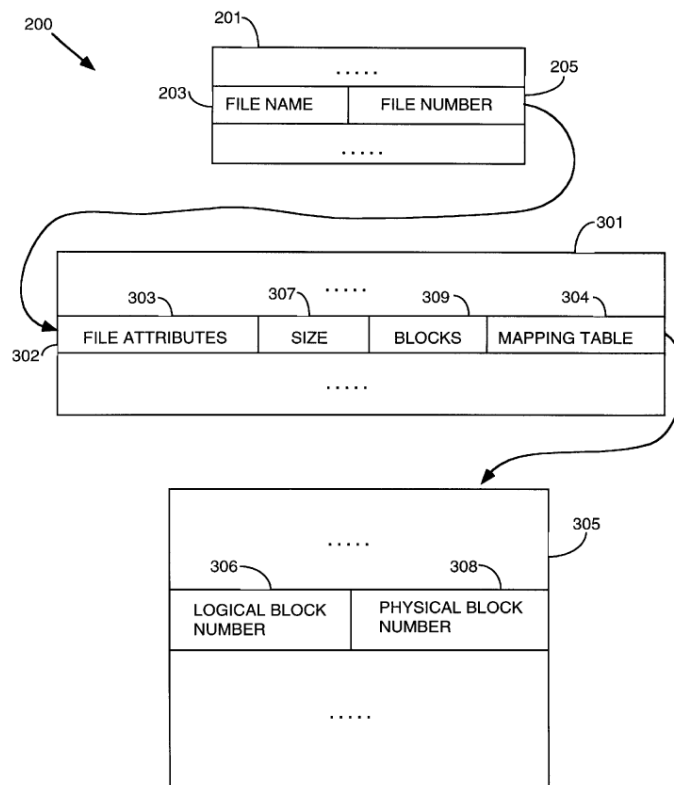


Figure 1

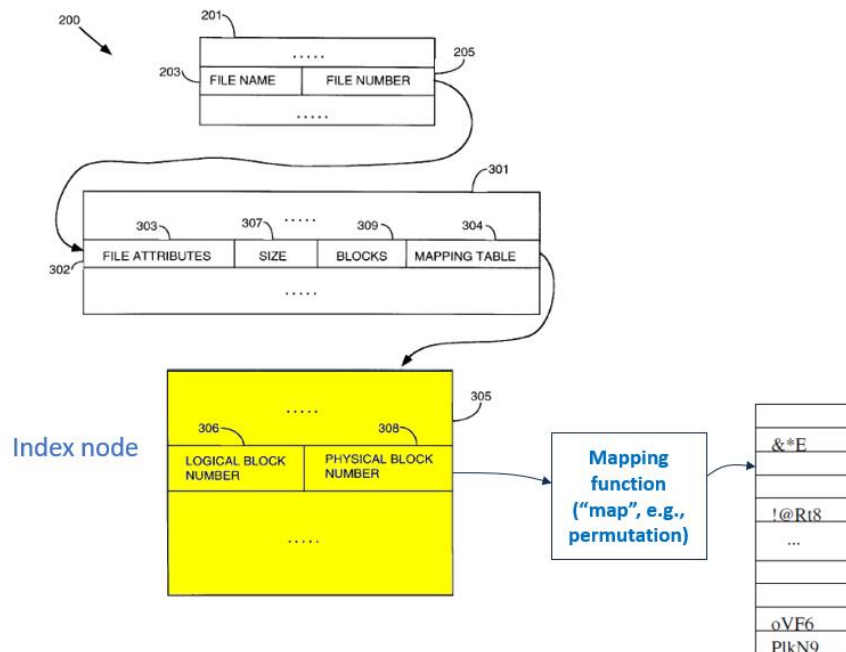
EX1004, Figure 1.

148. As discussed at Sections IX.C, the Madany-Locasto Combination in Ground 1B **applies a mapping function** to physical block numbers (**data block numbers**). Specifically, the Combination's "pseudo-random permutation (PRP)

function takes the” physical block number “and the bits of the spreading seed to return a” spread physical block number, which is the physical address of the data block on the physical storage disk. EX1005, 4-5.

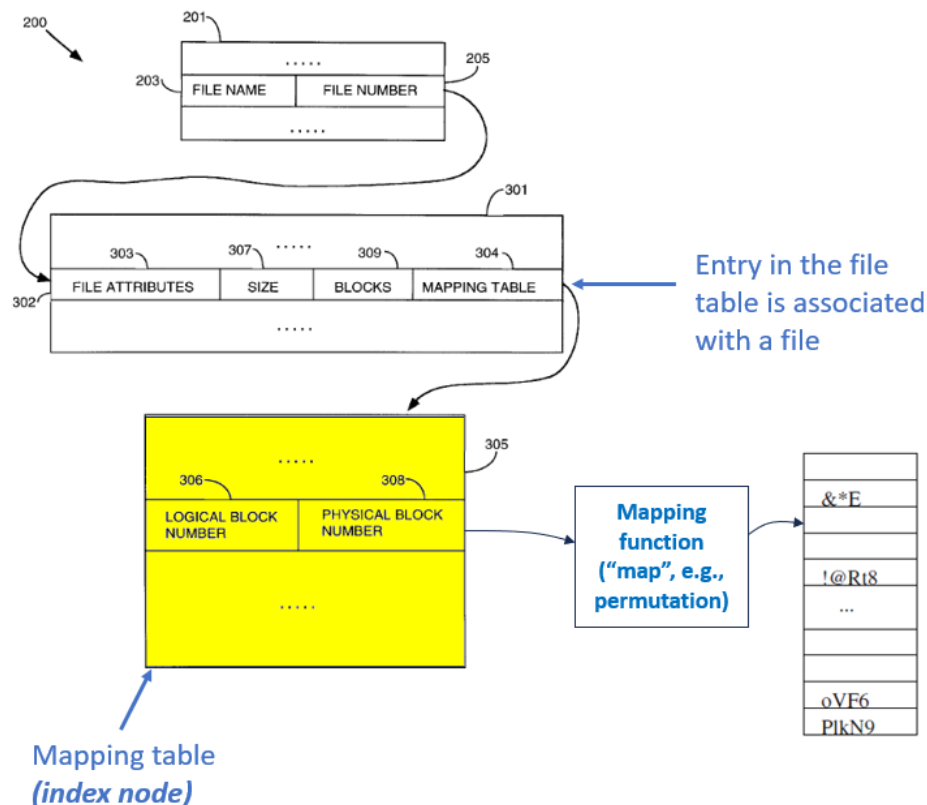
(iii) ***Element [1.1.2]: wherein the data block numbers are contained in an index node associated with said file; and***

149. The Madany-Locasto Combination renders obvious Element [1.1.2]. Madany makes clear that the physical block numbers (***data block numbers***) are ***contained*** in a mapping table (***index node***). For example, Madany explains that the first entry in the mapping table 305 “maps to the physical block number 308 for logical block number 0 of the file.” EX1004, 5:58-60. (I explain that a mapping table serves as an index node in §VIII.D.1, Element [1.1.2].)



Madany-Locasto Combination: Visual Aid

150. Madany further explains that “[e]ach entry 302 in the file table 301 identifies a **file** of the file system and is commonly known as an Inode.” EX1004, 5:35-37. And “[t]here is a mapping table 305 for each entry in the file table 301.” EX1004, 5:31-32. Thus, as would have been understood by or obvious to a POSITA, the mapping table (**index node**) is **associated with a file** at least since the mapping table maps logical block numbers to physical block numbers for a particular file.

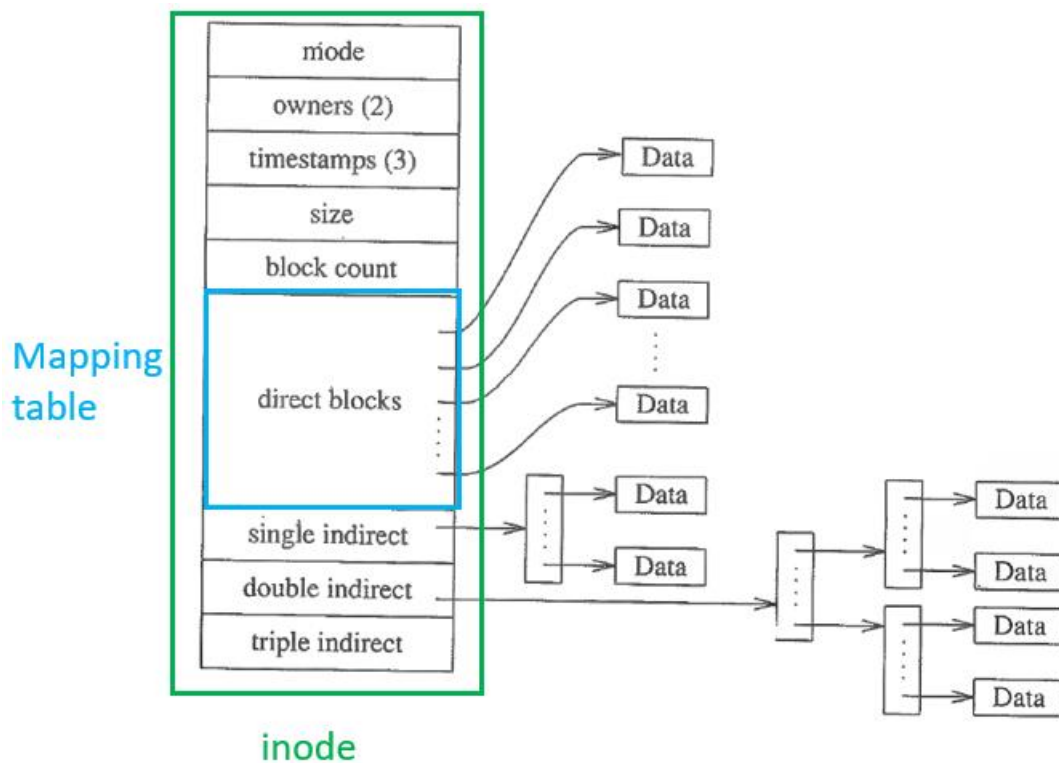


Madany-Locasto Combination: Visual Aid

151. Alternatively, while Madany explains that the Inode 302 contains a pointer 304 to the mapping table 305, it would have nevertheless been understood

by or obvious to a POSITA that the inode can include the mapping table itself (e.g., as implemented in the 4.3BSD Unix operating system), increasing efficiency by eliminating the need to follow a pointer to access the mapping table. (See also my discussion of this topic in §VIII.D.1, Element [1.1.2].)

152. For example, Leffler illustrates an example of an inode containing the mapping table in FIG. 7.6, reproduced below.



EX1006, FIG. 7.6 (annotated)

153. In this example, the inode contains the mapping table. In contrast, when the mapping table is referenced by the inode, instead of contained in the inode, an

extra dereference (following a pointer from one location to another) is required, as shown below in Madany's FIG. 1.

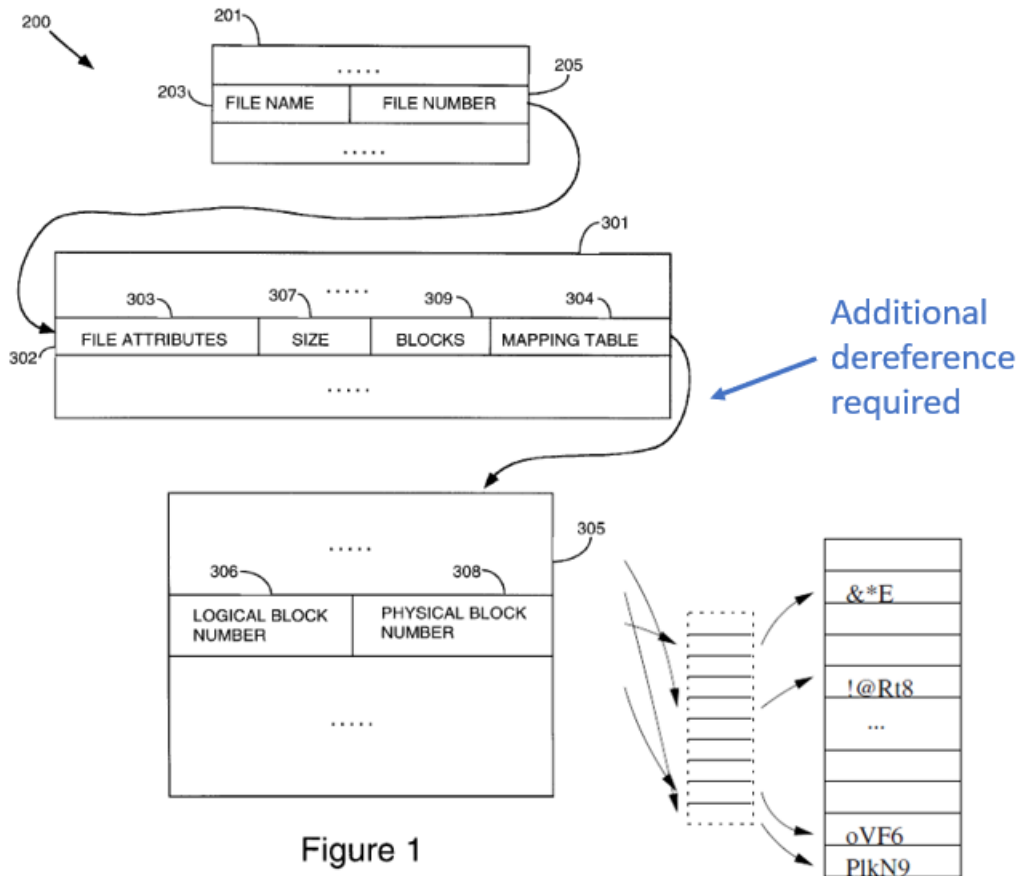


Figure 1

EX1004, FIG. 1 (annotated)

154. This predicable variation is consistent with Madany's explanation that "[i]t should be appreciated that other structures are possible for a file system, and that many file systems may be suitable for use with the present invention." EX1004, 47-51. In this alternative, the inode (**index node**) contains the physical block numbers (**data block numbers**) since the inode contains the mapping table 305.

(iv) ***Element [1.2.1]: obtaining mapped data block numbers after applying the mapping function,***

155. The Madany-Locasto Combination renders obvious Element [1.2.1]. As discussed at Section IX.C, the Combination applies “a pseudo-random permutation (PRP) function [that] takes the [Madany’s physical block number in Madany’s mapping table/inode] and the bits of the spreading seed to return a new index value.” EX1005, 4-5.

spread_pbn_index = map (pbn, spreading_seed)

156. The spread physical “block number is then derived from this new index.” EX1005, 5.

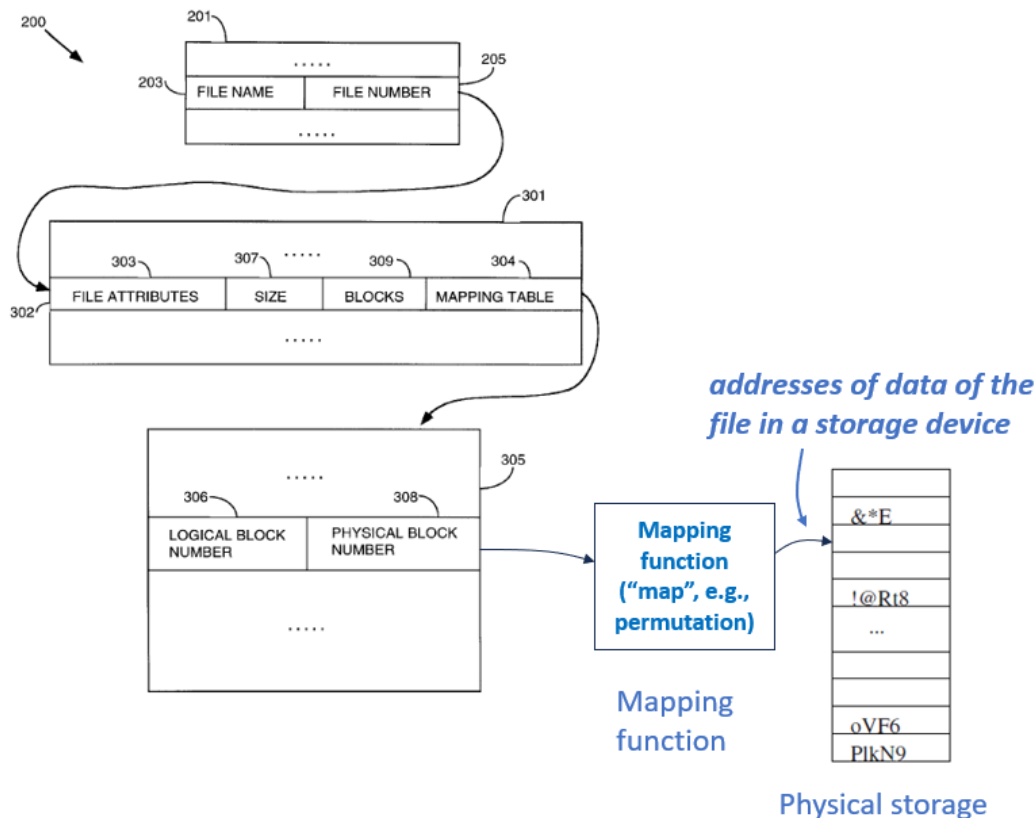
spread_pbn = get_from_index (spread_pbn_index)

157. *Id.* Thus, in the Madany-Locasto Combination, the spread physical block numbers (***mapped data block numbers***) ***are obtained after applying*** the “map” (***mapping***) ***function***.

(v) ***Element [1.2.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.***

158. The Madany-Locasto Combination renders obvious Element [1.2.2]. Specifically, it would have been understood by or obvious to a POSITA that the spread physical ***data block numbers are addresses*** of physical blocks on the disk

(*storage device*) storing *data* contained in *the file*. See *supra*, §IX.C. For example, Locasto explains that a block number determined by its mapping function is “the block on the disk” where the data at a particular file position resides. EX1005, 5.



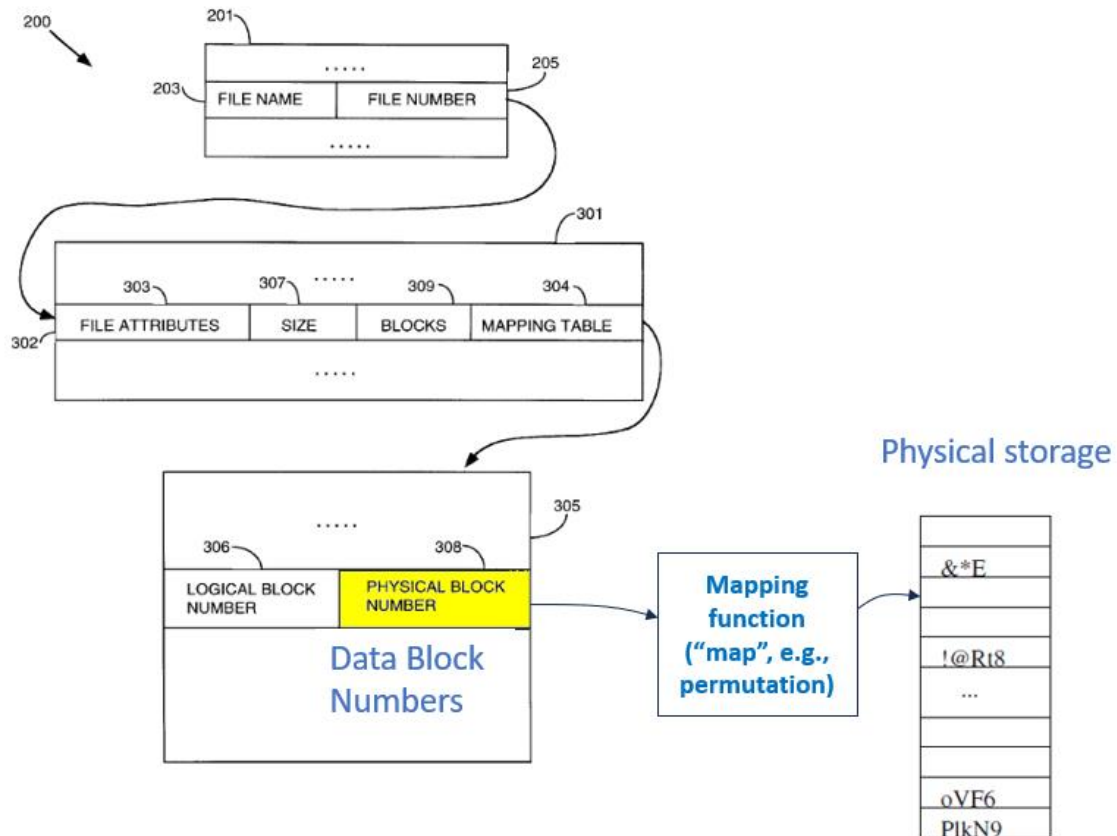
Madany-Locasto Combination: Visual Aid

2. Claim 2

- (i) **Claim [2]:** *The method of claim 1, further comprising: storing a mapping of the data block numbers and the mapped data block numbers.*

159. The Madany-Locasto Combination renders obvious Claim [2]. To start, as discussed at Element [1.1.1], the Madany-Locasto Combination applies a

mapping function to the physical block numbers (***data block numbers***) to obtain the physical addresses of the data blocks on the physical storage disk, as illustrated in the visual aid below. See §IX.C, *supra*.



Madany-Locasto Combination: Visual Aid

160. As discussed at Section IX.C, in the Madany-Locasto Combination, spread (***mapped***) data block numbers are determined from data block numbers using the equation:

$$spread_pbn = get_from_index(map(pbn, spreading_seed))$$

161. Since the Madany-Locasto Combination computes the spread physical block numbers (***mapped data block numbers***) from the physical block numbers (***block numbers***), a POSITA would have understood that the Madany-Locasto Combination ***stores the mapping*** of physical block numbers (***block numbers***) to spread physical block numbers (***mapped block numbers***) at least at the conclusion of the calculation of the spread physical block numbers.

162. For example, consider how a conventional computer system would process the formula given above, $spread_pbn = get_from_index(map(pbn, spreading_seed))$. The computer applies the function *map* to the variables *pbn* and *spreading_seed*, which are stored in memory,¹⁶ producing an interim result. The computer then applies the function *get_from_index* to that interim result, producing a final result that is stored in the variable *spread_pbn*. Thus, the computer stores the mapping of the variable *pbn* (***block number***) to the *spread_pbn* result (***mapped block number***) at least while the computer applies the composite function.

¹⁶ The details of how functions are implemented by conventional computers (including “pushing” variables onto a “stack” where they are available to the function’s code) are beyond the scope of this Declaration and do not alter my analysis.

3. Claim 5

- (i) ***Claim [5]: The method of claim 1, wherein the mapping function is an equation that is applied to each data block number of the file.***

163. The Madany-Locasto Combination renders obvious Claim [5]. For example, as described in §§IX.C, IX.D.1 (Element [1.1.1]), the Madany-Locasto Combination ***applies a mapping function*** to physical block numbers (***data block numbers***) ***of a file***. EX1005, 4-5. The concept of performing this technique at ***each*** logical block number is rudimentary and would have been understood by or obvious to a POSITA, as I discussed in reference to Claim 5 in Ground 1A. See §VIII.D.3. First, when each data block of a file is written, the mapping is performed on the data block number of that data block. For example, Locasto explains that “[t]he purpose of the spreading schedule is to randomly distribute the real file data throughout a large address space.” EX1005, 4. Therefore, since all of the data blocks are written, the mapping function is applied for each data block number of the file. Second, the mapping function is applied to each data block number for the purpose of retrieving the complete set of data blocks for the file and, at least in some instances, entire files

will be read.¹⁷ In fact, Tanenbaum explains that “[g]iven the i-node, it is then possible to find all the blocks of the file.” EX1016, 496.

164. As discussed at §IX.C the **mapping function** corresponds to the following composite **equation**:

spread_pbn = get_from_index(map(pbn, spreading_seed))

EX1005, 5.

165. Moreover, Locasto teaches that the mapping function includes a pseudo-random permutation function based on the well-known cipher algorithms “RC6” and “RC4,” both of which include multiple mathematical operations. EX1005, 4-5, 10. For a discussion of the mathematical operations used by RC6, see EX1014.

4. Claim 6

- (i) ***Claim [6]: The method of claim 1, further comprising: selecting a complexity level of the mapping function in order to set a security level and efficiency of the mapping function.***

166. See §VIII.D.4, Claim [6], *supra*.

¹⁷ To the extent Patent Owner argues Madany does not map each physical block number to a physical block number when a file is compressed, such an argument would fail because Madany’s teaches that some files are not compressed. EX1004, 10:57-11:4 (“compression would not be useful and will not be used”).

5. Claim 7

- (i) ***Claim [7]: The method of claim 1, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.***

167. See §VIII.D.5, Claim [7], *supra*.

6. Claim 8

- (i) ***Claim [8]: The method of claim 1, wherein the mapping function prevents copying of data of the file in the storage device.***

168. See §VIII.D.6, Claim [8], *supra*.

7. Claim 9

- (i) ***Element [9.pre]: An apparatus for providing data security for use with a file system, the apparatus comprising:***

169. The Madany-Locasto Combination renders obvious Element [9.pre]. *Supra*, §IX.D.1, Element [1.pre] (***data security for use with a file system***); EX1004, Abstract (“a method, ***apparatus***, and computer-usable medium”), 16:66-17:30 (describing “a typical computer system” including a CPU, ROM, RAM, etc.). My discussion of typical computer systems §VIII.D.7 (Claim 9 in Ground 1A) is also applicable here (Claim 9 in Ground 1B).

- (ii) ***Element [9.1.1]: a block distribution engine configured to apply a mapping function to data block numbers that are associated with a file,***

170. The Madany-Locasto Combination renders obvious Element [9.1.1]. See *supra*, §IX.D.1, Element [1.1.1] (discussing ***apply a mapping function to data block numbers***). Additionally, it would have been understood by or obvious to a POSITA that the Madany-Locasto Combination provides ***a block distribution engine***. Indeed, an “engine” in this context merely connotes “the part of a program that implements a special technique.” EX1007, 3. Here, the special technique is ***block distribution***, which the Madany-Locasto Combination provides via Locasto’s mapping function, which spreads (***distributes***) blocks across an address space. *Supra*, §IX.D.1, Elements [1.1.1]-[1.2.2].

- (iii) ***Element [9.1.2]: wherein the data block numbers are contained in an index node associated with said file, and***

171. See §IX.D.1, Element [1.1.2], *supra*.

- (iv) ***Element [9.1.3]: obtain mapped data block numbers after applying the mapping function;***

172. See §IX.D.1, Element [1.2.1].

- (v) ***Element [9.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.***

172(a). See §IX.D.1, Element [1.2.2].

8. Claim 10

- (i) *Claim [10]: The apparatus of claim 9, wherein the block distribution engine is configured to store a mapping of the data block numbers and the mapped data block numbers.*

173. See §IX.D.2, Claim [2].

9. Claim 13

- (i) *Claim [13] The apparatus of claim 9, wherein the mapping function is an equation that is applied to each data block number of the file.*

174. See §IX.D.3, Claim [5].

10. Claim 14

- (i) *Claim [14] The apparatus of claim 9, wherein a complexity level of the mapping function is selected in order to set a security level and efficiency of the mapping function.*

175. See §IX.D.4, Claim [6].

11. Claim 15

- (i) *Claim [15] The apparatus of claim 9, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.*

176. See §IX.D.5, Claim [7].

12. Claim 16

- (i) *Claim [16] The apparatus of claim 9, wherein the mapping function prevents copying of data of the file in the storage device.*

177. See §IX.D.6, Claim [8].

13. Claim 17

- (i) *Element [17.pre] An apparatus for providing data security for use with a file system, the apparatus comprising:*

178. See §IX.D.6, Claim [8] and §IX.D.7, Claim [9], *supra*.

- (ii) *Element [17.1.1] means for applying a mapping function to data block numbers that are associated with a file,*

179. See §IX.D.1, Element [1.1.1], *supra* (applying a mapping function to data block numbers that are associated with a file); see also §IX.D.9, Element [17.1.1] (means for).

- (iii) *Element [17.1.2] wherein the data block numbers are contained in an index node associated with said file; and*

180. See §IX.D.1, Element [1.1.2], *supra*.

- (iv) *Element [17.2.1] means for obtaining mapped data block numbers after applying the mapping function,*

181. See §IX.D.1, Element [1.2.1], *supra* (*obtaining mapped data block numbers after applying the mapping function*); see also §VIII.D.13, Element [17.2.1], *supra* (*means for*).

- (v) *Element [17.2.2] wherein the mapped data block numbers are addresses of data of the file in a storage device.*

182. See §IX.D.1, Element [1.2.2], *supra*.

14. Claim 18

- (i) *Element [18.pre] Article of manufacture, comprising: Element [18.1] a machine-readable medium having stored thereon instructions to:*

183. The Madany-Locasto Combination renders obvious Elements [18.pre]-[18.1]. For example, the Combination includes Madany's description of a "method, apparatus ([*article of manufacture*]), and computer-usable [*machine-readable*] medium for compressing data in a file system utilizing the concept of 'holes'." EX1004, Abstract. As I discussed in §VIII.D.14 (Ground 1A), a POSITA would have understood that Madany's computer-usable medium is *machine-readable* at least because the machine (computer) must read the medium for the information on the medium to be usable.

- (ii) *Element [18.2.1] apply a mapping function to data block numbers that are associated with a file,*

184. See § IX.D.1, Element [1.1.1].

- (iii) *Element [18.2.2] wherein the data block numbers are contained in an index node associated with said file; and*

185. See § IX.D.1, Element [1.1.2].

- (iv) *Element [18.3.1] obtain mapped data block numbers after applying the mapping function,*

186. See § IX.D.1, Element [1.2.1].

- (v) *Element [18.3.2] wherein the mapped data block numbers are addresses of data of the file in a storage device.*

187. See § IX.D.1, Element [1.2.2].

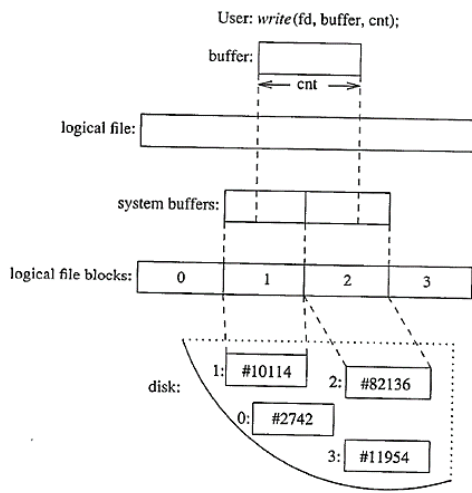
X. GROUND 2: MADANY-LOCASTO-LEFFLER

A. Leffler (EX1006)

188. Leffler is a textbook entitled “*The Design and Implementation of the 4.3BSD UNIX Operating System.*” Leffler provides a “full-length description of the design and implementation of the research versions of the UNIX system,” with “[m]ost detail [] given about 4.3BSD.” EX1006, vii. Leffler describes various functional elements of the operating system including “Processes,” the “I/O System,” “Interprocess Communication,” and “System Operation.” EX1006, ix-x.

189. In its discussion of the I/O system, Leffler explains that a disk is an example of a “block device,” and the “task of the block-device interface is to convert from the user abstraction of a disk as an array of bytes to the structure imposed by the underlying physical medium.” EX1006, 171. A typical disk has fixed-sized sectors, and the disk can only read and write in “multiples of sectors,” even when “the user may wish to write a single byte to the file.” EX1006, 191, 192. The “system must arrange to read in the sector containing the byte to be modified, to replace the affected byte, and to write the sector back to the disk.” EX1006, 192.

190. Leffler’s Figure 7.5 shows the “work required to access the filesystem on a disk.” EX1006, 192. The user creates an array of bytes (“buffer”) and issues a “write” command that indicates the number of bytes (“cnt” for “count”) from the buffer to be written. EX1006, 192, Figure 7.5. The user also provides a file descriptor (“fd,”) indicating the file to which the “write” command applies. *Id.*



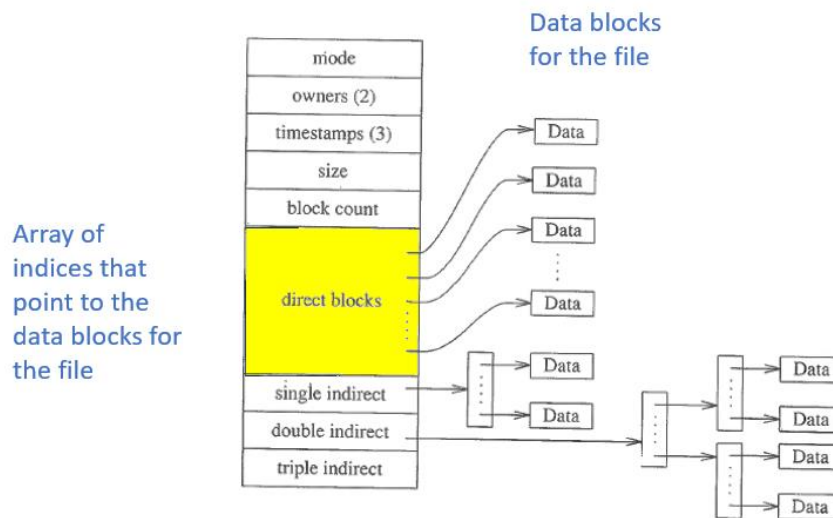
EX1006, Figure 7.5

191. To service the “write” operation, “the system breaks the user’s request into a set of operations to be done on each logical block of the file,” where “[l]ogical blocks describe block-sized pieces of a file.” EX1006, 192. “The system calculates the logical blocks by dividing the array of bytes into filesystem-sized pieces,” e.g., “if a filesystem’s block size is 8192 bytes, then logical block 0 would contain bytes 0 to 8191, logical block 1 would contain bytes 8192 to 16,383, and so on.” *Id.*

192. “The data in each logical block is stored in a physical block on the disk,” which “is the location on the disk to which the system maps a logical block.” EX1006, 193. “Although the contents of a logical block are contiguous on disk, the logical blocks of the file need not be laid out contiguously.” EX1006, 193. In the example of Figure 7.5, logical block 0 maps to physical block “#2742,” logical block 1 maps to physical block “#10114,” logical block #2 maps to physical block “#82136,” and logical block #3 maps to physical block “#11954.” EX1006, 192.

193. To enable mapping of logical blocks to physical blocks, “[e]very file has an associated inode that contains information describing access permissions, ownership of the file, timestamps marking last modification and access times for the file, and an array of indices that point to the data blocks for the file.” EX1006, 189. As described above, the system calculates the logical block number for the file operation and uses that logical block number as an index into the “array of indices

that point to the data blocks for the file” to determine the physical block. EX1006, 189, 192. Note that Leffler’s teaching is similar to Locasto’s teaching of index calculation ($index = pos/blocksize$). EX1005, 5.



EX1006, Figure 7.6 (annotated)

194. When a program reads or writes data from disk, the operating system performs the operations illustrated in Figure 7.12, reproduced below.

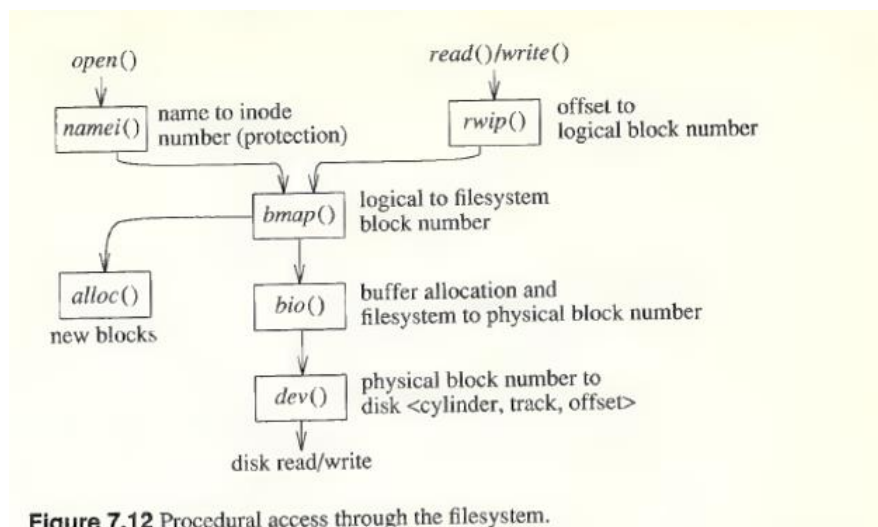


Figure 7.12 Procedural access through the filesystem.

EX1006, Figure 7.12.

195. First, a program opens the relevant file, as illustrated in the top left of Figure 7.12. Each “*open*” system call allocates a new entry in a “file table” that “points to a table containing filesystem specific information.” EX1006, 173-174. “For files, the file table points to the inode representing the open file.” EX1006, 205. Upon a successful *open* call, the system provides a “small integer file descriptor” (“fd”) that is an index into the file table. *Id.* Thus, a file’s file descriptor indicates the file’s inode. Accordingly, once a file is open, subsequent system calls that operate on the file include a “file descriptor as an argument to specify the file” and its inode. EX1006, 173. Leffler provides an example in which the file descriptor “fd” is included in the “*write*” system call of Figure 7.5, *write(fd, buffer, cnt)*. (The parameter *buffer* is a pointer to the buffer (area of memory) that contains the bytes to be written and *cnt* is the number of bytes to be written from that buffer.) Next, “[h]aving opened a file,” reads or writes are performed on it by calling “*rwip*,” which converts “the read or write into one or more reads or writes of logical file blocks.” EX1006, 206. The logical block request is passed to “*bmap*,” which converts “a logical block number to a physical block number” using the inode. EX1006, 207. The system uses the “*bio*” and “*dev*” calls to complete the *read* or *write* request on the physical disk. EX1006, 207.

B. Claim Element Analysis

1. Claims 1, 9, 17-18

196. The following analysis applies Leffler's supplemental teachings to provide an alternative basis for satisfying specific elements of independent claims 1, 9, 17-18. My analysis of Ground 2 incorporates my analysis of Grounds 1A-1B for all other elements.

- (i) *Element [1.1.2]: wherein the data block numbers are contained in an index node associated with said file; and*
- (ii) *Element [9.1.2]: wherein the data block numbers are contained in an index node associated with said file, and*
- (iii) *Element [17.1.2] wherein the data block numbers are contained in an index node associated with said file; and*
- (iv) *Element [18.2.2] wherein the data block numbers are contained in an index node associated with said file; and*

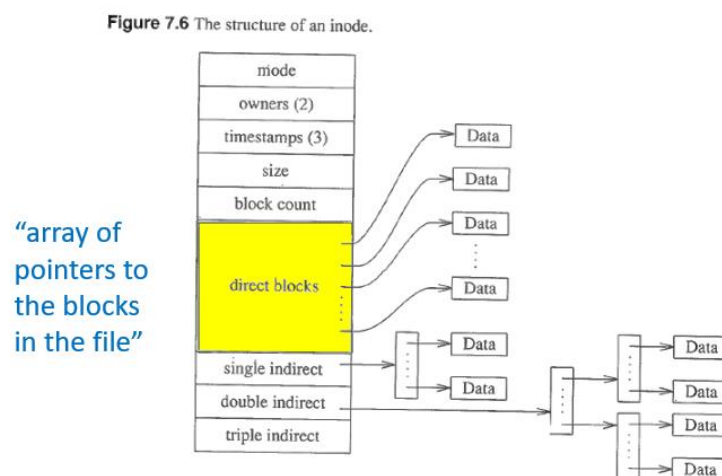
197. The Madany-Locasto-Leffler Combination renders obvious Elements [1.1.2], [9.1.2], [17.1.2], and [18.2.2].

198. In each of Grounds 1A-1B, the analysis points to **data block numbers** (e.g., logical block numbers or physical block numbers) **contained** in Madany's mapping table. *Supra*, §VIII.D.1, Element [1.1.2], § IX.D.1, Element [1.1.2].

199. Moreover, it would have been understood by or obvious to a POSITA in view of Leffler's file system teachings that Madany's mapping table is an integral

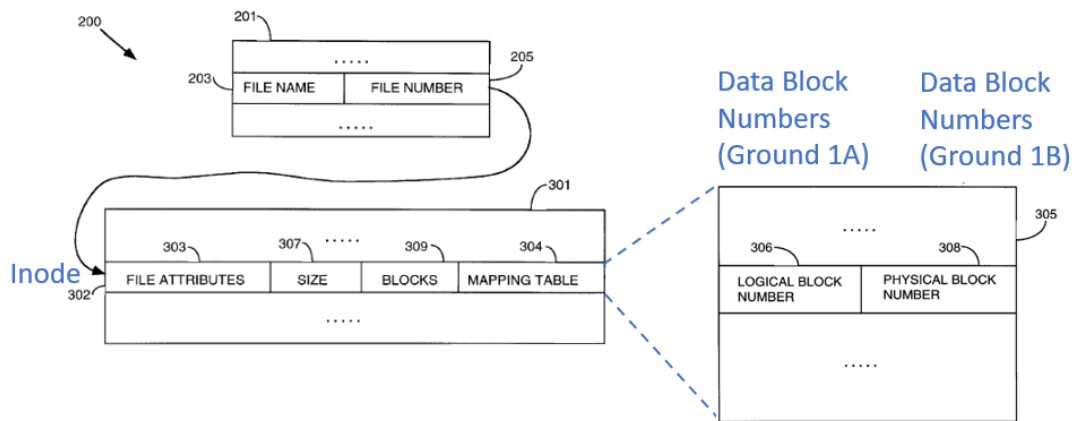
part of the file's inode, meaning the **data block numbers** are **contained** in the **index node** (the inode including the mapping table). *Supra*, §X.A.

200. For example, as discussed, Leffler's "Figure 7.6 shows that the inode contains an array of pointers to the blocks in the file," allowing the system to "convert from a logical block number to a physical sector number by indexing into the array using the logical block number." EX1006, 194; *supra*, §X.A. Since Madany explains that its "mapping table 305 contains information on how the logical blocks within a file are mapped to physical blocks on a disk" (EX1004, 5:52-54), a POSITA would have understood that Leffler's "array of pointers to blocks in the file" (EX1006, 194) is an example of Madany's mapping table 305 since both Madany's mapping table 305 and Leffler's direct blocks map indices to block numbers.



EX1006, Figure 7.6 (annotated).

201. A POSITA, therefore, would have understood and/or found obvious the notion of integrating Madany's mapping table in the inode of the Madany-Locasto-Leffler Combination, as illustrated below. Accordingly, in the Combination the **data block numbers** are **contained** in an Inode 302 (**index node**).



Madany-Locasto-Leffler Combination: Visual Aid

202. A POSITA would have been motivated to incorporate this rudimentary teaching of Leffler with the Madany-Locasto Combination for multiple reasons. To start, Madany explicitly references Leffler's description concerning the structure of an inode, writing that the "inode 302, or row of the file table 301, contains information about the given file. Inodes are described in 'The Design and Implementation of the 4.3BSD UNIX Operating System' [Leffler], Chapter 7." EX1004, 5:40-44. Since Leffler includes the mapping of logical to physical blocks in the inode itself, a POSITA would have been motivated to apply that technique in the context of the Madany-Locasto Combination. EX1006, 194, FIG. 7.6. Further,

including the mapping in the inode itself increases system efficiency by eliminating the need to follow a pointer before accessing the mapping table for blocks, as I discussed in §IX.D.1, Element [1.1.2]. EX1006, 194. Additionally, a POSITA would have understood that there are effectively only two viable options: include the mapping table in the inode or include a reference to the mapping table in the inode. Both approaches would have been obvious to a POSITA. For example, a POSITA would have known that it was common in programming both to nest one data structure within another data structure (e.g., an array contained within a larger structure, e.g., a “struct” in the C programming language) and to have one data structure point to another data structure (e.g., an array pointed to by or within a structure). The former approach is simpler and typically more efficient; the latter approach is more flexible since the size of the second data structure (the one pointed to) need not be fixed.

2. Claim 3

- (i) *Element [3.1]: The method of claim 1, wherein a read operation for the file comprises: obtaining the data block numbers of the file;***
- (ii) *Element [3.2]: obtaining the mapped data block numbers of the file based on the mapping function that is applied to the data block numbers; and***
- (iii) *Element [3.3]: reading the data in the mapped data block numbers in the storage device.***

203. The Madany-Locasto-Leffler Combination renders obvious Elements [3.1]-[3.3]. As discussed at Section X.A, Leffler provides a full-length discussion of the 4.3BSD UNIX operating system, including a breakdown of the internal file system and a description of the calls and operations involved in reading and writing to the storage disk. *See generally* EX1006, 169-223. A POSITA contemplating the Madany-Locasto Combination of Grounds 1A-1B would have been motivated to incorporate Leffler's conventional file system teachings regarding disk reads and writes for multiple reasons. First, as discussed above at independent claims 1, 9, and 17-18, Madany explicitly references Leffler's Chapter 7 on the 4.3BSD file system, which would have prompted the POSITA to consult Leffler for supplemental implementation details on relevant subjects like the flow of calls and procedures for reading and writing to a storage disk. EX1004, 5:40-44. Second, even without Madany's express reference to Leffler, the POSITA would have been motivated to pursue Leffler's supplemental file system teachings as a well-known solution to the

rudimentary design problem of “convert[ing] from the user abstraction of a file as an array of bytes to the structure imposed by the underlying physical medium [of the storage disk].” EX1006, 191.

204. Additionally, I note that not every file is compressed in Madany, and that it is “desirable for both compressed and expanded files to coexist” in the same system. EX1004, 3:37-43; *see also* 4:32-36 (similar), 11:1-4 (“compression would not be useful and will not be used”). Therefore, even if Madany’s sole focus was to create a system that compressed every file (which is not true), it would still have been obvious to rely on Madany’s description of, and explicit reference to, the well-known 4.3BSD UNIX file system taught by Leffler. Lastly, although it is not required for the reasons discussed, my analysis of claims 3 and 4 is applicable to implementations that employ Madany’s compression techniques. EX1004, 7:8-65 (“mapping table 305”). In short, Madany utilizes an inode and mapping table like the one in Leffler’s 4.3BSD Unix file system, even with compression, meaning the rudimentary steps of obtaining logical/physical block numbers are still performed during reading and writing operations. Of course, if modified to operate on a file system compressed using the techniques of Madany, the Madany-Locasto-Leffler Combination would compress data when writing (to block determined using the techniques of the Madany-Locasto-Leffler Combination described above) and decompress data when reading.

Applying Leffler's Teachings to the Ground 1A Madany-Locasto Combination

205. When reading or writing, the Madany-Locasto-Leffler Combination **obtains** the logical block number (**data block number**, per Ground 1A, Element [1.1.1]) of the data to be **read**/written. See EX1006, 192 (“The system calculates the logical blocks...”); *supra*, §X.A; EX1004, 2:47-51 (“a table containing N logical to physical block mappings”), 5:29-7:65 (Madany referencing Leffler and discussing “mapping table 305”).

206. Next, as discussed, the Combination **applies** Locasto’s **mapping function** to the logical block number to **obtain** a physical block number (**mapped data block number**) that is the address on the disk where the logical block data is stored. *Supra*, §VIII.D.1 (Ground 1A, Elements [1.1.1], [1.2.1], [1.2.2]). The Combination then completes the **read** or write request on the physical disk using the physical block number (**reading the data in the mapped data block numbers**). EX1006, 206-207 (Figure 7.12); §X.A.

Applying Leffler's Teachings to the Ground 1B Madany-Locasto Combination

207. When reading or writing, the Madany-Locasto-Leffler Combination obtains the logical block number of the data to be **read**/written and uses that logical block number, via the mapping table, to **obtain** the corresponding physical block number (**data block number**, per Ground 1B, Element [1.1.1]). See EX1006, 192,

194 (“The system can convert from a logical block number to a physical [block] number...”), 195 (“logical-to-physical block mapping”); *supra*, §X.A; EX1004, 2:47-51 (“a table containing N logical to physical block mappings”), 5:29-7:65 (Madany referencing Leffler and discussing “mapping table 305”).

208. Next, as discussed, the Combination ***applies*** Locasto’s ***mapping function*** to the physical block number to determine a spread physical block number (***mapped data block number***) that is the address on the disk where the logical block data is stored. *Supra*, §IX.D.1 (Ground 1B), Elements [1.1.1], [1.2.1], [1.2.2]). The Combination then completes the ***read*** or write request on the physical disk using the spread physical block number (***reading the data in the mapped data block numbers***). EX1006, 206-207 (Figure 7.12); §X.A.

3. Claim 4

- (i) ***Element [4.1]: The method of claim 1, wherein a write operation for the file comprises: allocating data block numbers of free data blocks in the storage device;***

209. The Madany-Locasto-Leffler Combination renders obvious Element [4.1]. Leffler explains that the operating system maintains a record of ***free blocks***. EX1006, 195, 197; *see also* §X.B.2, Claim 3 (motivation-to-combine Leffler’s conventional file system teachings with Madano-Locasto). For example, a “free list” contains “a list of all the ***free blocks*** in the filesystem.” EX1006, 195. Alternatively,

the system can maintain “a bit map describing available [~~(free)~~] **blocks** in the cylinder group.” EX1006, 197.

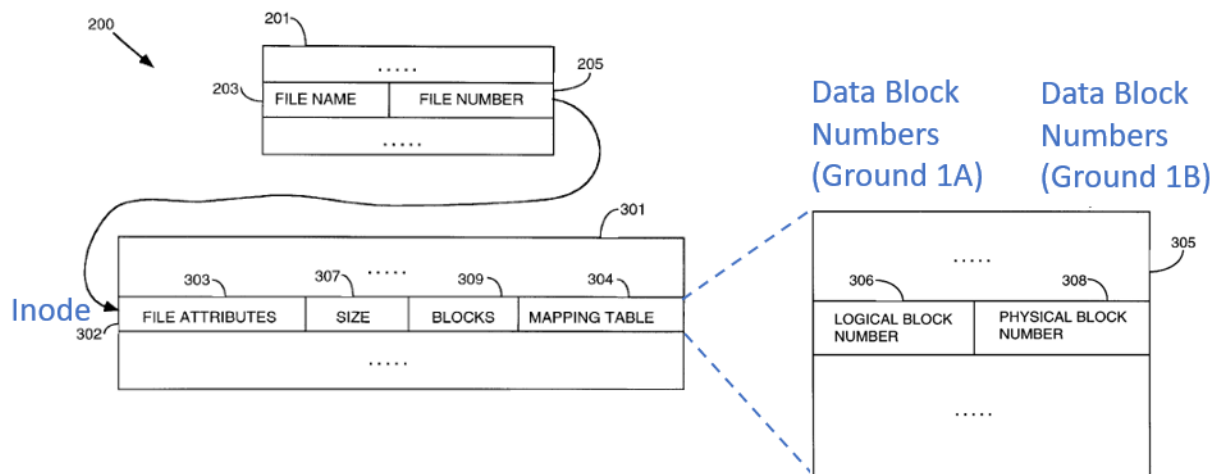
210. As files grow, the “filesystem [] handles the **allocation** of new blocks to files.” EX1006, 193. Leffler continues, “[e]ach time a process does a **write** system call, the system checks to see whether the size of the file has increased,” and if “the file needs to be expanded, the request is rounded up to the next fragment size and only that much space is **allocated**.” EX1006, 199. Since a single file system block can be divided into one or more fragments, a POSITA would have understood and/or found obvious that a block can be exactly one fragment. EX1006, 198 (“The developers obtained this efficiency by allowing the division of a single filesystem block into one or more *fragments*,” emphasis in the original). When a file is **allocated** a new block, a POSITA would have understood that the block would be assigned from among the ***free data blocks on the storage device***. In practice, the operating system identifies a free block on the storage device from the free list (list of free blocks), removes the block from the free list, and assigns the block to the file.

211. Leffler further teaches that an inode contains a file’s “logical-to-physical block mapping.” EX1006, 194-195. Therefore, a POSITA would have understood and/or found obvious that since the inode stores the block numbers of the physical blocks allocated during the ***write operation, data block numbers of free***

data blocks are also **allocated**. For example, since Leffler teaches that the “free list” contains “a list of all the free blocks in the file system,” a POSITA would have understood and/or found obvious that when a block is removed from the free list, its corresponding data block number is removed from the free list (“list of all the free blocks in the filesystem”) and added to the inode. EX1006, 194-195.

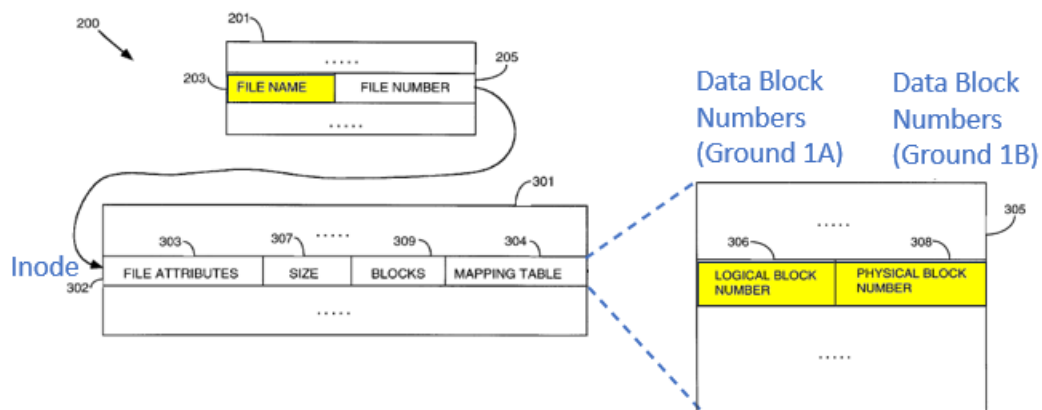
(ii) ***Element [4.2]: linking the data block numbers to a file name of the file;***

212. The Madany-Locasto-Leffler Combination renders obvious Element [4.2]. For example, as explained above in reference to Element [1.1.2], in the Madany-Locasto-Leffler Combination, the inode contains the **data block numbers of a file**, as illustrated in the visual aid below. *Supra*, §X.A.



Madany-Locasto-Leffler Combination: Visual Aid

213. Further, per Madany, the “file system has a hierarchy of directory structures 201” and “[w]ithin the directory 201 are entries of file names within the file system and corresponding file numbers that allow files to be indexed to an entry in the file table.” EX1004, 5:30-35. “Each entry 302 in the file table 301 identifies a file of the file system and is commonly known as an Inode.” EX1004, 5:35-37. Since the directory 201 contains a file name that is linked to the inode, and the inode contains the data block numbers, a POSITA would have understood and/or found obvious that ***the data block numbers are linked to a file name of the file.*** Further, since the data block numbers are added to the inode during the write operation, the write operation comprises linking the data block numbers to a file name of the file.



Madany-Locasto-Leffler Combination: Visual Aid

214. For example, the visual aid above shows the file name 203 linked to the inode, which contains the data block numbers (in inode’s the mapping table).

- (iii) **Element [4.3]: applying the mapping function to the data block numbers in order to obtain the mapped data block numbers; and**

215. The Madany-Locasto-Leffler Combination renders obvious Element [4.3]. See §X.B.2, Element [3.2] (describing the **application of the mapping function** in both read and write operations).

- (iv) **Element [4.4]: writing the data of the file into the mapped data block numbers in the storage device.**

216. The Madany-Locasto-Leffler Combination renders obvious Element [4.4]. For example, per Leffler, the Combination uses the “*bio*” and “*dev*” calls to complete the read or **write** request on the physical disk using the physical (**mapped**) **block number**. EX1006, 207; §X.A, §X.B.2, Claim 3.

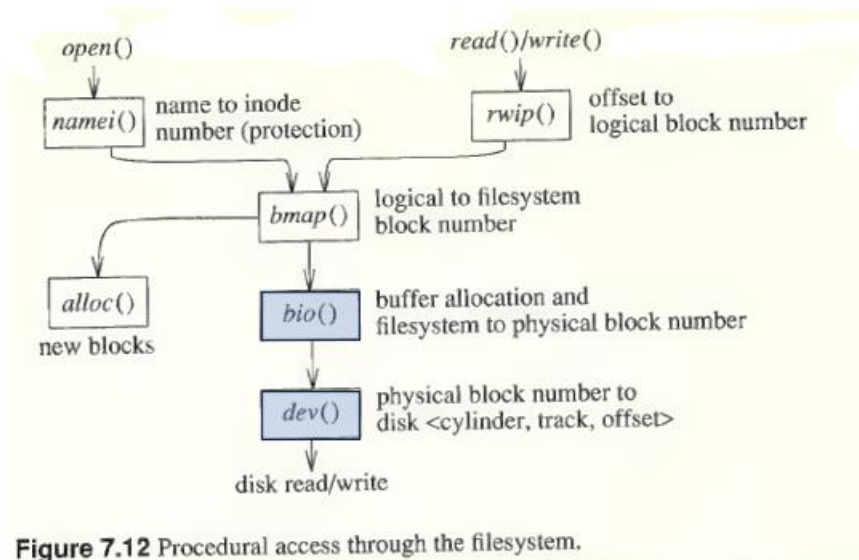


Figure 7.12 Procedural access through the filesystem.

EX1006, FIG. 7.12 (annotated).

4. Claim 11

- (i) *Element [11.1] The apparatus of claim 9, wherein for a read operation for the file, the block distribution engine obtains the data block numbers of the file from a file system, and*

217. See §X.B.2, Element [3.1], *supra*.

- (ii) *Element [11.2] obtains the mapped data block numbers of the file based on the mapping function that is applied to the data block numbers;*

218. See §X.B.2, Element [3.2], *supra*.

- (iii) *Element [11.3] and wherein a block device driver is configured to read the data in the mapped data block numbers in the storage device.*

219. See §X.B.2, Element [3.3], *supra*.

5. Claim 12

- (i) *Element [12.1] The apparatus of claim 9, wherein for a write operation for the file, the block distribution engine is configured to receive allocated data block numbers of free data blocks in the storage device from a file system,*

220. See §X.B.3, Element [4.1], *supra* (*receive allocated data block numbers of free data blocks in the storage device from a file system*); see also §VIII.D.13, Element [9.1.1] (*block distribution engine*).

- (ii) *Element [12.2] link the data block numbers to a file name of the file, and*

221. See §X.B.3, Element [4.2], *supra*.

- (iii) *Element [12.3] apply the mapping function to the data block numbers in order to obtain the mapped data block numbers;*

222. See §X.B.3, Element [4.3], *supra*.

- (iv) *Element [12.4] and wherein a block device driver is configured to write the data of the file into the mapped data block numbers in the storage device.*

223. See §X.B.3, Element [4.4], *supra*.

XI. GROUND 3: LEFFLER AND LOCASTO

A. Leffler (EX1006)

224. See §X.A, *supra*.

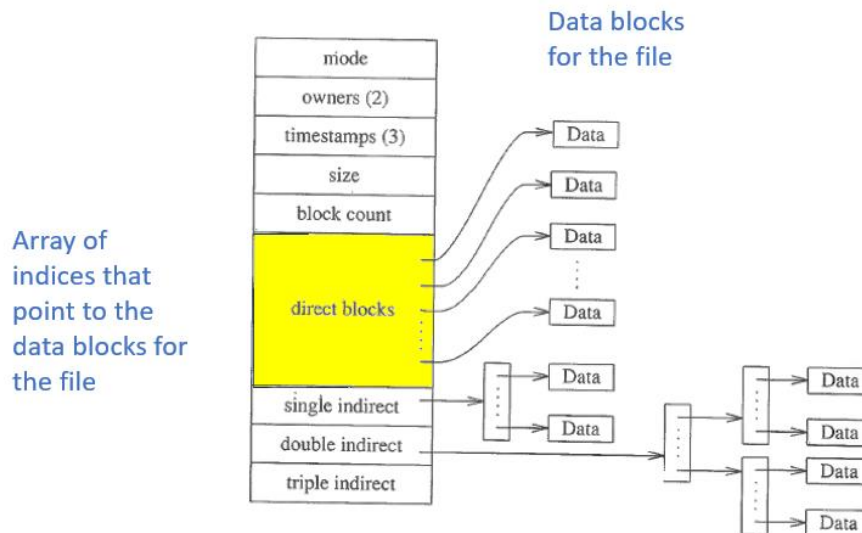
B. Locasto (EX1005)

225. See §VIII.B, *supra*.

C. The Leffler-Locasto Combination

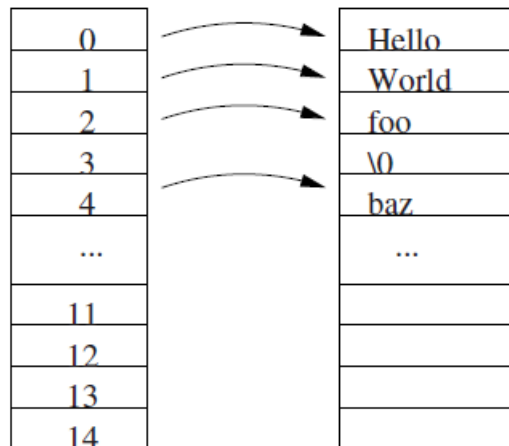
226. As discussed, Leffler is a textbook that provides a “full-length description of the design and implementation of the research versions of the UNIX system,” including a discussion of the I/O system. EX1006, ix-x. Leffler further

explains that, to enable mapping of logical blocks to physical blocks, “[e]very file has an associated inode that contains information describing access permissions, ownership of the file, timestamps marking last modification and access times for the file, and an array of indices that point to the data blocks for the file.” EX1006, 189. Leffler explains that the system calculates the logical block number for the file operation and uses that logical block number as an index into the “array of indices that point to the data blocks for the file” to determine the physical block. EX1006, 189, 192.



Leffler-Locasto-Combination: Visual Aid

227. Similar to Leffler, Locasto describes a file system in which “indices” are mapped to block numbers, as illustrated in FIG. 2 (reproduced below). EX1005, 3-4; §VIII.B, *supra*.



EX1005, FIG. 2.

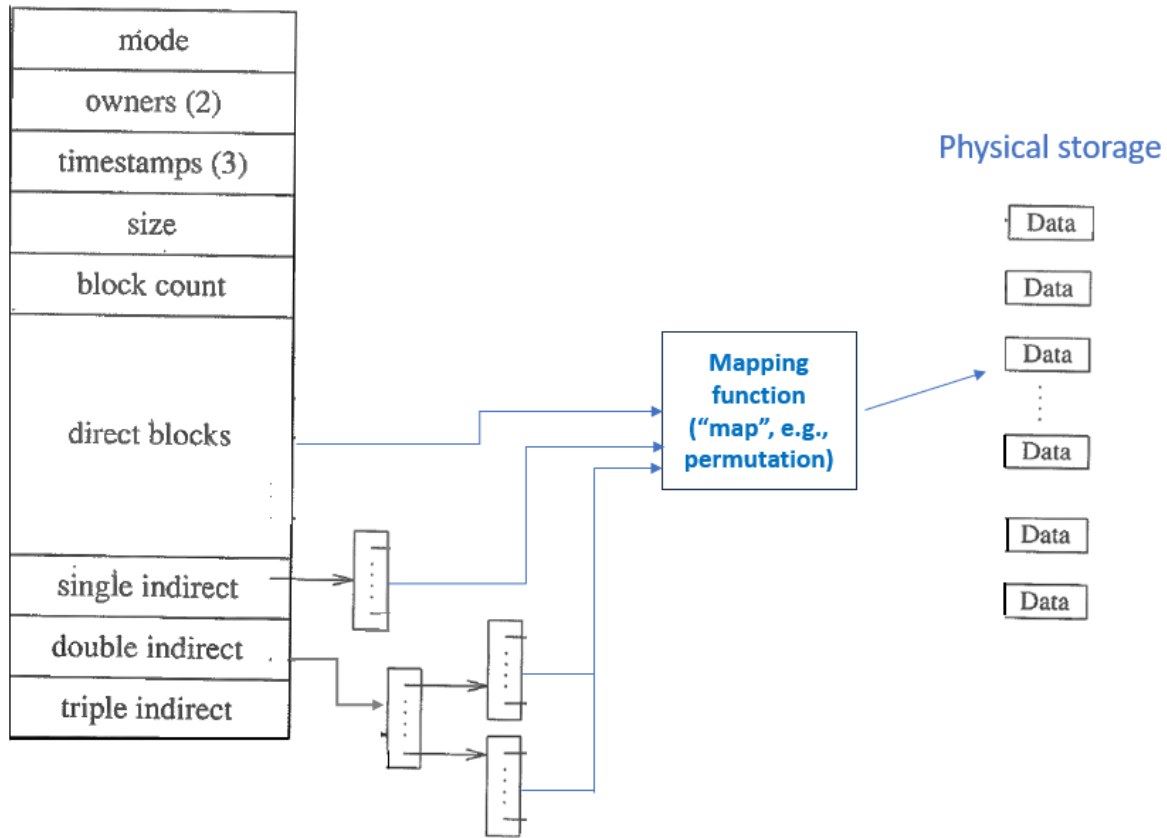
228. Locasto introduces two security features: [i] a cryptographic file system that encrypts the stored data and a mapping function that uses a permutation function and [ii] a spreading seed to randomly distribute real file data throughout a large address space. EX1005, 3-6, FIGS. 3-4; §VIII.B, *supra*.

229. In the Leffler-Locasto Combination, the file system Leffler teaches is enhanced with security features like those taught in Locasto, (e.g., adding a mapping function and encrypting the stored data, as illustrated below). As described by Locasto, such “[c]ryptographic file systems provide data confidentiality by employing encryption to protect files against unauthorized access.” EX1005, 1. Thus, as explained further below in §XI.C.1, a POSITA would have appreciated the predictable benefits of incorporating Locasto’s security features with Leffler’s efficient file system and associated data structures. Moreover, the POSITA would

have understood the compatibility of Locasto's teachings with Leffler, such that Locasto's technical improvements could be implemented while preserving the fundamental aspects of Leffler's disclosure. For example, Leffler describes the BSD (a Unix variant) file system and Locasto describes the Linux (another Unix variant) "Ext2fs" file system. EX1004, 1:45-50, 3:14-18, 5:41-42; EX1005, 3, 8; EX1008; [0027]; EX1009, [0400].

230. In operation, when a program accesses a file at a given position, the Leffler-Locasto Combination determines the inode (e.g., as illustrated in Leffler's FIGS. 7.3, 4.4, 7.6) and uses the block mappings in the inode to determine the (unspread) physical blocks. EX1006, 194, FIG. 7.6. Then, as shown in the visual aid below, the Combination employs a process like Locasto's (see §VIII.B) to facilitate the conversion of (unspread) physical block numbers in Leffler's inode to determine the spread physical block numbers. EX1005, 5. More specifically, the unspread physical numbers are converted to spread physical block numbers using Locasto's "map" function, e.g., a pseudo-random permutation function and a "spreading seed." EX1005, 4-5. The mapping function "randomly distribute[s] the real file data...throughout a large address space so that an attacker would have to first guess the spreading schedule before he attempts a brute force search on the rest of the key." EX1005, 4.

Leffler-Locasto Combination



Leffler-Locasto-Combination: Visual Aid

231. As shown in the visual aid above, the Leffler-Locasto Combination applies Locasto's mapping function (*map*) to the block numbers determined using Leffler's inode (e.g., by determining the block using the list of "direct blocks") and the bits of a *spreading seed* to determine the spread physical block number (*pbn*):

$$\text{spread_pbn_index} = \text{map}(\text{pbn}, \text{spreading_seed})$$

$$\text{spread} = \text{get_from_index}(\text{spread_pbn_index})$$

EX1005, 5-6.

232. A POSITA would have understood that these two functions can be expressed equivalently as the single (composite) mapping function:

spread_pbn = get_from_index(map(pbn, spreading_seed))

1. Reasons to Combine

233. The following reasons, taken separately or collectively, support the above-described Leffler-Locasto Combination.

234. First, a POSITA would have been motivated to pursue the Leffler-Locasto Combination for the purpose of maintaining confidentiality of the data in Leffler's file system. EX1005, 3-6. As Locasto explains, "[c]ryptographic file systems provide data confidentiality by employing encryption to protect files against unauthorized access." EX1005, 1. The POSITA would have appreciated the importance of protecting sensitive data against unauthorized access, e.g., as described by Custer, Garfinkel, and Arnold. EX1010, 5; EX1011, 5, 14; EX1015, 6; see §VII.C.1. Thus, a POSITA would have been prompted to bolster Leffler's file system disclosure with data encryption teachings from a supplemental reference like Locasto to achieve this benefit.

235. Second, a POSITA would have been motivated to pursue the Combination to achieve in Leffler the specific benefits espoused in Locasto. For example, Locasto's approach to encryption "reduc[es] the performance penalty

associated with using larger key sizes without decreasing the level of security.” EX1005, 2. As Locasto explains, by “using only part of the key for cryptographic operations” and “[t]he rest of the key...to unpredictably spread the data into the file’s address space,” “a user may decrease the performance penalty they pay for employing a cryptographic file system.” *Id.*; see also 1 (“improved performance relative to a cipher with a larger key-schedule”), 12 (“calculation of a new index has a negligible impact on performance”) (“we have increased the effective cryptographic key length by 28 bits, correspondingly increasing an attacker’s work factor by 2^{28}). From the POSITA’s perspective, Locasto’s balance of security and performance would have been an advantageous addition to Leffler’s file system. (In fact, Locasto explains that different ciphers can be used depending on various design objectives. EX1005, 10.)

236. Third, “design incentives” and “market forces” would have prompted a POSITA to pursue a combination of Leffler and Locasto. For example, a POSITA would have known that data security solutions were incorporated into popular file systems, such as the WINDOWS NT file system. For example, as explained in a WINDOWS NT reference book, “Data security is of primary importance to businesses and government agencies that process private or sensitive information...Such customers need guarantees that their data will be secure from unauthorized access.” EX1010, 5. It was likewise well known that skilled artisans

and market participants were actively pursuing secure variations of the UNIX operating system that Leffler describes, some of which included built-in encryption programs. EX1015, 6 (“Government purchases are the largest market for computers. The government requires highly trusted systems. The UNIX operating system has been augmented by many firms in order to gain better ratings.”); EX1011, 14 (“Although there is no standard UNIX program that performs encryption using the DES [encryption algorithm], Sun Microsystems’ operating system does include such a program[.]”). Accordingly, the POSITA would have sought data security improvements to Leffler’s file system in an effort to compete and distinguish over competitor products in the marketplace.

237. Fourth, by the time of the ’686 patent in 2007, as I discussed in §VIII.C.1, the scientific and patent literature was replete with proposals to incorporate data encryption techniques into existing file systems. *See, e.g.*, EX1012, Abstract, *generally*; EX1013, [0088]. A POSITA would have been motivated by this contemporaneous design and research trend to incorporate Locasto’s encryption-focused teachings into a file system like Leffler’s.

238. Fifth, a POSITA would have been led to the Leffler-Locasto Combination because it merely involves the application of a known technique (Locasto’s data encryption) to a known system (Leffler’s file system) to achieve

predictable results (a file system with improved data security). Each element in combination merely performs the same function as it does separately. Since Leffler describes Unix, including the file system, and Locasto describes a Linux prototype (where Linux is Unix-like), my discussion in §VIII.C.1 also applies to the Leffler-Locasto Combination.

2. Reasonable Expectation of Success

239. A POSITA would have had a reasonable expectation of success in creating a combination like the Leffler-Locasto Combination, which would have required only routine programming knowledge well within the skill of a POSITA. For example, the Leffler-Locasto Combination includes components of a file system and known encryption techniques to enhance the security of such file systems. *Supra*, §§VIII.A-C, XI.A-C.

240. Integrating these aspects of Leffler and Locasto would have been a routine programming task for a POSITA. As discussed, through the lens of a POSITA, this Combination would have amounted to nothing more than the use of a known technique (e.g., spreading, as taught by Locasto) to improve similar file systems (e.g., the BSD 4.3 file system of Leffler) in a similar way and combining prior art elements according to known methods to yield the predictable and beneficial results described above. Further, since Leffler describes Unix, including

the file system, and Locasto describes a Linux prototype (where Linux is Unix-like), my discussion in §VIII.C.2 also applies to the Leffler-Locasto Combination.

D. Claim Element Analysis

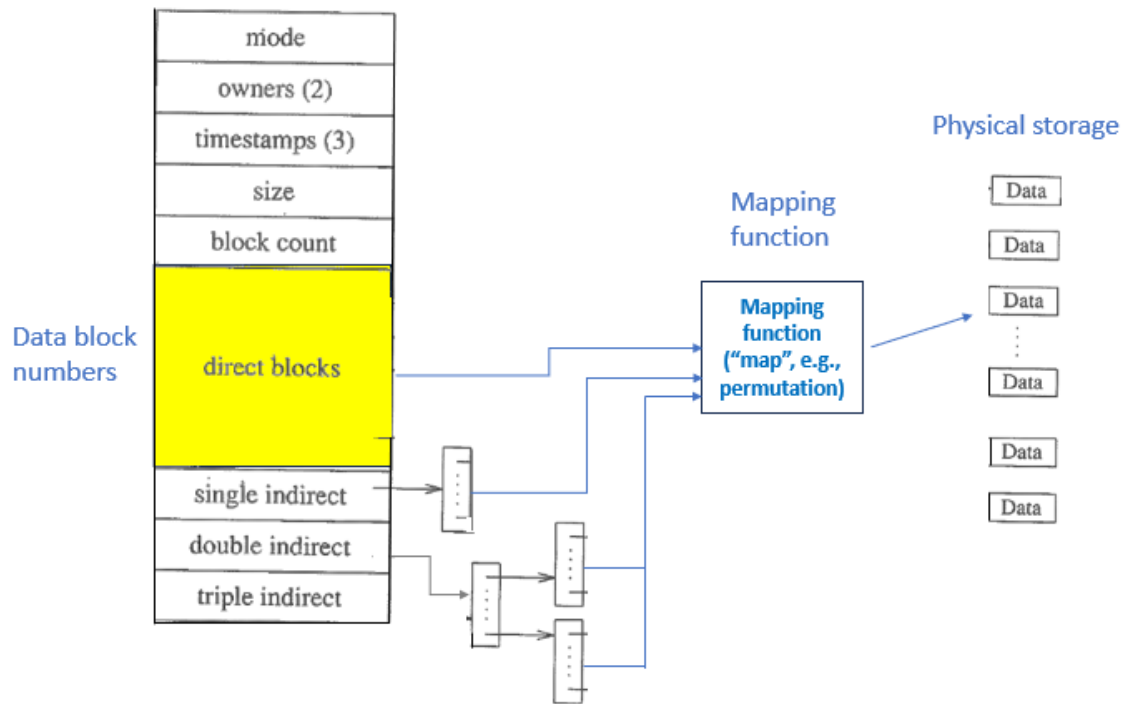
1. Claim 1

(i) *Element [1.pre]: A method of providing data security for use with a file system, the method comprising:*

241. The Leffler-Locasto Combination renders obvious Element [1.pre] and Claim 1 as a whole. For example, the Combination includes Leffler’s description of a “block-device interface” including “the ***filesystem***” that converts “from the user abstraction of a disk as an array the structure imposed by the physical medium.” EX1006, 171; *see also* 187, 191-192. The Combination additionally includes Locasto’s explanation that “[c]ryptographic file systems provide ***data*** confidentiality by employing encryption to protect files against unauthorized access.” EX1005, 1; *see also supra* §§XI.A-C (discussing the combination). Locasto then describes the Camouflage cryptographic file system (Camouflage FS) that “offers improved performance relative to a cipher with a larger key-schedule, while providing the same ***security*** properties.” EX1005, 1.

(ii) ***Element [1.1.1]: applying a mapping function to data block numbers that are associated with a file,***

242. The Leffler-Locasto Combination renders obvious Element [1.1.1]. For example, Leffler explains that “most UNIX systems use the concept of an index node, or inode” that “contains an array of pointers to the blocks in the file.” EX1006, 194. “The system can convert from a logical block number to a physical [block] number by indexing into the array using the logical block number.” EX1006, 194-195 (“In addition to the logical-to-physical block mapping, the inode contains other information about the contents of the file.”). Thus, the Combination includes ***data block numbers*** (e.g., Leffler’s physical block numbers) as illustrated below. For the same reasons I described in §VIII.D.1, Element [1.1.1], a POSITA would have understood or found obvious that physical block numbers are ***data block numbers*** at least since physical block numbers correspond to data managed by a file system and stored on a physical disk. In other words, as would have been clear to a POSITA, Leffler’s physical ***block numbers*** are used by the file system to store and access ***data***.



Leffler-Locasto-Combination: Visual Aid

243. Notably, Leffler’s disclosure of an “inode” in the context of a Unix operating system matches the description of the ’686 patent’s preferred embodiment. See EX1001, 2:1-28 (“The inode data structure is used in, for example, a Unix file system[.]”), 3:50-4:24 (describing an example where “an inode 210 of file 205 points to four (4) data block numbers”), Figures 1-2. Additionally, Leffler writes that “inode” is shorthand for “index node.” EX1006, 194 (“most UNIX systems use the concept of an *index node*, or *inode*,” emphasis in the original); see also ¶43, *supra* (citing EX1016).

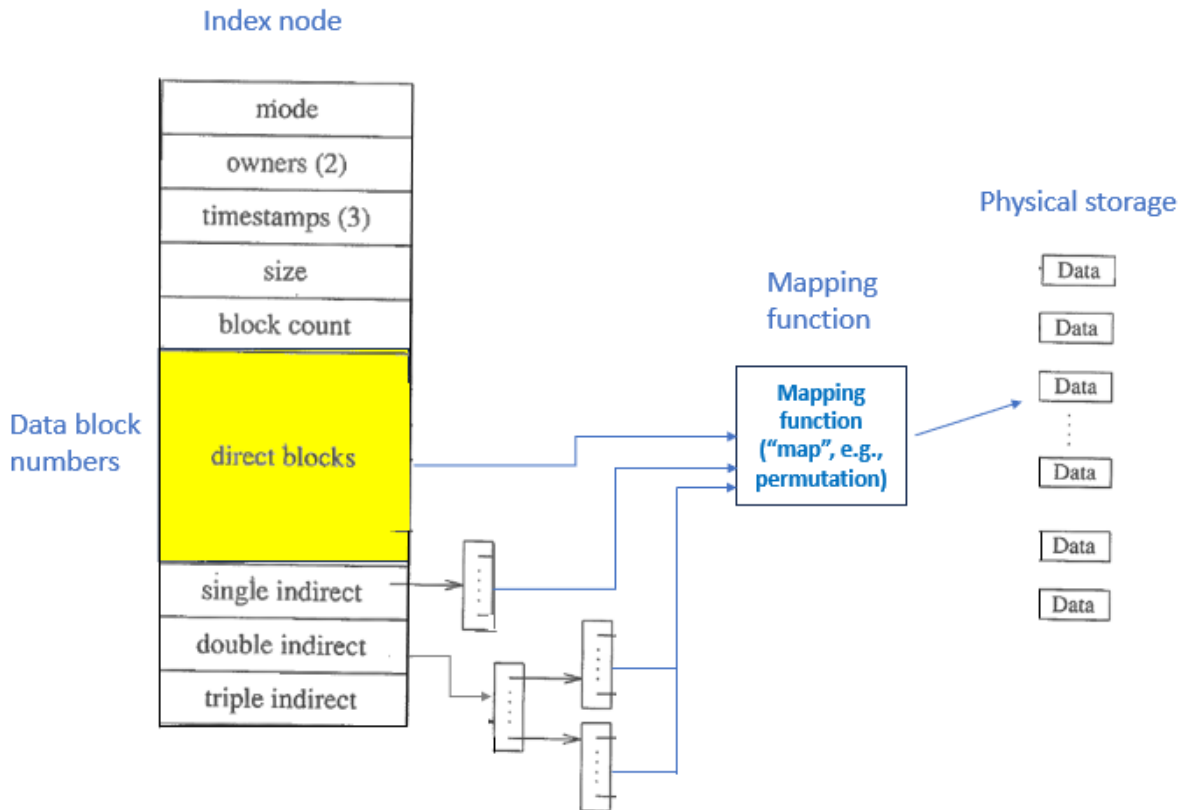
244. The Combination further teaches or renders obvious that the physical block numbers (***block numbers***) are ***associated with a file***. For example, Leffler explains that an “inode contains an array of pointers to the blocks in the file.” EX1006, 194. Thus, it would have been understood by or obvious to a POSITA that the physical ***block numbers*** contained in Leffler’s inode ***are associated with a file*** since the inode contains an array of pointers to the blocks in the file. Leffler further explains that an inode is “an on-disk data structure describing the file.” EX1006, 187.

245. As discussed at Section XI.C, the Leffler-Locasto Combination ***applies a mapping function*** to physical block numbers (***data block numbers***). Specifically, the Combination’s “pseudo-random permutation (PRP) function takes the” physical block number “and the bits of the spreading seed to return a” spread physical block number, which is the physical address of the data block on the physical storage disk. EX1005, 4-5.

(iii) ***Element [1.1.2]: wherein the data block numbers are contained in an index node associated with said file; and***

246. The Leffler-Locasto Combination renders obvious Element [1.1.2]. As discussed above at Element [1.2.1], the Combination includes physical (***data***) ***block numbers*** that are ***contained*** in an inode (***index node***). *E.g.*, EX1006, 194 (“index node, or inode”). Here again, I note that Leffler’s disclosure of an “inode” in the

context of a Unix operating system matches the description of the '686 patent's preferred embodiment. See EX1001, 2:1-28, 3:50-4:24, Figures 1-2.



Leffler-Locasto-Combination: Visual Aid

247. Further, as would have been understood by or obvious to a POSITA from Leffler's teachings, the inode is ***associated with a file***. For example, Leffler explains that "the inode contains an array of pointers to the blocks in the ***file***." EX1006, 194; see also 189 ("Every ***file*** has an ***associated inode***..." and "Each ***file*** has a single ***inode***...").

(iv) ***Element [1.2.1]: obtaining mapped data block numbers after applying the mapping function,***

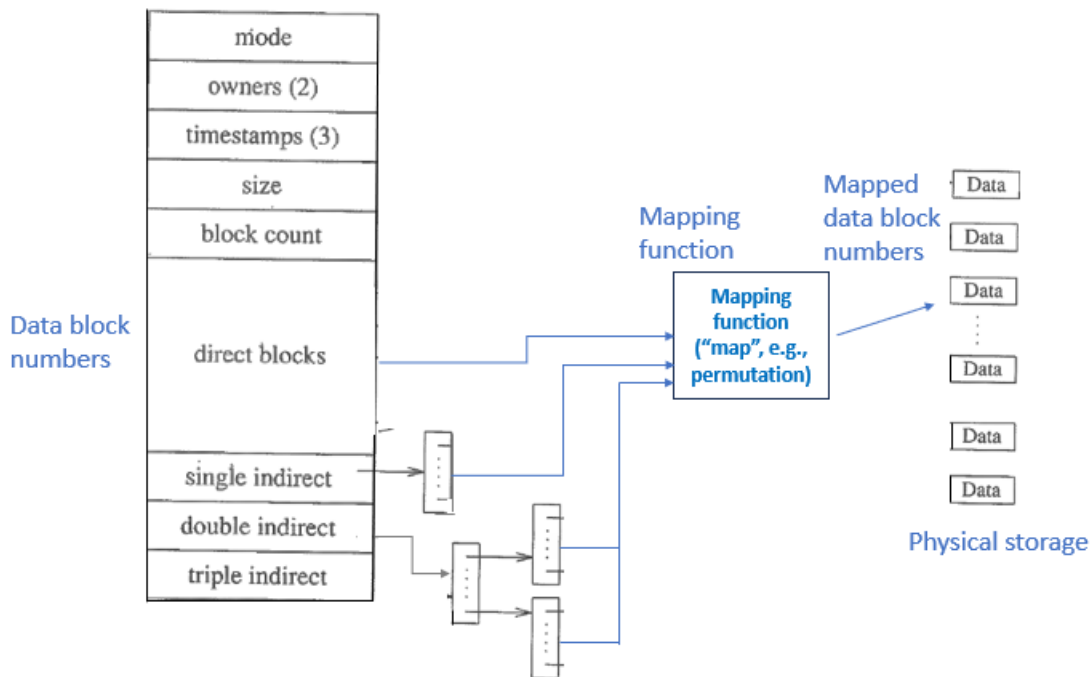
248. The Leffler-Locasto Combination renders obvious Element [1.2.1]. As discussed, the Combination applies “a pseudo-random permutation (PRP) function [that] takes the [physical block number in Leffler’s inode] and the bits of the spreading seed to return a new index value.” EX1005, 4-5.

spread_pbn_index = map(pbn, spreading_seed)

249. The spread physical “block number is then derived from this new index.” EX1005, 5.

spread_pbn = get_from_index(spread_pbn_index)

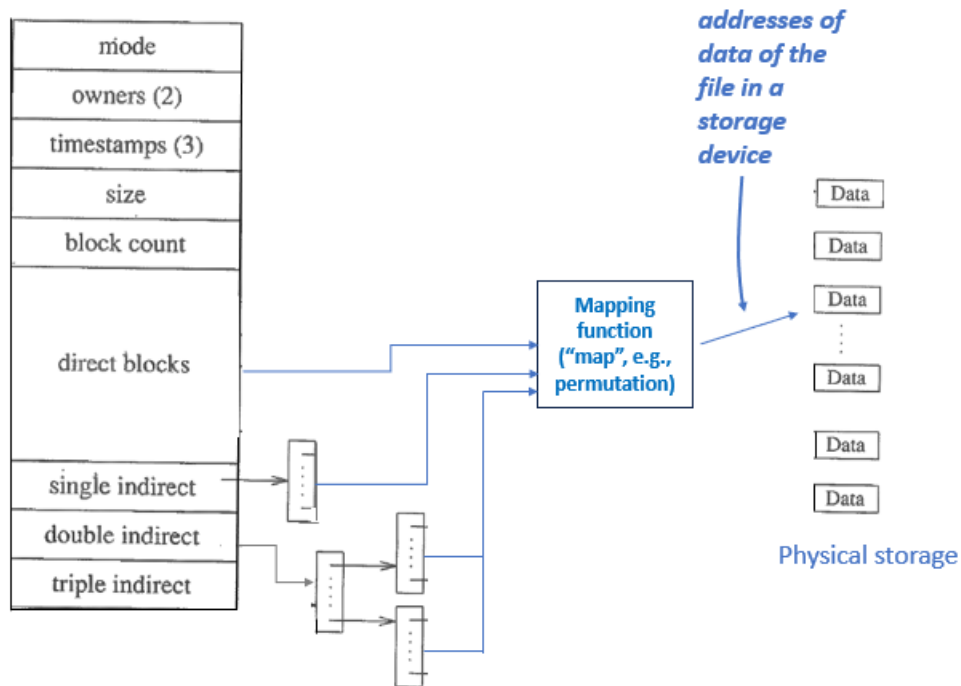
250. Thus, in the Leffler-Locasto Combination, the spread physical block numbers (***mapped data block numbers***) ***are obtained after applying*** the “map” (***mapping***) ***function***.



Leffler-Locasto-Combination: Visual Aid

- (v) ***Element [1.2.2]: wherein the mapped data block numbers are addresses of data of the file in a storage device.***

251. The Leffler-Locasto Combination renders obvious Element [1.2.2]. Specifically, it would have been understood by or obvious to a POSITA that the spread physical ***data block numbers are addresses*** of physical blocks on the disk (***storage device***) storing ***data of the file***. See *supra*, §XI.C. For example, Locasto explains that a block number determined by its mapping function is “the block on the disk” where the data at a particular file position resides. EX1005, 5.

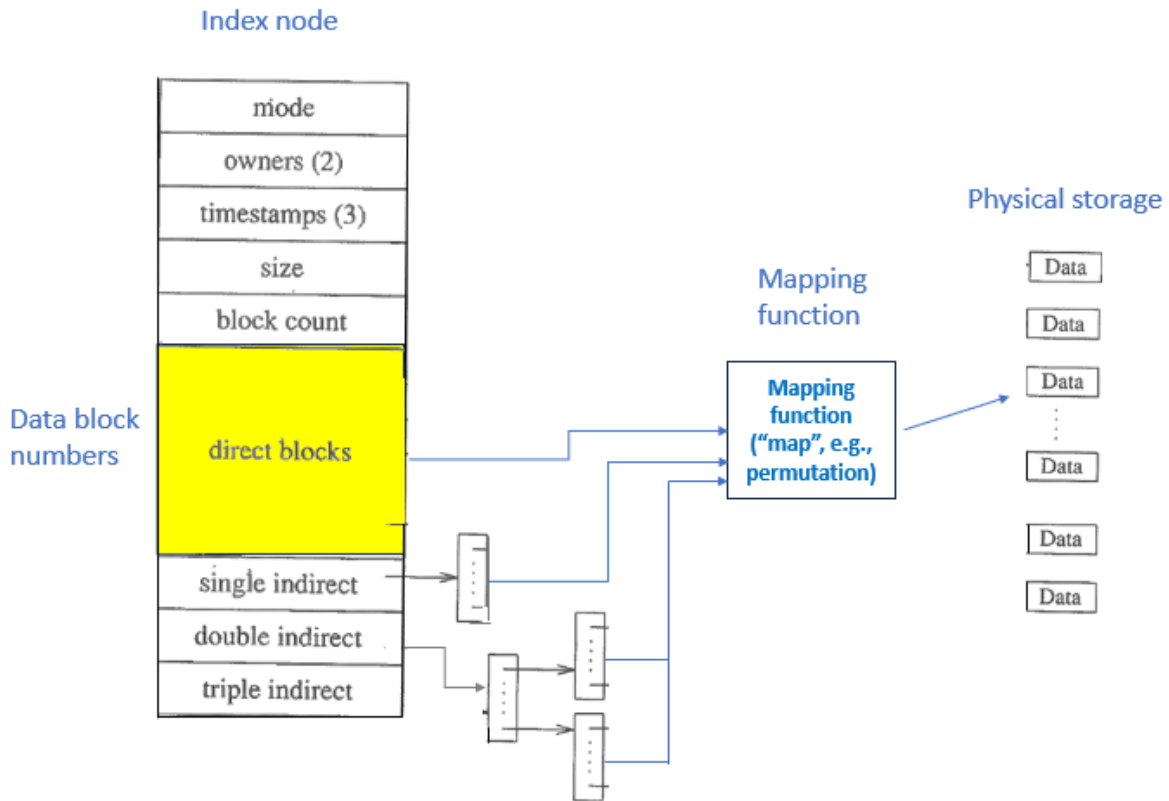


Leffler-Locasto-Combination: Visual Aid

2. Claim 2

- (i) **Claim [2]:** *The method of claim 1, further comprising: storing a mapping of the data block numbers and the mapped data block numbers.*

252. The Leffler-Locasto Combination renders obvious Claim [2]. For example, as discussed above at Element [1.1.1], the Combination applies a mapping function to the physical block numbers (**data block numbers**) to obtain the physical addresses of the data blocks on the physical storage disk, as illustrated in the visual aid below. See §XI.C, *supra*.



Leffler-Locasto-Combination: Visual Aid

253. As discussed at Section XI.C, in the Leffler-Locasto Combination, spread (mapped) data block numbers are determined from data block numbers using the equation:

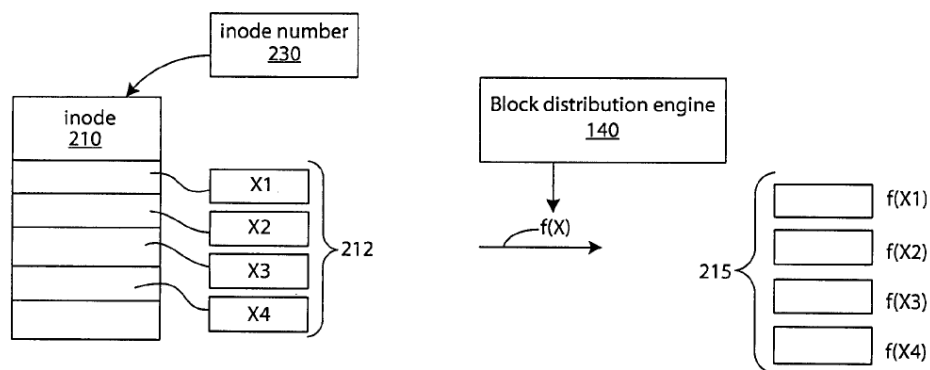
$$\text{spread_pbn} = \text{get_from_index}(\text{map}(\text{pbn}, \text{spreading_seed}))$$

254. Since the Leffler-Locasto Combination computes the spread physical block numbers (**mapped data block numbers**) from the physical block numbers (**block numbers**), a POSITA would have understood that the Leffler-Locasto Combination **stores the mapping** of physical block numbers (**block numbers**) to

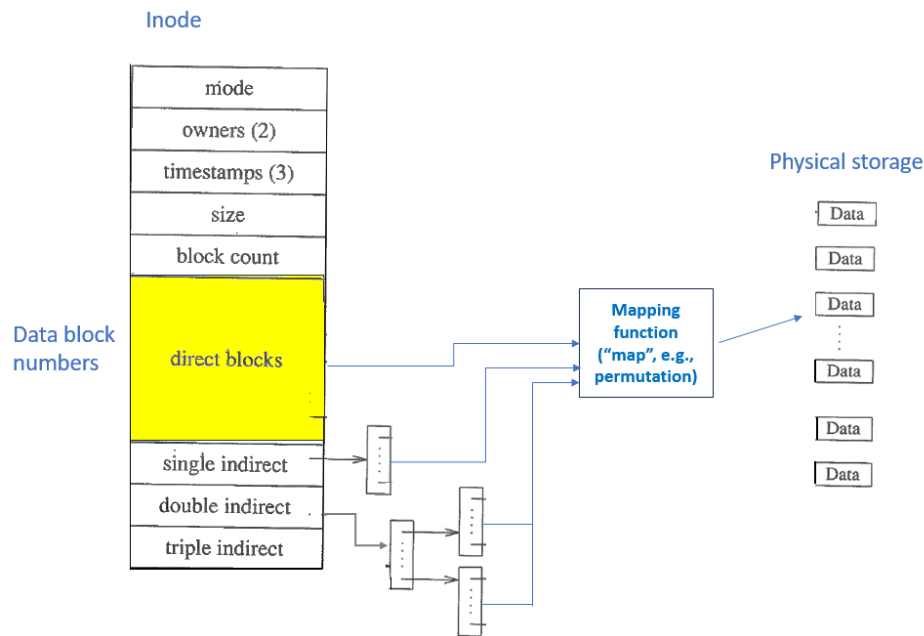
spread physical block numbers (***mapped block numbers***) at least at the conclusion of the calculation of the spread physical block numbers.

255. For example, consider how a conventional computer would process the formula given above, $spread_pbn = get_from_index(map(pbn, spreading_seed))$. The computer applies the function *map* to the variables *pbn* and *spreading_seed*, which are stored in memory, producing an interim result. The computer then applies the function *get_from_index* to that interim result, producing a final result that is stored in the variable *spread_pbn*. Thus, the computer stores the mapping of the variable *pbn* (***block number***) to the *spread_pbn* result (***mapped block number***) at least while the computer applies the composite function.

256. Notably, the Leffler-Locasto Combination has an analogous structure to '686 patent's structure, as illustrated below.



EX1001 ('686 patent), FIG. 2 (excerpt).



Leffler-Locasto Combination: Visual Aid

3. Claim 3

- (i) ***Element [3.1]: The method of claim 1, wherein a read operation for the file comprises: obtaining the data block numbers of the file;***
- (ii) ***Element [3.2]: obtaining the mapped data block numbers of the file based on the mapping function that is applied to the data block numbers; and***
- (iii) ***Element [3.3]: reading the data in the mapped data block numbers in the storage device.***

257. The Leffler-Locasto Combination renders obvious Elements [3.1]-[3.3]. As discussed at Section X.A, Leffler provides a full-length discussion of the 4.3BSD UNIX operating system, including a breakdown of the internal file system and a description of the calls and operations involved in reading and writing to the

storage disk. *See generally* EX1006, 169-223. Consistent with how 4.3BSD works, when reading or writing, the Leffler-Locasto Combination obtains the logical block number of the data to be **read**/written and uses that logical block number, via the mapping table, to **obtain** the corresponding physical block number (**data block number**, per Element [1.1.1]). *See* EX1006, 192, 194 (“The system can convert from a logical block number to a physical [block] number...”), 195 (“logical-to-physical block mapping”).

258. Next, as discussed at Elements [1.1.1], [1.2.1], [1.2.2], the Combination **applies** Locasto’s **mapping function** to the physical block number to determine a spread physical block number (**mapped data block number**) that is the address on the disk where the logical block data is stored. The Combination then completes the **read** or write request on the physical disk using the spread physical block number (**reading the data in the mapped data block numbers**). EX1006, 206-207 (Figure 7.12); §X.A.

4. Claim 4

- (i) ***Element [4.1]: The method of claim 1, wherein a write operation for the file comprises: allocating data block numbers of free data blocks in the storage device;***

259. The Leffler-Locasto Combination renders obvious Element [4.1]. Leffler explains that the operating system maintains a record of **free blocks**.

EX1006, 195, 197. For example, a “free list” contains “a list of all the **free blocks** in the filesystem.” EX1006, 195. Alternatively, the system can maintain “a bit map describing available [(**free**)] **blocks** in the cylinder group.” EX1006, 197. As files grow, the “filesystem [] handles the **allocation** of new blocks to files.” EX1006, 193. Leffler continues, “[e]ach time a process does a **write** system call, the system checks to see whether the size of the file has increased,” and if “the file needs to be expanded, the request is rounded up to the next fragment size and only that much space is **allocated**.” EX1006, 199. Since a single file system block can be divided into one or more fragments, a POSITA would have understood and/or found obvious that a block can be exactly one fragment. EX1006, 198 (“The developers obtained this efficiency by allowing the division of a single filesystem block into one or more *fragments*,” emphasis in the original). When a file is **allocated** a new block, a POSITA would have understood that the block would be assigned from among the **free data blocks on the storage device**.

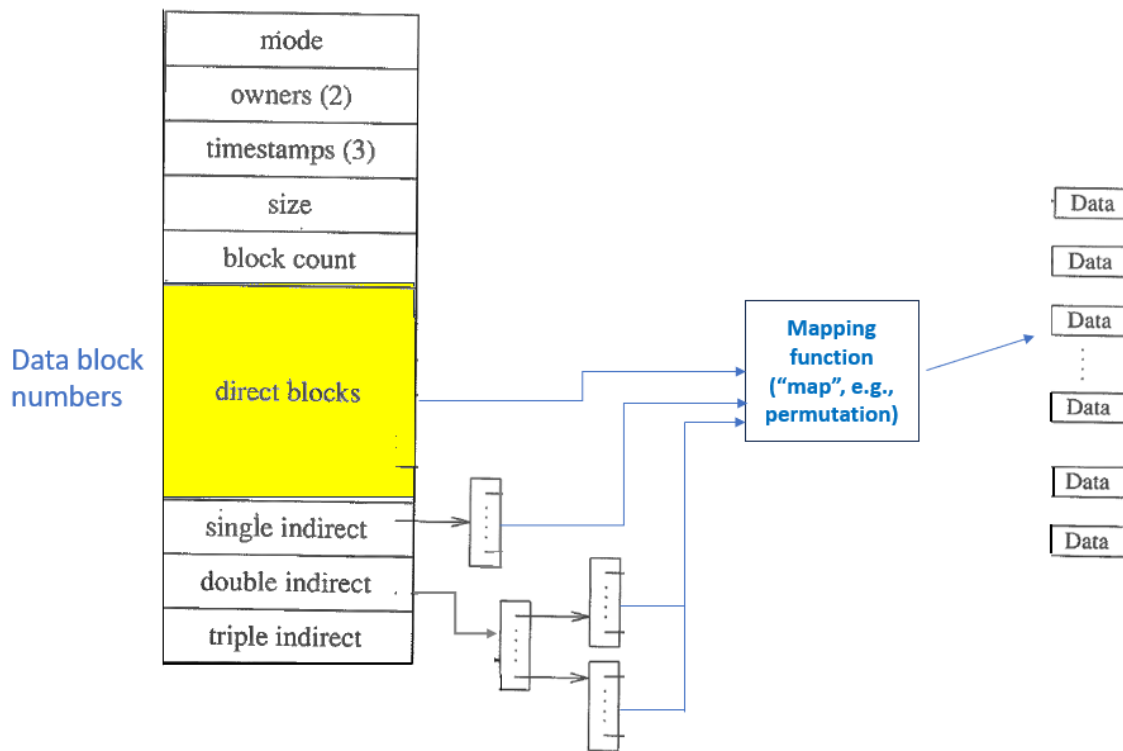
260. Leffler further teaches that an inode contains a file’s “logical-to-physical block mapping.” EX1006, 194-195. Therefore, a POSITA would have understood and/or found obvious that since the inode stores the block numbers of the physical blocks allocated during the **write operation, data block numbers of free data blocks** are also **allocated**. As I described in §X.B.3, Element [4.1], in practice, the operating system identifies a free block on the storage device from the free list

(list of free blocks), removes the block from the free list, and assigns the block to the file.

261. For example, since Leffler teaches that the “free list” contains “a list of all the free blocks in the filesystem,” a POSITA would have understood and/or found obvious that when a block is removed from the free list, its corresponding data block number is removed from the free list (“list of all the free blocks in the filesystem”) and added to the inode. EX1006, 194-195.

(ii) ***Element [4.2]: linking the data block numbers to a file name of the file;***

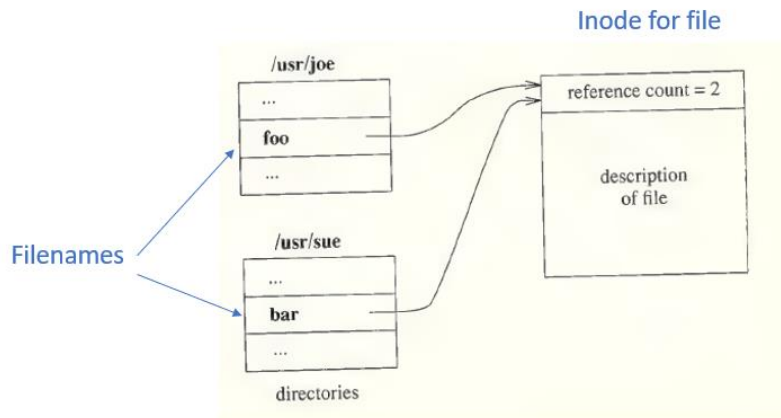
262. The Leffler-Locasto Combination renders obvious Element [4.2]. For example, as explained above at Element [1.1.2], in the Leffler-Locasto Combination, the inode contains the ***data block numbers of a file***, as illustrated in the visual aid below. *Supra*, §XI.C.



Leffler-Locasto-Combination: Visual Aid

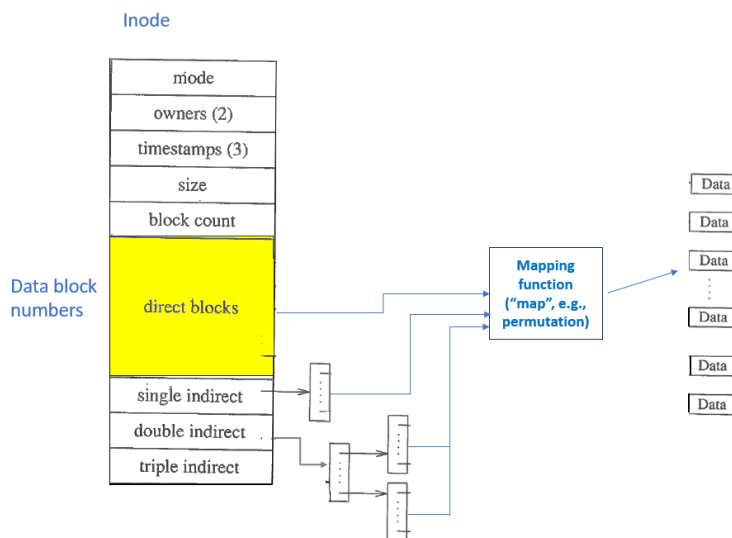
263. Further, per Leffler, “[f]ilenames are maintained in directories rather than in inodes because a file may have many names, or links, and the name of a file may be large (up to 255 bytes in length).” EX1006, 195. Directories further contain directory entries, and each entry contains a pointer to the file’s inode. EX1006, 187.

264. Leffler provides an example in which two different directory entries reference the same file, so the inode has a reference count of two. EX1006, 189, FIG. 7.3.



EX1006, FIG. 7.3 (annotated).

265. The inode further contains the data block numbers, as illustrated in the visual aid below. EX1006, 194.



Leffler-Locasto-Combination: Visual Aid

266. Thus, a POSITA would have understood or found obvious that when data are written to a new block for a file, the allocated physical block numbers (**data block numbers**) are added to the inode (as described re: Element [4.1]), which are

linked to at least one directory entry containing the **filename**. EX1006, 187, 189, FIGS. 7.3, 7.6; §XI.D.4, Element [4.1].

(iii) **Element [4.3]: applying the mapping function to the data block numbers in order to obtain the mapped data block numbers; and**

267. The Leffler-Locasto Combination renders obvious Element [4.3]. See §XI.D.4, Element [3.2] (describing the **application of the mapping function** in both read and write operations).

(iv) **Element [4.4]: writing the data of the file into the mapped data block numbers in the storage device.**

268. The Leffler-Locasto Combination renders obvious Element [4.4]. For example, per Leffler, the Combination uses the “*bio*” and “*dev*” calls to complete the read or **write** request on the physical disk using the physical (**mapped**) **block number**. EX1006, 207; §X.A, §X.B.2, Claim 3.

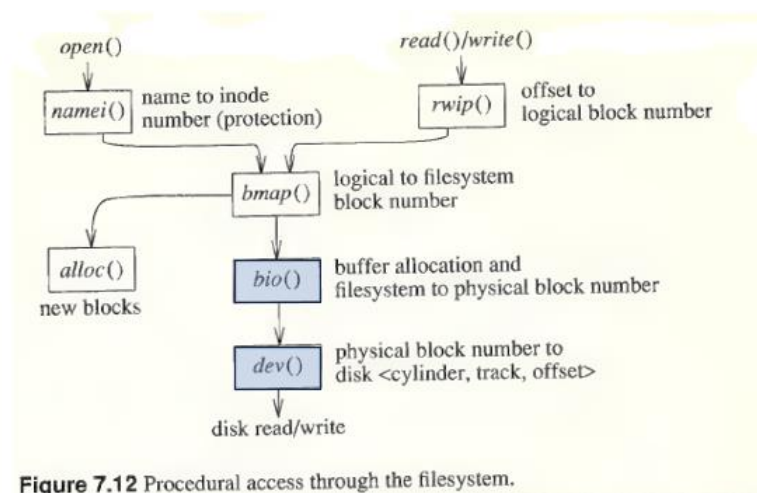


Figure 7.12 Procedural access through the filesystem.

EX1006, FIG. 7.12 (annotated).

5. Claim 5

- (i) ***Claim [5]: The method of claim 1, wherein the mapping function is an equation that is applied to each data block number of the file.***

269. The Leffler-Locasto Combination renders obvious Claim [5]. For example, as described in §§XI.C, XI.D.6, Element [1.1.1], the Leffler-Locasto Combination that ***applies a mapping function*** to physical block numbers (***data block numbers***) ***of a file***. EX1005, 4-5. The rudimentary concept of performing this technique at ***each*** logical block number would have been understood by or obvious to a POSITA. First, when each data block of a file is written, the mapping is performed on the data block number of that data block. EX1005, 4 (“The purpose of the spreading schedule is to randomly distribute the real file data throughout a large address space”). Therefore, since all of the data blocks are written, the mapping function is applied for each data block number of the file. Second, the mapping function is applied to each data block number for the purpose of retrieving the complete set of data blocks for the file and, at least in some instances, entire files will be read.

270. As discussed at §XI.C, the ***mapping function*** corresponds to the following composite ***equation***:

spread_pbn = get_from_index(map(pbn, spreading_seed))

EX1005, 5.

271. Moreover, Locasto teaches that the mapping function includes a pseudo-random permutation function based on the well-known cipher algorithms “RC6” and “RC4,” both of which include multiple mathematical operations. EX1005, 4-5, 10. For example, as described above, EX1014 describes the mathematical operations in RC6.

6. Claim 6

- (i) ***Claim [6]: The method of claim 1, further comprising: selecting a complexity level of the mapping function in order to set a security level and efficiency of the mapping function.***

272. The Leffler-Locasto Combination renders obvious Claim [6]. See also §VIII.D.4, Claim [6], *supra* (discussing Locasto’s ***mapping function***).

7. Claim 7

- (i) ***Claim [7]: The method of claim 1, wherein the data block numbers are addresses of data of the file in the storage device if the mapping function is not applied to the data block numbers.***

273. The Leffler-Locasto Combination renders obvious Claim [7]. See also §VIII.D.5, Claim [7], *supra* (explaining, *inter alia*, that Locasto’s ***mapping function*** is optional).

8. Claim 8

- (i) ***Claim [8]: The method of claim 1, wherein the mapping function prevents copying of data of the file in the storage device.***

274. The Leffler-Locasto Combination renders obvious Claim [8]. *see also* §VIII.D.6, Claim [8], *supra* (discussing Locasto's ***mapping function***)

XII. CONCLUSION

275. I declare that all statements made in this Declaration of my own knowledge are true and that all statements made on information and belief are believed to be true. I further declare that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both (under Section 1001 of Title 18 of the United States Code).

Executed this 28th day of August, 2025.



Dr. Nenad Medvidovic

APPENDIX A

Nenad Medvidović

TRW Professor in Engineering

Professor and Chair, Thomas Lord Department of Computer Science

School of Advanced Computing and Viterbi School of Engineering

1031 Downey Way

University of Southern California

Los Angeles, CA 90089-0262 U.S.A.

+1-310-702-3309 (cell)

nen@usc.edu (email)

<http://softarch.usc.edu/~nen/> (www)

Table of Contents

Education	2
Employment and Major Organization Leadership	2
Honors, Awards, Fellowships	3
Selected Statistics	4
Graduated Doctoral Students	4
Publications	5
Research Support	31
Patents	39
Teaching	39
Supervised Students	41
Formal Presentations	45
Professional Service	57
University Service	69
Professional Associations	72
Consulting	72

1 Education

Doctor of Philosophy (3/99, Cumulative GPA: 4.00)

University of California, Irvine — Department of Information and Computer Science
Dissertation: *Architecture-Based Specification-Time Software Evolution*

Master of Science (6/95, Cumulative GPA: 4.00)

University of California, Irvine — Department of Information and Computer Science

Bachelor of Science, Summa Cum Laude (8/92, Cumulative GPA: 4.00) Arizona State

University — Computer Science and Engineering Department

2 Employment and Major Organization Leadership

7/25 – present TRW Chair in Engineering

University of Southern California, Los Angeles

7/22 - present Chair, Thomas Lord Department of Computer Science

University of Southern California, Los Angeles

2/11 - present Professor, Thomas Lord Department of Computer Science

University of Southern California, Los Angeles

3/05 - 2/11 Associate Professor with Tenure, USC

1/99 - 3/05 Assistant Professor, USC

7/21 - 6/22 Vice-Chair for Tenure Affairs, Computer Science Department

University of Southern California, Los Angeles

1/18 - 12/22 Editor-in-Chief

IEEE Transactions on Software Engineering

7/15 - 6/18 Chair, Special Interest Group on Software Engineering (SIGSOFT)

Association for Computing Machinery (ACM)

7/15 – 9/17 Chair, Steering Committee

ACM SIGSOFT Symposium on the Foundations of Software Engineering

11/13 – 10/15 Chair, Steering Committee

International Conference on Software Engineering

7/11 - 6/15 Associate Chair for Ph.D. Affairs, Department of Computer Science, University

of Southern California, Los Angeles

1/09 - 1/13 Director, Center for Systems and Software Engineering

Viterbi School of Engineering, University of Southern California, Los Angeles

9/94 - 12/98 Graduate Student Researcher

University of California, Irvine

9/93 - 8/94 Teaching Assistant

University of California, Irvine

6/93 - 8/93 Software Engineering Intern

Rupp Technology, Phoenix, AZ

8/92 - 8/93 Teaching Assistant

3 Honors, Awards, Fellowships

2025	Most Influential Paper Award, IEEE Transactions on Software Engineering
2023	Alumni Hall of Fame, School of Information and Computer Sciences, University of California, Irvine
2023	IEEE Computer Society Golden Core Member
2022	Elected Member of IFIP Working Group 2.4 – Software Implementation Technologies
2020	ACM SIGSOFT Distinguished Service Award
2020	Most Influential Paper Award, International Symposium on Software Engineering for Adaptive and Self-Managing Systems
2018	Distinguished Engineering Merit Award, Orange County Engineering Council
2018	Best Paper Award, International Conference on Software Architecture
2017	Best Tool-Paper Award, International Conference on Automated Software Engineering
2017	Best Paper Award, International Conference on Software Architecture
2016	Fellow, IEEE
2015	Distinguished Scientist, ACM
2010	USC Mellon Mentoring Award
2009	Best Poster Award, International Conference on Program Comprehension
2008	Most Influential Paper Award, International Conference on Software Engineering
2005, 2009	Finalist, Most Influential Paper Award, International Conference on Software Engineering
2007	IBM Real-Time Innovation Faculty Award
2007	Most cited software engineering paper published in a scientific journal in 2000
2005	Okawa Foundation Research Grant
2000-2005	National Science Foundation (NSF) CAREER Award
1999, 2000	USC, School of Engineering Teaching Honor Roll
1993-94	UC Irvine, Department of Information and Computer Science Fellowship
1992	Arizona State University, Moeur Award (top student in the graduating class)
1991-93	UPE National Honor Society for the Computing Sciences
1990-93	Golden Key National Honor Society
1990-92	Arizona Regents International Student Scholarship
1989-92	Dean's Honor List, Arizona State University

4 Selected Statistics

- *Total number of citations* (scholar.google.com) 21,553
- *h-index* (scholar.google.com) 65
- *Most Influential Scholars in Software Engineering* (aminer.org) 6th all-time
- *Most cited publication* (scholar.google.com) 3,194 citations [J27]
- *Most cited papers, IEEE Trans. on Software Engr.* (academic.microsoft.com) 7th [J27]
- *Most cited papers, ACM Trans. on Software Engr. & Methodology* 10th [J25] (academic.microsoft.com)

5 Graduated Doctoral Students

- PhD1** Marija Mikic, July 2004
currently VP of AI Foundations at Cruise
previously Senior Engineering Director at Google Labs
- PhD2** Nikunj R. Mehta, September 2004
currently a Founder and CEO at Falconry
formerly Consulting Member of Technical Staff at Oracle
- PhD3** Roshanak Roshandel, December 2006
currently Project Manager at Amazon
previously *Associate Professor* and *Department Chair* at Seattle University
- PhD4** Sam Malek, May 2007
currently *Professor* at UC Irvine
- PhD5** Chris Mattmann, August 2007
currently CTIO at Jet Propulsion Laboratory and *Adjunct Professor* at USC
- PhD6** Yuriy Brun, March 2008
currently *Professor* at University of Massachusetts
formerly *Computing Innovation (CI) Post-Doctoral Fellow* at U. of Washington
- PhD7** Chiyounng Seo, May 2008
currently Senior Engineer at AirBnB
formerly a Software Architect at Couchbase
- PhD8** George Edwards, July 2010
currently founder of Quandary Peak Research
- PhD9** David Woollard, January 2011
currently VP of Engineering at MoveWith
- PhD10** Daniel Popescu, November 2011
currently Staff Software Engineer at Google

- PhD11** Ivo Krka, January 2014
currently Co-Founder and Chief Software Engineer at Sportening
- PhD12** Hossein Tajalli, May 2014
currently Machine Learning Performance Architect at Intel
previously Senior Software Architect at NVIDIA
- PhD13** Joshua Garcia, May 2014
currently Assistant Professor at UC Irvine
- PhD14** Jae young Bang, March 2015
currently Software Litigation Consultant at Quandary Peak Research
- PhD15** Gholamreza Safi, July 2016
currently Software Engineer at Cloudera
- PhD16** Youn Kyu Lee, October 2017
currently Assistant Professor at Hongik University, Seoul, South Korea
- PhD17** Arman Shahbazian, May 2018
currently Software Engineer at Google
- PhD18** Duc Minh Le, August 2018
currently Software Engineer at Bloomberg
- PhD19** Yixue Zhao, December 2020
currently Researcher, Information Sciences Institute
formerly Computing Innovation (CI) Post-Doctoral Fellow at U. of Massachusetts
- PhD20** Adriana Sejfia, December 2023
currently Assistant Professor (Lecturer) at University of Edinburgh
- PhD21** Saghar Talebipour, August 2024
currently Software Engineer at Snap

6 Publications

Authored Books

- AB1** Richard N. Taylor, **Nenad Medvidović**, and Eric M. Dashofy. Software Architecture: Foundations, Theory, and Practice. *John Wiley & Sons*, January 2009, 736 pages.
ISBN-10: 0470167742. ISBN-13: 978-0470167748.

Edited Books and Journals

- EB2** Hakan Erdogmus, **Nenad Medvidović**, and Frances Paulisch, eds. IEEE Software – Special Issue on 50 Years of Software Engineering. *IEEE Computer Society*, vol. 35, no. 5, September/October 2018.
Winner of the 2019 APEX Award for Publication Excellence.
- EB3** Domenico Bianculli, **Nenad Medvidović**, and David Rosenblum, eds. 40 Editions of ICSE – The Ruby Anniversary Celebration. International Conference on Software Engineering, May 2018.
- EB4** **Nenad Medvidović** and Santonu Sarkar, eds. Proceedings of the 2017 Innovations in Software Engineering Conference, ACM, February 2017.

- EB5** Barbora Buhnova, Jorge Cuellar, Magnus Larsson, Javier López, **Nenad Medvidović**, Antonio Vallecillo, eds. Special Issue on Component-Based Software Engineering and Software Architecture, *Science of Computer Programming*, September 2014.
- EB6** Magnus Larsson and **Nenad Medvidović**, eds. Proceedings of the The 15th International ACM SIGSOFT Symposium on Component Based Software Engineering (CBSE-2012). ACM, June 2012.
- EB7** Harald Gall and **Nenad Medvidović**, eds. Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011). ACM, May 2011.
- EB8** Fabrice Saffre, **Nenad Medvidović**, and Alberto Montresor, eds. Proceedings of the 3rd International Conference on Self-Adaptive and Self-Organizing Systems. IEEE Computer Society, September 2009.
- EB9** **Nenad Medvidović** and Tetsuo Tamai, eds. Proceedings of the Warm Up Workshop for the 32nd International Conference on Software Engineering. ACM, April 2009.
- EB10** Robert Nord, **Nenad Medvidović**, Rene Krikhaar, Judith Stafford, Jan Bosch, eds. Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture. *IEEE Computer Society*, March 2006.

Refereed Journal Articles

- J1** Benyamin Beyzaei, Saghar Talebipour, Ghazal Rafiei, **Nenad Medvidović**, and Sam Malek. Automated Test Transfer across Android Apps using Large Language Models. Proceedings of the ACM on Software Engineering, vol. 2, no. ISSTA, article ISSTA098 (July 2025).
- J2** Jae young Bang, Yuriy Brun, and **Nenad Medvidović**. Collaborative Design Conflicts: Costs and Solutions. *IEEE Software*, vol. 35, no. 6 (November/December 2018).
- J3** Thibaud Lutellier, Devin Chollak, Joshua Garcia, Lin Tan, Derek Rayside, **Nenad Medvidović**, and Robert Kroeger. Measuring the Impact of Code Dependencies on Software Architecture Recovery Techniques. *IEEE Transactions on Software Engineering*, vol. 44, no. 2 (February 2017).
- J4** Pooyan Behnamghader, Duc Minh Le, Joshua Garcia, Daniel Link, Arman Shahbazian, and **Nenad Medvidović**. A Large-Scale Study of Architectural Evolution in Open-Source Software Systems. *Journal of Empirical Software Engineering*, vol. 22, no. 3 (June 2017).
- J5** Hamid Bagheri, Joshua Garcia, Alireza Sadeghi, Sam Malek, and **Nenad Medvidović**. Software Architectural Principles in Contemporary Mobile Software: from Conception to Practice. *Journal of Systems and Software*, vol. 119 (September 2016).
- J6** Yuriy Brun, Jae young Bang, George Edwards, and **Nenad Medvidović**. Self-Adapting Reliability in Distributed Systems. *IEEE Transactions on Software Engineering*, vol. 41, no. 8 (August 2015).
- J7** Chris Mattmann, Joshua Garcia, Ivo Krka, Daniel Popescu, and **Nenad Medvidović**. Revisiting the Anatomy and Physiology of the Grid. *Journal of Grid Computing*, vol. 13, no. 1 (March 2015).

- J8** Yuriy Brun and **Nenad Medvidović**. Entrusting Private Computation and Data to Untrusted Networks. *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 4 (July/ August 2013).
- J9** Hossein Tajalli and **Nenad Medvidović**. iDARE — A Reference Architecture for Integrated Software Environments. *Journal of Software: Practice and Experience*, vol. 44, no. 3 (March 2014).
- J10** Chris A. Mattmann, **Nenad Medvidović**, Sam Malek, George Edwards, and Somo Banerjee. A Middleware Platform for Providing Mobile and Embedded Computing Instruction to Software Engineering Students. *IEEE Transactions on Education*, vol. 55, no. 3 (August 2012).
- J11** Sam Malek, **Nenad Medvidović**, and Marija Mikic-Rakic. An Extensible Framework for Improving a Distributed Software System's Deployment Architecture. *IEEE Transactions on Software Engineering*, vol. 38, no. 1 (January/February 2012).
- J12** **Nenad Medvidović**, Hossein Tajalli, Joshua Garcia, Yuriy Brun, Ivo Krka, and George Edwards. Engineering Heterogeneous Robotics Systems: A Software Architecture-Based Approach. *IEEE Computer*, vol. 44, no. 5 (May 2011).
- J13** Sam Malek, George Edwards, Yuriy Brun, Hossein Tajalli, Joshua Garcia, Ivo Krka, **Nenad Medvidović**, Marija Mikic-Rakic, and Gaurav Sukhatme. An Architecture-Driven Software Mobility Framework. *Journal of Systems and Software*, special issue on Software Architecture and Mobility, vol. 83, no. 6 (June 2010).
- J14** **Nenad Medvidović** and George Edwards. Software Architecture and Mobility: A Roadmap. *Journal of Systems and Software*, special issue on Software Architecture and Mobility, vol. 83, no. 6 (June 2010).
- J15** David Woollard, **Nenad Medvidović**, Yolanda Gil, and Chris A. Mattmann. Scientific Software as Workflows: From Discovery to Distribution. *IEEE Software*, special issue on Developing Scientific Software, vol. 25, no. 4 (July/August, 2008).
- J16** George Edwards, Chiyong Seo, and **Nenad Medvidović**. Model Interpreter Frameworks: A Foundation for the Analysis of Domain-Specific Software Architectures. *Journal of Universal Computer Science*, special issue on Software Components, Architectures, and Reuse, volume 14, number 8 (2008).
- J17** **Nenad Medvidović**, Eric M. Dashofy, and Richard N. Taylor. Moving Architectural Description from Under the Technology Lamppost. *Journal of Information and Software Technology*, vol. 49, no. 1 (January 2007).
- J18** **Nenad Medvidović**, Rene Krikhaar, Robert Nord, and Judith Stafford. Understanding the Past, Improving the Present, and Mapping out the Future of Software Architecture. *Journal of Systems and Software*, vol. 79, no. 12 (December 2006).
- J19** **Nenad Medvidović** and Vladimir Jakobac. Using Software Evolution to Focus Architectural Recovery. *Journal of Automated Software Engineering*, vol. 13, no. 2 (April 2006).
- J20** Sam Malek, Marija Mikic-Rakic, and **Nenad Medvidović**. A Style-Aware Architectural Middleware for Resource-Constrained, Distributed Systems. *IEEE Transactions on Software Engineering*, vol. 31, no. 3 (March 2005).

- J21** Roshanak Roshandel, Andre van der Hoek, Marija Mikic-Rakic, and **Nenad Medvidović**. Mae - A System Model and Environment for Managing Architectural Evolution. *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 2 (April 2004).
- J22** Paul Gruenbacher, Alexander Egyed, and **Nenad Medvidović**. Reconciling Software Requirements and Architectures with Intermediate Models. *Springer Journal of Software and System Modeling*, vol. 3, no. 3, pages 235-253 (August 2004).
- J23** **Nenad Medvidović**, Paul Gruenbacher, Alexander Egyed, and Barry W. Boehm. Bridging Models across the Software Lifecycle. *Journal of Systems and Software*, vol. 68, no. 3 (December 2003).
- J24** **Nenad Medvidović**, Marija Mikic-Rakic, Nikunj Mehta, and Sam Malek. Software Architectural Support for Handheld Computing. *IEEE Computer*, special issue on Handheld Computing, vol. 36, no. 9 (September 2003).
- J25** **Nenad Medvidović**, Eric M. Dashofy, and Richard N. Taylor. The Role of Middleware in Architecture-Based Software Development. *International Journal of Software Engineering and Knowledge Engineering*, vol. 13, no. 4 (August 2003).
- J26** **Nenad Medvidović**, David S. Rosenblum, David F. Redmiles, and Jason E. Robbins. Modeling Software Architectures in the Unified Modeling Language. *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 1 (January 2002).
- J27** Alexander Egyed, **Nenad Medvidović**, and Cristina Gacek. A Component-Based Perspective on Software Mismatch Detection and Resolution. *IEE Proceedings - Software Engineering*, vol. 147, no. 6 (December 2000).
- J28** **Nenad Medvidović** and Richard N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering*, vol. 26, no. 1 (January 2000).
Reprinted in *Rational Developer Network: Seminal Papers on Software Architecture*, Rational Software Corp. (July 2001). Named most widely cited software engineering paper published in 2000 by the *Journal of Information and Software Technology* (January 2007). Selected Most Influential Paper of IEEE TSE's 3rd decade (March 2025).
- J29** Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, **Nenad Medvidović**, Alex Quilici, David S. Rosenblum, and Alexander L. Wolf. An Architecture-Based Approach to Self-Adaptive Software. *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 3 (May/June 1999).
- J30** **Nenad Medvidović** and Richard N. Taylor. Exploiting Architectural Style to Develop a Family of Applications. *IEE Proceedings - Software Engineering*, vol. 144, no. 5-6 (October-December 1997).
- J31** Richard N. Taylor, **Nenad Medvidović**, Kenneth M. Anderson, E. James Whitehead, Jr., Jason E. Robbins, Kari A. Nies, Peyman Oreizy, and Deborah L. Dubrow. A Component- and Message-Based Architectural Style for GUI Software. *IEEE Transactions on Software Engineering*, vol. 22, no. 6 (June 1996).

Refereed Conference Publications

- C1** Satyaki Das, Syeda Tasnim Fabiha, Saad Shafiq, and **Nenad Medvidović**. Are We Learning the Right Features? A Framework for Evaluating DL-Based Software Vulnerability Detection Solutions. In Proceedings of the 47th IEEE/ACM International Conference on Software Engineering (ICSE 2025), Ottawa, Canada, April-May 2025.
- C2** Sadra Sabouri, Philipp Eibl, Xinyi Zhou, Morteza Ziyadi, **Nenad Medvidović**, Lars Lindemann, and Souti Chattopadhyay. Trust Dynamics in AI-Assisted Development: Definitions, Factors, and Implications. In Proceedings of the 47th IEEE/ACM International Conference on Software Engineering (ICSE 2025), Ottawa, Canada, April-May 2025.
- C3** Adriana Sejfia, Satyaki Das, Saad Shafiq, and **Nenad Medvidović**. Toward Improved Deep Learning-based Vulnerability Detection. In Proceedings of the 46th IEEE/ACM International Conference on Software Engineering (ICSE 2024), Lisbon, Portugal, April 2024.
- C4** Yutian Yan, Yunhui Zheng, Xinyue Liu, **Nenad Medvidović**, and Weihang Wang. AdHere: Automated Detection and Repair of Intrusive Ads. In Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE 2023), Melbourne, Australia, May 2023.
- C5** Saghar Talebipour, Hyojae Park, Kesina Baral, Leon Yee, Safwat Ali, Kevin Moran, Yuriy Brun, **Nenad Medvidović**, and Yixue Zhao. AVGUST: A tool for Generating Usage-Based Tests from Videos of App Executions. In Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE 2023), Demonstrations Track, Melbourne, Australia, May 2023.
- C6** Yixue Zhao, Saghar Talebipour, Kesina Baral, Hyojae Park, Leon Yee, Safwat Ali Khan, Yuriy Brun, **Nenad Medvidović**, and Kevin Moran. AVGUST: Automating Usage-Based Test Generation from Videos of App Executions. In Proceedings of the 30th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022), Singapore, November 2022.
- C7** Saghar Talebipour, Yixue Zhao, Luka Dojcilovic, Chenggang Li, and **Nenad Medvidović**. UI Test Migration Across Mobile Platforms. In Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE 2021), November 2021.
- C8** Adriana Sejfia, Yixue Zhao, and **Nenad Medvidović**. Identifying Casualty Changes in Software Patches. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021), September 2021.
- C9** Yixue Zhao, Siwei Yin, Adriana Sejfia, Marcelo Schmitt Laser, Haoyu Wang, and **Nenad Medvidović**. Assessing the Feasibility of Web-Request Prediction Models on Mobile Platforms. In Proceedings of the 8th International Conference on Mobile Software Engineering and Systems (MobileSoft 2021), May 2021.
- C10** Marcelo Schmitt Laser, Duc Minh Le, Joshua Garcia and **Nenad Medvidović**. Architectural Archipelagos: Technical Debt in Long-Lived Software Research Platforms. In Proceedings of the 4th International Conference on Technical Debt (TechDebt 2021), May 2021.

- C11** Joshua Garcia, Mehdi Mirakhorli, Lu Xiao, Yutong Zhao, Ibrahim Mujhid, Khoi Pham, Ahmet Okutan, Sam Malek, Rick Kazman, Yuanfang Cai, and **Nenad Medvidović**. Constructing a Shared Infrastructure for Software Architecture Analysis and Maintenance. In Proceedings of the IEEE 18th International Conference on Software Architecture (ICSA 2021), March 2021.
- C12** Duc Minh Le, Suhrid Karthik, Marcelo Schmitt Laser, and **Nenad Medvidović**. Architectural Decay as Predictor of Issue-and Change-Proneness. In Proceedings of the IEEE 18th International Conference on Software Architecture (ICSA 2021), March 2021.
- C13** Arman Shahbazian, Suhrid Karthik, Yuriy Brun, and **Nenad Medvidović**. eQual: Informing Early Design Decisions. In Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020), Sacramento, CA, November 2020.
- C14** Yixue Zhao, Justin Chen, Adriana Sejfia, Marcelo Schmitt Laser, Jie Zhang, Federica Sarro, Mark Harman, and **Nenad Medvidović**. FrUITeR - A Framework for Evaluating UI Test Reuse. In Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020), Sacramento, CA, November 2020.
- C15** Gvozden Marinkovic, Nikola Lukic, and **Nenad Medvidović**. Online Sports Betting Through the Prism of Software Engineering. In Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020), Industrial Track, Sacramento, CA, November 2020.
- C16** Marcelo Schmitt Laser, Duc Minh Le, Joshua Garcia, and **Nenad Medvidović**. ARCADE: An Extensible Workbench for Architecture Recovery, Change, and Decay Evaluation. In Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020), Tool Demonstrations Track, Sacramento, CA, November 2020.
- C17** Nikola Lukic, Saghar Talebipour, and **Nenad Medvidović**. Remote Control of iOS Devices via Accessibility Features. In Proceedings of the Workshop on Forming an Ecosystem Around Software Transformation (FEAST 2020), November 2020.
- C18** Nikola Lukic, Saghar Talebipour, and **Nenad Medvidović**. AirMochi - A Tool for Remotely Controlling iOS Devices. In Proceedings of the 35th International Conference on Automated Software Engineering (ASE 2020), Melbourne, Australia, September 2020.
- C19** Adriana Sejfia and **Nenad Medvidović**. Strategies for Pattern-Based Detection of Architecturally-Relevant Software Vulnerabilities (Don't Try This at Home - It Doesn't Work). In Proceedings of the 4th IEEE International Conference on Software Architecture (ICSA 2020), Salvador, Brazil, March 2020.
- C20** Yixue Zhao and **Nenad Medvidović**. A Microservice Architecture for Online Mobile App Optimization. In Proceedings of the 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft 2019), Montreal, Canada, 2019.
- C21** Suhrid Karthik and **Nenad Medvidović**. Automatic Detection of Latent Component Relationships from Online Q&A Sites. In Proceedings of the 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE 2019), Montreal, Canada, May 2019.

- C22** Yixue Zhao, Paul Wat, Marcelo Laser, and **Nenad Medvidović**. Empirically Assessing Opportunities for Prefetching and Caching in Mobile Apps. In Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2018), Montpellier, France, September 2018.
- C23** Daye Nam, Youn Kyu Lee, and **Nenad Medvidović**. EVA: A Tool for Visualizing Software Architectural Evolution. In Proceedings of the 40th International Conference on Software Engineering (ICSE 2018), Tool Demonstrations Track, Gothenburg, Sweden, May 2018.
- C24** Yixue Zhao, Marcelo Laser, Yingjun Lyu, and **Nenad Medvidović**. Leveraging Program Analysis to Reduce User-Perceived Latency in Mobile Applications. In Proceedings of the 40th International Conference on Software Engineering (ICSE 2018), Gothenburg, Sweden, May 2018.
- C25** Arman Shahbazian, Daye Nam, and **Nenad Medvidović**. Toward Predicting Architectural Significance of Implementation Issues. In Proceedings of the 15th IEEE/ACM International Conference on Mining Software Repositories (MSR 2018), Gothenburg, Sweden, May 2018.
- C26** Duc Minh Le, Daniel Link, Arman Shahbazian, and **Nenad Medvidović**. An Empirical Study of Architectural Decay in Open-Source Software. In Proceedings of the 15th IEEE International Conference on Software Architecture (ICSA 2018), Seattle, WA, April 2018.
Winner of the ICSA 2018 Best Paper Award
- C27** Arman Shahbazian, Youn Kyu Lee, and **Nenad Medvidović**. Recovering Architectural Design Decisions. In Proceedings of the 15th IEEE International Conference on Software Architecture (ICSA 2018), Seattle, WA, April/May 2018.
- C28** Youn Kyu Lee, Peera Yoodde, Arman Shahbazian, Daye Nam, and **Nenad Medvidović**. SEALANT: A Detection and Visualization Tool for Inter-app Security Vulnerabilities in Android. In Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017), Urbana-Champaign, USA, November 2017.
Winner of the ASE 2017 Best Tool-Paper Award
- C29** Youn Kyu Lee, Jae Young Bang, Gholamreza Safi, Arman Shahbazian, Yixue Zhao, and **Nenad Medvidović**. A SEALANT for Inter-App Security Holes in Android. In Proceedings of the 39th International Conference on Software Engineering (ICSE 2017), Buenos Aires, Argentina, May 2017.
- C30** Jae young Bang, Yuriy Brun, and **Nenad Medvidović**. Continuous Analysis of Collaborative Design. In Proceedings of the 14th IEEE International Conference on Software Architecture (ICSA 2017), Gothenburg, Sweden, April 2017.
Winner of the ICSA 2017 Best Paper Award
- C31** Martin Robillard and **Nenad Medvidović**. Disseminating Architectural Knowledge on Open-Source Projects. In Proceedings of the 38th International Conference on Software Engineering (ICSE 2016), Austin, TX, May 2016.
- C32** Arman Shahbazian, George Edwards, and **Nenad Medvidović**. An End-to-End Domain-Specific Modeling and Analysis Platform. In Proceedings of the 8th International Workshop on Modeling in Software Engineering (MISE 2016), Austin, TX, May 2016.

- C33** Michael Langhammer, Arman Shahbazian, **Nenad Medvidović**, and Ralf Reussner. Automated Extraction of Rich Software Models from Limited System Information. In Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture (WICSA 2016), Venice, Italy, April 2016.
- C34** Duc Minh Le, Carlos Carrillo, Rafael Capilla, and **Nenad Medvidović**. Relating Architectural Decay and Sustainability of Software Systems. In Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture (WICSA 2016), Venice, Italy, April 2016.
- C35** Gholamreza Safi, Arman Shahbazian, William G.J. Halfond, and **Nenad Medvidović**. Detecting Event Anomalies in Event-Based Systems. In Proceedings of the 10th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2015), Bergamo, Italy, September 2015.
- C36** Duc Minh Le, Pooyan Behnamghader, Joshua Garcia, Daniel Link, Arman Shahbazian, and **Nenad Medvidović**. An Empirical Study of Architectural Change in Open-Source Software Systems. In Proceedings of the 12th Working Conference on Mining Software Repositories (MSR 2015), Florence, Italy, May 2015.
Fast-tracked for journal publication as one of the best papers of the conference.
- C37** Thibaud Lutellier, Devin Chollak, Joshua Garcia, Lin Tan, Derek Rayside, **Nenad Medvidović**, and Robert Kroeger. Comparing Software Architecture Recovery Techniques Using Accurate Dependencies. In Proceedings of the 37th International Conference on Software Engineering (ICSE 2015), Software Engineering in Practice Track, Florence, Italy, May 2015.
- C38** Jae young Bang and **Nenad Medvidović**. Proactive Detection of Higher-Order Software Design Conflicts. In Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture (WICSA 2015), Montreal, Canada, May 2015.
- C39** Ivo Krka, Yuriy Brun, and **Nenad Medvidović**. Automatic Mining of Specifications from Invocation Traces and Method Invariants. In Proceedings of the 22nd ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 2014), Hong Kong, November 2014.
- C40** Youn Kyu Lee, Jae young Bang, Joshua Garcia, and **Nenad Medvidović**. ViVA: A Visualization and Analysis Tool for Distributed Event-Based Systems. In Proceedings of the 36th International Conference on Software Engineering (ICSE 2014), Formal Demonstration Track, Hyderabad, India, June 2014.
- C41** Ivo Krka, Nicolas D'Ippolito, **Nenad Medvidović**, and Sebastian Uchitel. Revisiting Compatibility of Input-Output Modal Transition Systems. In Proceedings of the 19th International Symposium on Formal Methods (FM 2014), Singapore, May 2014.
- C42** Ivo Krka and **Nenad Medvidović**. Component-Aware Triggered Scenarios. In Proceedings of the 11th Working IEEE/IFIP Conference on Software Architecture (WICSA 2014), Sydney, Australia, April 2014.

- C43** Joshua Garcia, Igor Ivkovic, and **Nenad Medvidović**. A Comparative Analysis of Software Architecture Recovery Techniques. In Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE 2013), Palo Alto, CA, November 2013.
- C44** Joshua Garcia, Daniel Popescu, Gholamreza Safi, William G.J. Halfond, and **Nenad Medvidović**. Identifying Message Flow in Distributed Event-Based Systems. In Proceedings of the 9th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2013), St. Petersburg, Russia, August 2013.
- C45** Ivo Krka and **Nenad Medvidović**. Distributing Refinements of a System-Level Partial Behavior Model. In Proceedings of the 21st International Conference on Requirements Engineering, Rio de Janeiro, Brazil, July 2013.
- C46** Joshua Garcia, Ivo Krka, Chris Mattmann, and **Nenad Medvidović**. Obtaining Ground-Truth Software Architectures. In Proceedings of the 35th International Conference on Software Engineering (ICSE 2013), Software Engineering in Practice Track, San Francisco, CA, May 2013.
- C47** Jae young Bang, Ivo Krka, **Nenad Medvidović**, Naveen Kulkarni, and Srinivas Padmanabhuni. How Software Architects Collaborate: Insights from Collaborative Software Design in Practice. In Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2013), San Francisco, CA, May 2013.
- C48** Ran Mo, Joshua Garcia, Yuanfang Cai, and **Nenad Medvidović**. Mapping Architectural Smells into Dependency Models. In Proceedings of the 4th International Workshop on Managing Technical Debt (MTD 2013), San Francisco, CA, May 2013.
- C49** Christoph Dorn, George Edwards, and **Nenad Medvidović**. Analyzing Design Tradeoffs in Large-scale Socio-Technical Systems through Simulation of Dynamic Collaboration Patterns. In Proceedings of the 20th International Conference on Cooperative Information Systems (CoopIS 2012), Rome, Italy, September 2012.
- C50** Joshua Garcia, Ivo Krka, **Nenad Medvidović**, and Chris Douglas. A Framework for Obtaining the Ground-Truth in Architectural Recovery. In Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA 2012), Helsinki, Finland, August 2012.
- C51** George Edwards, Yuriy Brun, and **Nenad Medvidović**. Automated Analysis and Code Generation for Domain-Specific Models. In Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA 2012), Helsinki, Finland, August 2012.
- C52** Daniel Popescu, Joshua Garcia, Kevin Bierhoff, and **Nenad Medvidović**. Impact Analysis for Distributed Event-Based Systems. In Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems (DEBS 2012), Berlin, Germany, July 2012.
- C53** Ivo Krka and **Nenad Medvidović**. Revisiting Modal Interface Automata. In Proceedings of the Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA), Zurich, Switzerland, June 2012.

- C54** Hossein Tajalli and **Nenad Medvidović**. A Reference Architecture for Integrated Development and Run-Time Environments. In Proceedings of the 2nd Workshop on Developing Tools as Plug-ins (TOPI), San Francisco, CA, June 2012.
- C55** Yuriy Brun and **Nenad Medvidović**. Keeping Data Private while Computing in the Cloud. In Proceedings of the 5th International Conference on Cloud Computing (IEEE CLOUD 2012), Honolulu, HI, June 2012.
- C56** Leslie Cheung, Ivo Krka, Leana Golubchik, and **Nenad Medvidović**. Architecture-Level Reliability Prediction for Concurrent Systems. In Proceedings of the 3rd International Conference on Performance Engineering (ICPE 2012), Boston, MA, April 2012.
- C57** Isela Macia Bertrán, Joshua Garcia, Daniel Popescu, Alessandro Garcia, **Nenad Medvidović**, and Arndt Von Staa. Are Automatically-Detected Code Anomalies Relevant to Architectural Modularity? An Exploratory Analysis of Evolving Systems. In Proceedings of the 11th International Conference on Aspect-Oriented Software Development, Potsdam, Germany, March 2012.
- C58** Jae young Bang, Daniel Popescu, and **Nenad Medvidović**. Enabling Workspace Awareness for Collaborative Software Modeling. In Proceedings of the Future of Collaborative Software Development Workshop, Seattle, WA, February 2012.
- C59** George Edwards, Yuriy Brun, and **Nenad Medvidović**. Isomorphism in Model Tools and Editors. In Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, November 2011.
- C60** Joshua Garcia, Daniel Popescu, Chris A. Mattmann, **Nenad Medvidović**, and Yuanfang Cai. Enhancing Architectural Recovery Using Concerns. In Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, November 2011.
- C61** Isela Macia Bertrán, Alessandro Garcia, Arndt von Staa, Joshua Garcia, and **Nenad Medvidović**. On the Impact of Aspect-Oriented Code Smells on Architecture Modularity: An Exploratory Study. In Proceedings of the 5th Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS 2011), Sao Paulo, Brazil, September 2011.
- C62** Yuriy Brun, George Edwards, Jae young Bang, and **Nenad Medvidović**. Smart Redundancy for Distributed Computation. In Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS), Minneapolis, MN, June 2011.
- C63** Hossein Tajalli, Joshua Garcia, George Edwards, and **Nenad Medvidović**. PLASMA: A Plan-based Layered Architecture for Software Model-driven Adaptation. In Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010), Antwerp, Belgium, September 2010.
- C64** David Woollard, Chris A. Mattmann, Daniel Popescu, and **Nenad Medvidović**. KADRE: Domain-Specific Architectural Recovery for Scientific Software Systems. In Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010), Antwerp, Belgium, September 2010.

- C65** Jae young Bang, Daniel Popescu, George Edwards, **Nenad Medvidović**, Naveen Kulkarni, Girish M. Rama, and Srinivas Padmanabhuni. CoDesign - A Highly Extensible Collaborative Software Modeling Framework. In Proceedings of the 32nd International Conference on Software Engineering (ICSE 2010), Formal Demonstration Track, Cape Town, South Africa, May 2010.
- C66** Ivo Krka, Yuriy Brun, Daniel Popescu, Joshua Garcia, and **Nenad Medvidović**. Using Dynamic Execution Traces and Program Invariants to Enhance Behavioral Model Inference. In Proceedings of the 32nd International Conference on Software Engineering (ICSE 2010), New Ideas and Emerging Results, Cape Town, South Africa, May 2010.
- C67** Ivo Krka, Leana Golubchik, and **Nenad Medvidović**. Probabilistic Automata for Architecture-Based Reliability Assessment. In Proceedings of the Workshop on Quantitative Stochastic Models in the Verification and Design of Software Systems (QUOVADIS), Cape Town, South Africa, May 2010.
- C68** Leslie Cheung, Leana Golubchik, and **Nenad Medvidović**. SHARP: A Scalable Approach to Architecture-Level Reliability Prediction of Concurrent Systems. In Proceedings of the Workshop on Quantitative Stochastic Models in the Verification and Design of Software Systems (QUOVADIS), Cape Town, South Africa, May 2010.
- C69** T. S. Mohan, **Nenad Medvidović**, and Chris A. Mattmann. Leveraging Domain-Specific Software Architectures for Classifying Cloud Service Abstractions. In Proceedings of the Cloud Futures 2010 Workshop, Redmond, WA, April 2010.
- C70** Susan Ferreira, Ricardo Valerdi, **Nenad Medvidović**, John Hess, Indira Deonandan, Tsoline Mikaelian, Tom Tenorio, Filiberto Macias, and Gayle Shull. Unmanned and Autonomous Systems of Systems Test and Evaluation: Challenges and Opportunities. In Proceedings of the 2010 IEEE International Systems Conference (SysCon), San Diego, CA, April 2010.
- C71** Chris Mattmann, Joshua Garcia, Ivo Krka, Daniel Popescu, and **Nenad Medvidović**. The Anatomy and Physiology of the Grid Revisited. In Proceedings of the Joint 8th Working IEEE/IFIP Conference on Software Architecture (WICSA-8) and the 3rd European Conference on Software Architecture (ECSA-3), Cambridge, UK, September 2009.
- C72** Richard N. Taylor, **Nenad Medvidović**, and Peyman Oreizy. Architectural Styles for Runtime Software Adaptation In Proceedings of the Joint 8th Working IEEE/IFIP Conference on Software Architecture (WICSA-8) and the 3rd European Conference on Software Architecture (ECSA-3), Cambridge, UK, September 2009.
- C73** Ivo Krka, Yuriy Brun, George Edwards, and **Nenad Medvidović**. Synthesizing Partial Component-Level Behavior Models from System Specifications. In Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE09), Amsterdam, the Netherlands, August 2009.
- C74** Chiyong Seo, George Edwards, Daniel Popescu, Sam Malek, and **Nenad Medvidović**. A Framework for Estimating the Energy Consumption Induced by a Distributed System's Architectural Style. In Proceedings of the 8th Workshop on Specification and Verification of Component-Based Systems (SAVCBS), Amsterdam, the Netherlands, August 2009.

- C75** Joshua Garcia, Daniel Popescu, George Edwards, and **Nenad Medvidović**. Toward a Catalogue of Architectural Bad Smells. In Proceedings of the 5th International Conference on the Quality of Software Architectures (QoSA), East Stroudsburg, PA, June 2009.
- C76** Donna Spruijt-Metz, Ming Li, Gautam Thatte, Gaurav Sukhatme, Murali Annavaram, Sabyasachi Ghosh, Viktor Rozgic, Urbashi Mitra, **Nenad Medvidović**, Britni Belcher, and Shrikanth Narayanan. Differentiating Physical Activity Modalities in Youth Using Heartbeat Waveform Shape and Differences between Adjacent Waveforms. In Proceedings of the 7th International Conference on Diet and Activity Methods (ICDAM 7), Washington, DC, June 2009.
- C77** Yuriy Brun and **Nenad Medvidović**. Crystal-Growth-Inspired Algorithms for Computational Grids. In Proceedings of the Workshop on Bio-Inspired Algorithms for Distributed Systems (BADS09), Barcelona, Spain, June 2009.
- C78** Ivo Krka, George Edwards, Yuriy Brun, and **Nenad Medvidović**. From System Specifications to Component Behavioral Models. In Proceedings of the 31st International Conference on Software Engineering (ICSE 2009), New Ideas and Emerging Results Track, Vancouver, Canada, May 2009.
- C79** Daniel Popescu, Joshua Garcia, and **Nenad Medvidović**. Enabling More Precise Dependency Analysis in Event-Based Systems. In Proceedings of the 17th International Conference on Program Comprehension (ICPC 2009), Vancouver, Canada, May 2009.
Winner of the ICPC 2009 Best Poster Award
- C80** George Edwards, Joshua Garcia, Hossein Tajalli, Daniel Popescu, **Nenad Medvidović**, Gaurav Sukhatme, and Brad Petrus. Architecture-Driven Self-Adaptation and Self-Management in Robotics Systems. In Proceedings of the 2009 ACM/IEEE International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Vancouver, Canada, May 2009.
- C81** David Woollard, Chris Mattmann, and **Nenad Medvidović**. Injecting Software Architectural Constraints into Legacy Scientific Applications. In Proceedings of the ICSE 2009 Workshop on Software Engineering for Computational Science and Engineering, Vancouver, Canada, May 23, 2009.
- C82** Joshua Garcia, Daniel Popescu, George Edwards and **Nenad Medvidović**. Identifying Architectural Bad Smells. In Proceedings of the 13th European Conference on Software Maintenance and Reengineering (CSMR 2009), Kaiserslautern, Germany, March 2009.
- C83** Murali Annavaram, **Nenad Medvidović**, Urbashi Mitra, Shrikanth Narayanan, Donna Spruijt-Metz, Gaurav Sukhatme, Zhaoshi Meng, Shi Qiu, Rohit Kumar, and Gautam Thatte. Multimodal Sensing for Pediatric Obesity Applications. In Proceedings of the International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems (UrbanSense08), Raleigh, NC, November 4, 2008.
- C84** Chiyoung Seo, Sam Malek, and **Nenad Medvidović**. Component-Level Energy Consumption Estimation for Distributed Java-Based Software Systems. In Proceedings of the 11th International Symposium on Component Based Software Engineering (CBSE 2008), Karlsruhe, Germany, October 2008.

- C85** George Edwards and **Nenad Medvidović**. A Methodology and Framework for Creating Domain-Specific Development Infrastructures. In Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), L'Aquila, Italy, September 2008.
- C86** Daniel Popescu and **Nenad Medvidović**. Ensuring Architectural Conformance in Message- Based Systems. In Proceedings of the DSN 2008 Workshop on Architecting Dependable Systems (WADS 2008), Anchorage, AK, USA, June 2008.
- C87** Ivo Krka, Leslie Cheung, George Edwards, Leana Golubchik, and **Nenad Medvidović**. Architecture-Based Software Reliability Estimation: Problem Space, Challenges, and Strategies. In Proceedings of the DSN 2008 Workshop on Architecting Dependable Systems (WADS 2008), Anchorage, AK, USA, June 2008.
- C88** Leslie Cheung, Roshanak Roshandel, **Nenad Medvidović**, and Leana Golubchik. Early Prediction of Software Component Reliability. In Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 2008.
- C89** Marija Mikic-Rakic, Sam Malek, and **Nenad Medvidović**. Architecture-Driven Software Mobility in Support of QoS Requirements. In Proceedings of the International Workshop on Software Architectures and Mobility (SAM 2008), Leipzig, Germany, May 2008.
- C90** Chris Mattmann, David Woollard, and **Nenad Medvidović**. Exploiting Connector Knowledge To Efficiently Disseminate Highly Voluminous Data Sets. In Proceedings of the 3rd Workshop on SHaring and Reusing architectural Knowledge (SHARK 2008), Leipzig, Germany, May 2008.
- C91** Chiyoung Seo, Sam Malek, and **Nenad Medvidović**. Estimating the Energy Consumption in Pervasive Java-Based Systems. In Proceedings of the 6th IEEE International Conference on Pervasive Computing and Communications, Hong Kong, March 2008.
- C92** Chiyoung Seo, George Edwards, Sam Malek, and **Nenad Medvidović**. A Framework for Estimating the Impact of a Distributed Software System's Architectural Style on its Energy Consumption. In Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture (WICSA 2008), Vancouver, BC, Canada, February 2008.
- C93** George Edwards, Chiyoung Seo, Daniel Popescu, Sam Malek, and **Nenad Medvidović**. Self-* Software Architectures and Component Middleware in Pervasive Environments. In Proceedings of the 5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2007), Newport Beach, CA, November 2007.
- C94** Chiyoung Seo, Sam Malek, and **Nenad Medvidović**. An Energy Consumption Framework for Distributed Java-Based Systems. In Proceedings of the 22nd International Conference on Automated Software Engineering (ASE 2007), Atlanta, GA, November 2007.
- C95** Daniel Popescu, Spencer Rugaber, **Nenad Medvidović**, and Daniel M. Berry. Improving the Quality of Requirements Specifications via Automatically Created Object-Oriented Models. In Proceeding of the 14th Monterey Workshop on Requirements Analysis, Monterey, CA, September 2007.

- C96** Yuriy Brun and **Nenad Medvidović**. Fault and Adversary Tolerance as an Emergent Property of Distributed Systems' Software Architectures. In Proceedings of the 2nd International Workshop on Engineering Fault Tolerant Systems (EFTS 2007), Dubrovnik, Croatia, September 2007.
- C97** **Nenad Medvidović** and Sam Malek. Software Deployment Architecture and Quality-of-Service in Pervasive Environments. In Proceedings of the International Workshop on the Engineering of Software Services for Pervasive Environments (ESSPE 2007), Dubrovnik, Croatia, September 2007.
- C98** George Edwards, Chiyong Seo, and **Nenad Medvidović**. Construction of Analytic Frameworks for Component-Based Architectures. In Proceedings of the Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS 2007), Campinas, Brazil, August 2007.
- C99** Vito Perrone, Chris A. Mattmann, Sean Kelly, Daniel J. Crichton, Anthony Finkelstein, and **Nenad Medvidović**. A Reference Framework for Requirements and Architecture in Biomedical Grid Systems. In Proceedings of the IEEE International Conference on Information Reuse and Integration (IEEE IRI-2007), Las Vegas, NV, August 2007.
- C100** Roshanak Roshandel, **Nenad Medvidović**, and Leana Golubchik. A Bayesian Model for Predicting Reliability of Software Systems at the Architectural Level. In Proceedings of the 3rd International Conference on Quality of Software Architectures (QoSA 2007), Boston, MA, July 2007.
- C101** Roshanak Roshandel and **Nenad Medvidović**. A Classification of Architectural Reliability Models. In Proceedings of the International Workshop on Software Architecture Research and Practice, Silicon Valley, CA, July 2007.
- C102** Yuriy Brun and **Nenad Medvidović**. An Architectural Style for Solving Computationally Intensive Problems on Large Networks. In Proceedings of the 2nd Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2007), Minneapolis, MN, May 2007.
- Winner of the SEAMS 2007 Most Influential Paper Award, awarded at SEAMS 2020
- C103** Sam Malek, Chiyong Seo, Sharmila Ravula, Brad Petrus, and **Nenad Medvidović**. Reconceptualizing a Family of Heterogeneous Embedded Systems via Explicit Architectural Support. In Proceedings of the 29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, May 2007.
- C104** Chiyong Seo, Sam Malek, George Edwards, Daniel Popescu, **Nenad Medvidović**, Brad Petrus, and Sharmila Ravula. Exploring the Role of Software Architecture in Dynamic and Fault-Tolerant Pervasive Systems. In Proceedings of the Workshop on Software Engineering of Pervasive Computing Applications, Systems, and Environments (SEPCASE 2007), Minneapolis, MN, May 2007.
- C105** Chris A. Mattmann, David Woollard, **Nenad Medvidović**, and Reza Mahjourian. Software Connector Classification and Selection for Data-intensive Systems. In Proceedings of the 2nd Workshop on Incorporating COTS Software into Software Systems: Tools and Techniques (IWICSS 2007), Minneapolis, MN, May 2007.

- C106** George Edwards, Sam Malek, and **Nenad Medvidović**. Scenario-Driven Dynamic Analysis of Distributed Architecture. In Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering (FASE 2007), Braga, Portugal, March- April 2007.
- C107** Chris A. Mattmann and **Nenad Medvidović**. The GridLite DREAM: Bringing the Grid to Your Pocket. In Proceedings of the 12th Monterey Workshop, LNCS 4322, Springer Verlag, February 2007.
- C108** Jesal Bhuta, Chris A. Mattmann, **Nenad Medvidović**, and Barry Boehm. A Framework for the Assessment and Selection of Software Components and Connectors in COTS-based Architectures. In Proceedings of the 2007 IEEE/IFIP Working Conference on Software Architecture (WICSA-6), Mumbai, India, January 2007.
- C109** Sam Malek, Chiyong Seo, Sharmila Ravula, Brad Petrus, and **Nenad Medvidović**. Providing Middleware-Level Facilities to Support Architecture-Based Development of Software Systems in Pervasive Environments. In Proceedings of the 4th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2006), Melbourne, Australia, November 27 - December 2006.
- C110** Sam Malek, Chiyong Seo, and **Nenad Medvidović**. Tailoring an Architectural Middleware Platform to a Heterogeneous Embedded Environment. In Proceedings of the 6th International Workshop on Software Engineering and Middleware (SEM 2006), Portland, Oregon, November 2006.
- C111** Marija Mikic-Rakic and **Nenad Medvidović**. A Classification of Disconnected Operation Techniques. In Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), Dubrovnik, Croatia, August 29 - September 2006.
- C112** Chris A. Mattmann, Daniel J. Crichton, **Nenad Medvidović**, and Steven Hughes. A Software Architecture-Based Framework for Highly Distributed and Data Intensive Scientific Applications. In Proceedings of the 28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 2006.
- C113** Roshanak Roshandel, Somo Banerjee, Leslie Cheung, **Nenad Medvidović**, and Leana Golubchik. Estimating Software Component Reliability by Leveraging Architectural Models. In Proceedings of the 28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 2006.
- C114** David Woollard and **Nenad Medvidović**. An Architectural Style for High-Performance Asymmetrical Parallel Computations. In Proceedings of the 28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 2006.
- C115** Sam Malek, Marija Mikic-Rakic, and **Nenad Medvidović**. A Decentralized Redeployment Algorithm for Improving the Availability of Distributed Systems. In Proceedings of the 3rd International Working Conference on Component Deployment (CD 2005), Grenoble, France, November 2005.
- C116** Marija Mikic-Rakic, Sam Malek, and **Nenad Medvidović**. Improving Availability in Large, Distributed Component-Based Systems via Redeployment. In Proceedings of the 3rd International Working Conference on Component Deployment (CD 2005), Grenoble, France, November 2005.

- C117** Somo Banerjee, Chris A. Mattmann, **Nenad Medvidović**, and Leana Golubchik. Leveraging Architectural Models to Inject Trust into Software Systems. In Proceedings of the ICSE 2005 Workshop on Software Engineering for Secure Systems - Building Trustworthy Applications (SESS'05), St. Louis, MO, May 2005.
- C118** Omer Erdem Demir, Prem Devanbu, **Nenad Medvidović**, and Eric Wohlstadter. DISCOA: Architectural Adaptations for Security and QoS. In Proceedings of the ICSE 2005 Workshop on Software Engineering for Secure Systems - Building Trustworthy Applications (SESS'05), St. Louis, MO, May 2005.
- C119** Vladimir Jakobac, **Nenad Medvidović**, and Alexander Egyed. Separating Architectural Concerns to Ease Program Understanding. In Proceedings of the ICSE 2005 Workshop on the Modeling and Analysis of Concerns in Software (MACS), St. Louis, MO, May 2005.
- C120** Chris A. Mattmann, **Nenad Medvidović**, Paul M. Ramirez, and Vladimir Jakobac. Unlocking the Grid. In Proceedings of the 8th International Symposium on Component Based Software Engineering (CBSE-8), St. Louis, MO, May 2005.
- C121** David Woollard, **Nenad Medvidović**, Walter Yamada, and Theodore Berger. ADaPT: An Event-Passing Protocol for Reducing Delivery Costs in Scatter-Gather Parallel Processes. In Proceedings of the Workshop for Patterns in High Performance Computing, Urbana, IL, May 2005.
- C122** Vladimir Jakobac, Alexander Egyed, and **Nenad Medvidović**. Improving System Understanding via Interactive, Tailorable, Source Code Analysis. In Proceedings of the 2005 Conference on Fundamental Approaches to Software Engineering (FASE 2005), Edinburgh, UK, April 2005.
- C123** Chris A. Mattmann, Sam Malek, Nels Beckman, Marija Mikic-Rakic, **Nenad Medvidović**, and Daniel J. Crichton. GLIDE: A Grid-based Light-weight Infrastructure for Data-intensive Environments. In Proceedings of the 2005 European Grid Conference (EGC 2005), Amsterdam, the Netherlands, February 2005.
- C124** Sam Malek, Marija Mikic-Rakic, **Nenad Medvidović**. An Extensible Framework for Autonomic Analysis and Improvement of Distributed Deployment Architectures. In Proceedings of the ACM SIGSOFT Workshop on Self-Managed Systems (WOSS 2004), Newport Beach, CA, October 31 - November 2004.
- C125** Nikunj Mehta, **Nenad Medvidović**, Marjan Sirjani, and Farhad Arbab. Modeling Behavior in Compositions of Software Architectural Primitives. In Proceedings of the 19th IEEE International Conference on Automated Software Engineering (ASE 2004), Linz, Austria, September 2004.
- C126** Barry Boehm, Jesal Bhuta, David Garlan, Eric Gradman, LiGuo Huang, Alexander Lam, Ray Madachy, **Nenad Medvidović**, Kenneth Meyer, Steven Meyers, Gustavo Perez, Kirk Reinholtz, Roshanak Roshandel, and Nicolas Rouquette. Using Testbeds to Accelerate Technology Maturity and Transition: The SCROver Experience. In Proceedings of the ACM-IEEE 2004 International Symposium on Empirical Software Engineering (ISESE 2004), Redondo Beach, CA, August 2004.

- C127** Roshanak Roshandel, Bradley Schmerl, **Nenad Medvidović**, David Garlan, and Dehua Zhang. Understanding Tradeoffs among Different Architectural Modeling Approaches. In Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture (WICSA-4), Oslo, Norway, June 2004.
- C128** Roshanak Roshandel and **Nenad Medvidović**. Toward Architecture-Based Reliability Estimation. In Proceedings of Twin Workshops on Architecting Dependable Systems (WADS 2004), Edinburgh, UK, May 25, 2004 and Florence, Italy, June 2004.
- C129** Marija Mikic-Rakic, Sam Malek, Nels Beckman, and **Nenad Medvidović**. Improving Availability of Distributed Event-Based Systems via Run-Time Monitoring and Analysis. In Proceedings of Twin Workshops on Architecting Dependable Systems (WADS 2004), Edinburgh, UK, May 25, 2004 and Florence, Italy, June 2004.
- C130** Nikunj Mehta, Ramakrishna Soma, and **Nenad Medvidović**. Style-Based Software Architectural Compositions as Domain-Specific Models. In Proceedings of the Workshop on Directions in Software Engineering Environments (WoDiSEE 2004), Edinburgh, UK, May 2004.
- C131** Marija Mikic-Rakic, Sam Malek, Nels Beckman, and **Nenad Medvidović**. A Tailorable Environment for Assessing the Quality of Deployment Architectures in Highly Distributed Settings. In Proceedings of the 2nd International Working Conference on Component Deployment (CD 2004), Edinburgh, UK, May 2004.
- C132** Marija Mikic-Rakic and **Nenad Medvidović**. Software Architectural Support for Disconnected Operation in Highly Distributed Environments. In Proceedings of the 7th International Symposium on Component Based Software Engineering (CBSE-7), Edinburgh, UK, May 2004.
- C133** Marija Mikic-Rakic and **Nenad Medvidović**. Support for Disconnected Operation via Architectural Self-Reconfiguration. In Proceedings of the First International Conference on Autonomic Computing (ICAC-04), New York, NY, May 2004.
- C134** **Nenad Medvidović**, Sam Malek, and Marija Mikic-Rakic. Software Architectures and Embedded Systems. In Proceedings of the Monterey Workshop on Software Engineering for Embedded Systems (SEES 2003), Chicago, IL, September 2003.
- C135** Andre van der Hoek, Ebru Dincel, and **Nenad Medvidović**. Using Service Utilization Metrics to Assess the Structure of Product Line Architectures. In Proceedings of the 9th International Software Metrics Symposium, Sydney, Australia, September 2003.
- C136** Nikunj R. Mehta and **Nenad Medvidović**. Composing Architectural Styles from Architectural Primitives. In Proceedings of the Joint 9th European Software Engineering Conference and 11th ACM Symposium on the Foundations of Software Engineering (ESEC/ FSE 2003), Helsinki, Finland, September 2003.
- C137** Roshanak Roshandel and **Nenad Medvidović**. Modeling Multiple Aspects of Software Components. In Proceedings of the Workshop on Specification and Verification of Component-Based Systems, Helsinki, Finland, September 2003.
- C138** Marija Mikic-Rakic and **Nenad Medvidović**. Adaptable Architectural Middleware for Programming-in-the-Small-and-Many. In Proceedings of the ACM/IFIP/USENIX International Middleware Conference (Middleware'03), Rio de Janeiro, Brazil, June 2003.

- C139** Marija Mikic-Rakic and **Nenad Medvidović**. A Connector-Aware Middleware for Distributed Deployment and Mobility. In Proceedings of the First International Workshop on Mobile Computing Middleware, Providence, RI, May 2003.
- C140** **Nenad Medvidović**, Alexander Egyed, and Paul Gruenbacher. Stemming Architectural Erosion by Coupling Architectural Discovery and Recovery. In Proceedings of the Second International Software Requirements to Architectures Workshop, Portland, OR, May 2003.
- C141** Marija Mikic-Rakic and **Nenad Medvidović**. Toward a Framework for Classifying Disconnected Operation Techniques. In Proceedings of the Second International Workshop on Software Architectures for Dependable Systems, Portland, OR, May 2003.
- C142** Nikunj Mehta and **Nenad Medvidović**. Understanding Software Connector Compatibilities Using A Connector Taxonomy. In Proceedings of the First Workshop on Software Design and Architecture (SoDA'02), Bangalore, India, December 2002.
- C143** Marija Mikic-Rakic, Nikunj Mehta, and **Nenad Medvidović**. Architectural Style Requirements for Self-Healing Systems. In Proceedings of the First International Workshop on Self-Healing Systems (WOSS'02), Charleston, SC, November 2002.
- C144** **Nenad Medvidović**, Nikunj Mehta, and Marija Mikic-Rakic. A Family of Software Architecture Implementation Frameworks. In Proceedings of the 2002 Working IEEE/IFIP Conference on Software Architectures (WICSA-3), Montreal, Canada, August 2002.
- C145** **Nenad Medvidović**. On the Role of Middleware in Architecture-Based Software Development. In Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE), Ischia, Italy, July 2002.
Fast-tracked for journal publication as one of the best papers of the conference.
- C146** Marija Mikic-Rakic and **Nenad Medvidović**. Architecture-Level Support for Software Component Deployment in Resource Constrained Environments. In Proceedings of the IFIP/ACM Working Conference on Component Deployment, Berlin, Germany, June 2002.
- C147** **Nenad Medvidović** and Marija Mikic-Rakic. Programming-in-the-Many: A Software Engineering Paradigm for the 21st Century. In Proceedings of the Workshop on New Visions for Software Design and Productivity. Nashville, TN, December 2001.
- C148** Roshanak Roshandel and **Nenad Medvidović**. Coupling Static and Dynamic Semantics in an Architecture Description Language. In Proceedings of the Working Conference on Complex and Dynamic Systems Architecture (CDSA), Brisbane, Australia, December 2001.
- C149** Ebru Dincel, **Nenad Medvidović**, and Andre van der Hoek. Measuring Product Line Architectures. In Proceedings of the 4th International Workshop on Product Family Engineering (PFE-4), pages 346-352, Bilbao, Spain, October 2001.
- C150** Andre van der Hoek, Marija Mikic-Rakic, Roshanak Roshandel, and **Nenad Medvidović**. Taming Architectural Evolution. In Proceedings on the Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2001), Vienna, Austria, September 2001.
Fast-tracked for journal publication as one of the best papers of the conference.

- C151** Paul Gruenbacher, Alexander Egyed, and **Nenad Medvidović**. Reconciling Software Requirements and Architectures: The CBSP Approach. In Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01), Toronto, Canada, August 2001.
- C152** Lei Ding and **Nenad Medvidović**. Focus: A Light-Weight, Incremental Approach to Software Architecture Recovery and Evolution. In Proceedings of the 2001 Working IEEE/IFIP Conference on Software Architectures (WICSA-2), Amsterdam, the Netherlands, August 2001.
- C153** **Nenad Medvidović**, Paul Gruenbacher, Alexander Egyed, and Barry W. Boehm. Software Model Connectors: Bridging Models across the Software Lifecycle. In Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering (SEKE 2001), Buenos Aires, Argentina, June 2001.
- Fast-tracked for journal publication as one of the best papers of the conference.
- C154** Marija Rakic and **Nenad Medvidović**. Increasing the Confidence in Off-the-Shelf Components: A Software Connector-Based Approach. In Proceedings of the 2001 Symposium on Software Reusability (SSR 2001), Toronto, Canada, May 2001.
- C155** Alexander Egyed and **Nenad Medvidović**. Consistent Architectural Refinement and Evolution Using the Unified Modeling Language. In Proceedings of the Workshop on Describing Software Architecture with UML, Toronto, Canada, May 2001.
- C156** Alexander Egyed, Paul Gruenbacher, and **Nenad Medvidović**. Refinement and Evolution Issues in Bridging Requirements and Architecture - The CBSP Approach. In Proceedings of the First International Software Requirements to Architectures Workshop (STRAW 2001), Toronto, Canada, May 2001.
- C157** Marija Rakic and **Nenad Medvidović**. Runtime Support for Architecture-Level Configuration Management. In Proceedings of the Tenth International Workshop on Software Configuration Management (SCM-10), Toronto, Canada, May 2001.
- C158** **Nenad Medvidović** and Marija Rakic. Exploiting Software Architecture Implementation Infrastructure in Facilitating Component Mobility. In Proceedings of the Workshop on Software Engineering and Mobility, Toronto, Canada, May 2001.
- C159** Nicolas Rouquette, **Nenad Medvidović**, and David Garlan. Dependable Autonomous Systems = knowing well what to do + knowing how to do it well. In Proceedings of the NASA High Dependability Computing Consortium Workshop, NASA AMES, Moffet Field, CA, January 2001.
- C160** Rohit Khare, Michael Guntersdorfer, Peyman Oreizy, **Nenad Medvidović**, Richard N. Taylor. xADL: Enabling Architecture-Centric Tool Integration With XML. In Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS-34), Maui, Hawaii, January 2001.
- C161** Nikunj R. Mehta, **Nenad Medvidović**, and Sandeep Phadke. Towards a Taxonomy of Software Connectors. In Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, June 2000.

- C162** Paul Gruenbacher, Alexander Egyed, and **Nenad Medvidović**. Dimensions of Concerns in Requirements Negotiation and Architecture Modeling. In Proceedings of the Workshop on Multi-Dimensional Separation of Concerns in Software Engineering, Limerick, Ireland, June 2000.
- C163** **Nenad Medvidović**, Rose F. Gamble, and David S. Rosenblum. Towards Software Multioperability: Bridging Heterogeneous Software Interoperability Platforms. In Proceedings of the Fourth International Software Architecture Workshop (ISAW-4), Limerick, Ireland, June 2000.
- C164** Alexander Egyed and **Nenad Medvidović**. A Formal Approach to Heterogeneous Software Modeling. In Proceedings of the 2000 Conference on the Fundamental Approaches to Software Engineering (FASE 2000), Berlin, Germany, March-April 2000.
- C165** Alexander Egyed, Nikunj R. Mehta, and **Nenad Medvidović**. Software Connectors and Refinement in Family Architectures. In Proceedings of the Third International Workshop on Software Architectures for Product Families (IW-SAPF3), Las Palmas de Gran Canaria, Spain, March 2000.
- C166** Alexander Egyed and **Nenad Medvidović**. Extending Architectural Representation in UML with View Integration. In Proceedings of the 2nd International Conference on The Unified Modeling Language (UML'99), Fort Collins, CO, October 1999.
- C167** Marwan Abi-Antoun and **Nenad Medvidović**. Enabling the Refinement of a Software Architecture into a Design. In Proceedings of the 2nd International Conference on The Unified Modeling Language (UML'99), Fort Collins, CO, October 1999.
- C168** **Nenad Medvidović**, Alexander Egyed, and David S. Rosenblum. Round-Trip Software Engineering Using UML: From Architecture to Design and Back. In Proceedings of the Second International Workshop on Object-Oriented Reengineering (WOOR'99), Toulouse, France, September 1999.
- C169** Eric M. Dashofy, **Nenad Medvidović**, and Richard N. Taylor. Using Off-the-Shelf Middleware to Implement Connectors in Distributed Software Architectures. In Proceedings of the 21st International Conference on Software Engineering (ICSE'99), Los Angeles, CA, May 1999.
- C170** **Nenad Medvidović**, David S. Rosenblum, and Richard N. Taylor. A Language and Environment for Architecture-Based Software Development and Evolution. In Proceedings of the 21st International Conference on Software Engineering (ICSE'99), Los Angeles, CA, May 1999.
- Finalist for the ICSE'99 Most Influential Paper Award
- C171** **Nenad Medvidović** and David S. Rosenblum. Assessing the Suitability of a Standard Design Method for Modeling Software Architectures. In Proceedings of the First Working IFIP Conference on Software Architecture (WICSA-1), San Antonio, TX, February 1999.
- C172** **Nenad Medvidović** and Richard N. Taylor. Separating Fact from Fiction in Software Architecture. In Proceedings of the Third International Software Architecture Workshop (ISAW-3), Orlando, FL, November 1998.

- C173 Nenad Medvidović**, Eric M. Dashofy, and Richard N. Taylor. Employing Off-the-Shelf Connector Technologies in C2-Style Architectures. In Proceedings of the California Software Symposium (CSS'98), Irvine, CA, October 1998.
- C174** Jason E. Robbins, **Nenad Medvidović**, David F. Redmiles, and David S. Rosenblum. Integrating Architecture Description Languages with a Standard Design Method. In Proceedings of the 20th International Conference on Software Engineering (ICSE'98), Kyoto, Japan, April 1998.
- C175** Peyman Oreizy, **Nenad Medvidović**, and Richard N. Taylor. Architecture-Based Runtime Software Evolution. In Proceedings of the 20th International Conference on Software Engineering (ICSE'98), Kyoto, Japan, April 1998.
- Winner of the ICSE'98 Most Influential Paper Award, awarded at ICSE 2008
- C176** **Nenad Medvidović**, Richard N. Taylor, and David S. Rosenblum. An Architecture-Based Approach to Software Evolution. In Proceedings of the International Workshop on the Principles of Software Evolution (IWPSE-1), Kyoto, Japan, 1998.
- C177** Peyman Oreizy, **Nenad Medvidović**, Richard N. Taylor, and David S. Rosenblum. Software Architecture and Component Technologies: Bridging the Gap. In Proceedings of the Workshop on Compositional Software Architectures, Monterey, CA, January 1998.
- C178** **Nenad Medvidović** and David S. Rosenblum. Domains of Concern in Software Architectures and Architecture Description Languages. In Proceedings of the USENIX Conference on Domain-Specific Languages (DSL'97), Santa Barbara, CA, October 1997.
- C179** **Nenad Medvidović** and Richard N. Taylor. A Framework for Classifying and Comparing Architecture Description Languages. In Joint proceedings of the Sixth European Software Engineering Conference together and Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC-FSE'97), Zurich, Switzerland, September 1997.
- C180** **Nenad Medvidović**, Peyman Oreizy, and Richard N. Taylor. Reuse of Off-the-Shelf Components in C2-Style Architectures. In Proceedings of the 1997 Symposium on Software Reusability (SSR'97), Boston, MA, May 1997. Also in Proceedings of the 1997 International Conference on Software Engineering (ICSE'97), Boston, MA, May 1997.
- C181** **Nenad Medvidović** and Richard N. Taylor. Reusing Off-the-Shelf Components to Develop a Family of Applications in the C2 Architectural Style. In Proceedings of the First International Workshop on Software Architectures for Product Families (IW-SAPF-1), Las Navas del Marques, Avila, Spain, November 1996.
- C182** **Nenad Medvidović**, Peyman Oreizy, Jason E. Robbins, and Richard N. Taylor. Using Object-Oriented Typing to Support Architectural Design in the C2 Style. In Proceedings of the Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE4), San Francisco, CA, October 1996.
- C183** **Nenad Medvidović**. ADLs and Dynamic Architecture Changes. In Alexander L. Wolf, ed., Proceedings of the Second International Software Architecture Workshop (ISAW-2), San Francisco, CA, October 1996.

- C184** Nenad Medvidović, Richard N. Taylor, and E. James Whitehead, Jr. Formal Modeling of Software Architectures at Multiple Levels of Abstraction. In Proceedings of the California Software Symposium 1996, Los Angeles, CA, April 1996.
- C185** Richard N. Taylor, Nenad Medvidović, Kenneth M. Anderson, E. James Whitehead Jr. and Jason E. Robbins. A Component- and Message-Based Architectural Style for GUI Software. In Proceedings of the Seventeenth International Conference on Software Engineering (ICSE17), Seattle, WA, April 1995.
- Fast-tracked journal publication as one of the best papers of the conference Finalist for the ICSE'95 Most Influential Paper Award
- C186** E. James Whitehead, Jr., Jason E. Robbins, Nenad Medvidović, and Richard N. Taylor. Software Architecture: Foundation of a Software Component Marketplace. In Proceedings of the First International Workshop on Architectures for Software Systems (ISAW-1), Seattle, WA, April 1995.

Refereed Book Chapters

- B1** Nenad Medvidović and Alessandro Orso. Software Analytics Under the Lamppost (or What Star Trek Teaches Us About the Importance of Asking the Right Questions). In T. Menzies, L. Williams, and T. Zimmermann, eds., *Perspectives on Data Science for Software Engineering*, Morgan Kaufmann, July 2016.
- B2** Chris A. Mattmann, Daniel J. Crichton, Andrew F. Hart, Cameron Goodale, J. Steven Hughes, Sean Kelly, Luca Cinquini, Thomas H. Painter, Joseph Lazio, Duane Waliser, Nenad Medvidović, Jinwon Kim, and Peter Lean. Architecting Data-Intensive Systems. In B. Furht and A. Escalante, eds., *Handbook of Data Intensive Computing*, Springer Verlag, 2011.
- B3** Ivo Krka, George Edwards, Leslie Cheung, Leana Golubchik, and Nenad Medvidović. A Comprehensive Exploration of Challenges in Architecture-Based Reliability Estimation. In R. de Lemos, J.-C. Fabre, C. Gacek, F. Gadducci, and M. ter Beek, eds., *Architecting Dependable Systems VI*, Springer Verlag, December 2009.
- B4** Sam Malek, Nels Beckman, Marija Mikic-Rakic, and Nenad Medvidović. A Framework for Ensuring and Improving Dependability in Highly Distributed Systems. In R. de Lemos, et al., eds., *Architecting Dependable Systems III*, Springer Verlag, October 2005.
- B5** Paris Avgeriou, Nicolas Guelfi, and Nenad Medvidović. Software Architecture Description and UML. In N. J. Nunes, B. Selic, A. Rodrigues da Silva, eds., *UML Modeling Languages and Applications*, LNCS 3297, Springer Verlag, February 2005.
- B6** Roshanak Roshandel and Nenad Medvidović. Multi-View Software Component Modeling for Dependability. In R. de Lemos, C. Gacek, and A. Romanowski, eds., *Architecting Dependable Systems II*, Lecture Notes in Computer Science 3069, Springer Verlag, June 2004.
- B7** Nenad Medvidović and Nikunj Mehta. Java Beans and Software Architecture. In Hossein Bidgoli, ed., *The Internet Encyclopedia*, John Wiley & Sons, Inc., vol. 2, December 2003.

- B8** **Nenad Medvidović**, Marija Mikic-Rakic, and Nikunj Mehta. Improving Dependability of Component-Based Systems via Multi-Versioning Connectors. In R. de Lemos, et al., eds., *Architecting Dependable Systems*, Lecture Notes in Computer Science (LNCS 2677), Springer Verlag, August 2003.

Invited, Weakly Refereed, and Non-Refereed Publications

- I1** **Nenad Medvidović**. Software Engineering Research Trends 1994-2024: Stepping Beyond the Lamppost. *IEEE Transactions on Software Engineering*, vol. 51, no. 3 (March 2025).
- I2** **Nenad Medvidović**, Richard N. Taylor, and Eric M. Dashofy. Software Architecture Description Revisited. *IEEE Transactions on Software Engineering*, vol. 51, no. 3 (March 2025).
- I3** Joshua Garcia, Mehdi Mirakhorli, Lu Xiao, Sam Malek, Rick Kazman, Yuanfang Cai, and **Nenad Medvidović**. SAIN: A Community-Wide Software Architecture Infrastructure. In Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE 2023), Technical Briefing Track, Melbourne, Australia, May 2023.
- I4** Joshua Garcia, Mehdi Mirakhorli, Lu Xiao, Sam Malek, Rick Kazman, Yuanfang Cai, and **Nenad Medvidović**. SAIN: A Community-Wide Software Architecture Infrastructure. In Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE 2023), Technical Briefing Track, Melbourne, Australia, May 2023.
- I5** **Nenad Medvidović**. Pains and Gains of Peer Reviewing in Software Engineering: A Journal-Centric Perspective. *ACM SIGSOFT Software Engineering Notes*, vol. 46, no. 3, July 2021.
- I6** Yixue Zhao and **Nenad Medvidović**. Mobile-App Analysis and Instrumentation Techniques Reimagined with DECREE. In Recognizing and Rewarding Open Science in Software Engineering (ROSE Festival), held at ICSE 2019, Montreal, Canada, May 2019.
- I7** **Nenad Medvidović**. Editorial: State of the Journal. *IEEE Transactions on Software Engineering*, vol. 45, no. 1, January 2019.
- I8** Hakan Erdogmus, **Nenad Medvidović**, and Frances Paulisch. 50 Years of Software Engineering. *IEEE Software*, vol. 25, no. 6, November/December 2018.
- I9** Arman Shahbazian, Yuriy Brun, and **Nenad Medvidović**. eQual: Search-Based Software Design Optimization Poster paper. In Proceedings of the 40th International Conference on Software Engineering (ICSE 2018), Gothenburg, Sweden, May 2018.
- I10** Barbora Buhnova, Antonio Vallecillo, **Nenad Medvidović**, Magnus Larsson, Javier López, Jorge Cuellar. Guest editorial to the Special Issue on Component-Based Software Engineering and Software Architecture. *Science of Computer Programming*, vol. 90, part B, September 2014.
- I11** **Nenad Medvidović** and Chris A. Mattmann. Leveraging Software Architecture to Reconcile the Promise and Reality of Grid Computing. Infosys Technology Roundtable, September 2009. <http://www.infosys.com/newsletter/technology-roundtable/web.htm>
- I12** Peyman Oreizy, **Nenad Medvidović**, and Richard N. Taylor. Runtime Software Adaptation: Framework, Approaches, and Styles. In Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), follow-on to the Most Influential Paper of ICSE 1998, Leipzig, Germany, May 2008.

- I13 Nenad Medvidović.** Foreword to Rogerio De Lemos, Cristina Gacek, and Alexander Romanovsky, eds., *Architecting Dependable Systems*, LNCS 4615, Springer Verlag, 2007.
- I14 Leslie Cheung, Leana Golubchik, Nenad Medvidović, and Gaurav Sukhatme.** Identifying and Addressing Uncertainty in Architecture-Level Software Reliability Modeling. In *Proceedings of the NSF Next Generation Software Program (NSFNGS) Workshop*, Long Beach, CA, February 2007.
- I15 Nenad Medvidović.** Moving Architectural Description from Under the Technology Lamppost. In *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications*, Dubrovnik, Croatia, August 2006.
- I13 David Woollard and Nenad Medvidović.** High Performance Software Architectures: A Connector-Oriented Approach. In *Proceedings of the Institute for Software Research Graduate Research Symposium*, Irvine, CA, June 2006.
- I16 Somo Banerjee, Leslie Cheung, Leana Golubchik, Nenad Medvidović, Roshanak Roshandel, and Gaurav Sukhatme.** Engineering Reliability into Hybrid Systems via Rich Design Models: Recent Results and Current Directions. In *Proceedings of the NSF Next Generation Software Program (NSFNGS) Workshop*, Rhodes Island, Greece, April 2006.
- I17 Nenad Medvidović.** Software Architectures and Embedded Systems: A Match Made in Heaven? *IEEE Software*, September-October 2005.
- I18 Nenad Medvidović and Vladimir Jakobac.** A Focused Approach to Architectural Recovery. In *Proceedings of the 7th Ground System Architectures Workshop (GSAW 2003)*, Manhattan Beach, CA, March 2003.
- I19 Nenad Medvidović.** Coupling Architectural Discovery and Recovery to Stem Architectural Erosion. Invited paper for the Dagstuhl Workshop on Software Architecture Recovery and Modeling (SWARM), Dagstuhl, Germany, February 2003.
- I20 Andre van der Hoek, Ebru Dincel, and Nenad Medvidović.** Using Service Utilization Metrics to Assess and Improve Product Line Architectures. In *Proceedings of the 6th Ground System Architectures Workshop (GSAW 2002)*, El Segundo, CA, March 2002.
- I21 Roshanak Roshandel and Nenad Medvidović.** Static and Dynamic Modeling of Software Architecture. In *Proceedings of the 6th Ground System Architectures Workshop (GSAW 2002)*, El Segundo, CA, March 2002.
- I22 Nikunj R. Mehta, Nenad Medvidović, and Sandeep Phadke.** Rich Software Interconnections. In *Proceedings of the Fourth Ground System Architectures Workshop (GSAW 2000)*, El Segundo, CA, February 2000.
- I23 Nenad Medvidović.** Modeling Software Architectures in UML. Invited paper for the Workshop on Software Architectures and the Unified Modeling Language, Rational Software Corporation, Denver, CO, April 1999.
- I24 Nenad Medvidović.** An Approach to Architecture-Based Software Integration. In *Proceedings of the Ground System Architectures Workshop (GSAW 99)*, El Segundo, CA, March 1999.
- I25 Richard N. Taylor, Nenad Medvidović, and Peyman Oreizy.** Architectural Implications of Common Operator Interfaces. In *Proceedings of the Ground System Architectures Workshop (GSAW 98)*, El Segundo, CA, February 1998.

- I26** Jeffrey J. Blevins, Deborah L. Dubrow, Nancy Eickelmann, Rebecca Grinter, **Nenad Medvidović**, Ronald Reimer, Jonathan Shaw, Clark S. Turner, and Gary Wong. Report on the Software Project Management Technical Research Review. In Proceedings of the California Software Symposium (CSS'95), Irvine, CA, March 1995.

Non-Refereed Technical Reports

- N1** Youn Kyu Lee, Ruhollah Shemirani, Jae young Bang, Arman Shahbazian, Gholamreza Safi, and **Nenad Medvidović**. SEALANT: Preventing Inter-Application Attacks in Android. USC Center for Systems and Software Engineering Technical Report, USC-CSSE-2016-601, March 2016.
- N2** Duc M. Le, Pooyan Behnamghader, Joshua Garcia, Daniel Link, Arman Shahbazian, and **Nenad Medvidović**. An Empirical Study of Architectural Change in Open-Source Software Systems. USC Center for Systems and Software Engineering Technical Report, USC-CSSE-2014-509, December 2014.
- N3** Ivo Krka, Yuriy Brun, and **Nenad Medvidović**. Automatically Mining Specifications from Invocation Traces and Method Invariant. USC Center for Systems and Software Engineering Technical Report, USC-CSSE-2013-509, September 2013.
- N4** George Edwards and **Nenad Medvidović**. Model Interpreter Frameworks. USC Center for Systems and Software Engineering Technical Report, USC-CSSE-2009-514, July 2009.
- N5** Yuriy Brun, George Edwards, Jae Young Bang, and **Nenad Medvidović**. Online Reliability Improvement via Smart Redundancy in Systems with Faulty and Untrusted Participants. USC Center for Systems and Software Engineering Technical Report, USC-CSSE-2009- 510, May 2009.
- N6** Yuriy Brun and **Nenad Medvidović**. Preserving Privacy in Distributed Computation via Self-Assembly. USC Center for Systems and Software Engineering Technical Report, USC- CSSE-2008-819, September 2008.
- N7** Yuriy Brun and **Nenad Medvidović**. Discreetly Distributing Computation via Self-Assembly. USC Center for Systems and Software Engineering Technical Report, USC-CSSE-2007-714, April 2007.
- N8** David Woollard, Chris A. Mattmann, and **Nenad Medvidović**. Injecting Software Architectural Constraints into Legacy Scientific Applications. USC Center for Software Engineering Technical Report, USC-CSE-2007-701, January 2007.
- N9** Chiyoung Seo, Sam Malek, and **Nenad Medvidović**. An Energy Consumption Framework for Distributed Java-Based Software Systems. Technical Report USC-CSE-2006-604, Center for Software Engineering, University of Southern California, December 2006.
- N10** Sam Malek, Marija Mikic-Rakic, and **Nenad Medvidović**. A Decentralized Redeployment Algorithm for Improving the Availability of Distributed Systems. Technical Report USC-CSE-2004-506, USC Center for Software Engineering, March 2004.
- N11** Nikunj Mehta and **Nenad Medvidović**. Checking Style Conformance of Software Architectural Compositions. Technical Report USC-CSE-2004-504, USC Center for Software Engineering, February 2004.

- N12** Nikunj Mehta and **Nenad Medvidović**. Composition of Style-Based Software Architectures from Architectural Primitives. Technical Report USC-CSE-2004-503, USC Center for Software Engineering, February 2004.
- N13** Nikunj Mehta and **Nenad Medvidović**. Toward Composition Of Style-Conformant Software Architectures. Technical Report USC-CSE-2004-500, USC Center for Software Engineering, January 2004.
- N14** Marija Mikic-Rakic, Sam Malek, and **Nenad Medvidović**. Improving Availability in Large, Distributed, Component-Based Systems via Redeployment. Technical Report USC-CSE- 2003-515, USC Center for Software Engineering, December 2003.
- N15** Nikunj Mehta and **Nenad Medvidović**. Concise Composition of Architectural Styles from Architectural Primitives. Technical Report USC-CSE-2003-510, USC Center for Software Engineering, October 2003
- N16** Roshanak Roshandel and **Nenad Medvidović**. Relating Software Component Models. Technical Report USC-CSE-2003-504, USC Center for Software Engineering, March 2003.
- N17** Nikunj R. Mehta and **Nenad Medvidović**. Distilling Software Architecture Primitives from Architectural Styles. Technical Report USC-CSE-2002-509, USC Center for Software Engineering, September 2002.
- N18** Marija Mikic-Rakic and **Nenad Medvidović**. Software Architecture-Based Development Support for Ubiquitous Systems. Technical Report USC-CSE-2002-508, USC Center for Software Engineering, September 2002.
- N19** Ebru Dincel, **Nenad Medvidović**, and Andre van der Hoek. An Example Product Line Architecture: The Library System. Technical Report USC-CSE-2002-507, USC Center for Software Engineering, September 2002.
- N20** Ebru Dincel, **Nenad Medvidović**, and Andre van der Hoek. An Example Product Line Architecture: Troops Deployment System. Technical Report USC-CSE-2002-506, USC Center for Software Engineering, September 2002.
- N21** Ebru Dincel, **Nenad Medvidović**, and Andre van der Hoek. An Example Product Line Architecture: Digital Library Projects. Technical Report USC-CSE-2002-505, USC Center for Software Engineering, September 2002.
- N22** Marija Mikic-Rakic, **Nenad Medvidović**, and Vladimir Jakobac. Middleware for Software Architecture-Based Development in Distributed, Mobile, and Resource-Constrained Environments. Technical Report USC-CSE-2002-501, USC Center for Software Engineering, February 2002.
- N23** **Nenad Medvidović** and Marija Mikic-Rakic. Architectural Support for Programming-in-the-Many. Technical Report USC-CSE-2001-506, USC Center for Software Engineering, September 2001.
- N24** Ebru Dincel, Roshanak Roshandel, and **Nenad Medvidović**. ADL-Independent Architectural Representation in XML. Technical Report USC-CSE-00-519, USC Center for Software Engineering, May 2000.

- N25 Nenad Medvidović.** On the Role of Middleware in Architecture-Based Software Development. Available as Technical Report USC-CSE-00-517, USC Center for Software Engineering, April 2000.
- N26 Nenad Medvidović,** Peyman Oreizy, Richard N. Taylor, Rohit Khare, and Michael Guntersdorfer. An Architecture-Centered Approach to Software Environment Integration. Technical Report USC-CSE-00-516, USC Center for Software Engineering, March 2000.
- N27 Nikunj R. Mehta, Nenad Medvidović,** and Marija Rakic. Why Consider Implementation-Level Decisions in Software Architectures? Technical Report USC-CSE-00-514, USC Center for Software Engineering, February 2000.
- N28 Alexander Egyed, Paul Gruenbacher, and Nenad Medvidović.** Refinement and Evolution Issues between Requirements and Product Line Architectures. Technical Report USC-CSE-00-515, USC Center for Software Engineering, February 2000.
- N29 Nenad Medvidović,** David S. Rosenblum, and Richard N. Taylor. A Type Theory for Software Architectures. Technical Report, UCI-ICS-98-14, Department of Information and Computer Science, University of California, Irvine, April 1998.
- N30 Nenad Medvidović** and Richard N. Taylor. Reuse of Off-the-Shelf Constraint Solvers in C2-Style Architectures. Technical Report UCI-ICS-96-28, Department of Information and Computer Science, University of California, Irvine, July 1996.
- N31 Nenad Medvidović.** Formal Definition of the Chiron-2 Software Architectural Style. Technical Report UCI-ICS-95-24, Department of Information and Computer Science, University of California, Irvine, August 1995.
- N32 Jason E. Robbins, E. James Whitehead Jr., Nenad Medvidović,** and Richard N. Taylor. A Software Architecture Design Environment for Chiron-2 Style Architectures. Arcadia Technical Report UCI-95-01, University of California, Irvine, January 1995.

7 Research Support

Current Support

- G1** National Science Foundation — Software and Hardware Foundations
Sole PI at USC; collaborative proposal with Georgia Institute of Technology and UC Irvine
 A General Framework for Automated Test Transfer
 Duration: 10/1/21 - 9/30/25
 Grant Number: 2105871
 Award Amount to USC: \$400,000

Past Support

- G2** Department of Homeland Security
Sole PI; subcontract to Rochester Institute of Technology
 National Vulnerability Intelligence Platform
 Duration: 1/1/23 - 10/1/23
 Subcontract Amount: \$493,522

- G3** Google Research
Sole PI
Automatic Program Patching and Synthesis
Duration: 5/1/22 - 12/31/23
Award Amount: \$50,000
- G4** National Science Foundation — Community Infrastructure Planning
Sole PI at USC; lead institution on collaborative proposal with UC Irvine, Rochester
Institute of Technology, University of Hawaii,
Drexel University, and Stevens Institute of Technology
Constructing a Community-Wide Software Architecture Infrastructure
Duration: 9/1/18 - 8/31/23
Grant Number: 1823354
Award Amount to USC: \$684,000
- G5** National Science Foundation — Software and Hardware Foundations
Sole PI
Rooting Out Data- and Control-Flow Anomalies in Event-Based Systems Duration:
8/15/17 - 7/31/22
Grant Number: 1717963
Award Amount: \$500,000
- G6** Office of Naval Research
Co-PI
Ensuring Security of Android Software via Tailorable Multi-Layer Customization
Co-PIs: Chao Wang, USC (lead PI)
William GJ Halfond, USC
Duration: 9/1/17 - 8/31/21
Grant Number: N00014-17-1-2896
Award Amount: \$1,082,320
- G7** Google
Sole PI
Mining Architectural Information to Stem Technical Debt
Duration: 10/1/19 - 4/30/21
Award Amount: \$42,500
- G8** National Science Foundation — Community Infrastructure Planning
Sole PI
Planning and Prototyping a Community-Wide Software Architecture Instrument
Duration: 8/1/16 - 7/31/19
Grant Number: 1629977
Award Amount: \$70,000
- G9** Huawei Technologies
Sole PI
Advanced Support for Software Architecture Recovery and Maintenance
Duration: 2/15/18 - 2/14/19
Award Amount: \$200,000

- G10** National Science Foundation — Software and Hardware Foundations
Sole PI
 Techniques for Pruning Problem and Solution Spaces to Enable Methodical Exploration
 of Software Development Alternatives
 Duration: 6/1/16 - 5/31/19
 Grant Number: 1618231
 Award Amount: \$500,000
- G11** Huawei Technologies
Sole PI
 Intelligent Support for Architecture-Based Forward- and Reverse-Engineering
 Duration: 12/1/16 - 11/30/17
 Award Amount: \$145,000
- G12** Northrop Grumman
Sole PI
 Parametric and Simulation Models Research and Calibration
 Duration: 8/16/2014 - 5/15/2017
 Award Amount: \$30,000
- G13** National Science Foundation — Software and Hardware Foundations
Co-PI
 Helping Developers Improve the Energy Consumption of Smartphone Applications
 Co-PIs: William GJ Halford, USC (lead PI)
 Ramesh Govindan, USC
 Duration: 7/1/13 - 6/30/17
 Grant Number: 1321141
 Award Amount: \$500,000
- G14** National Science Foundation — Software and Hardware Foundations
Sole PI
 From Scenario-Based Software Requirements to Component-Level Behavior
 Duration: 9/01/12 - 8/31/16
 Grant Number: 1218115
 Award Amount: \$500,000
- G15** National Science Foundation — Software and Hardware Foundations
Sole PI
 Automating the Detection of Architectural Degradation in Software Systems
 Duration: 8/01/11 - 7/31/15
 Grant Number: 1117593
 Award Amount: \$300,000
- G16** Defense Advanced Research Projects Agency (DARPA)
Lead PI
 Testing Privacy-Preserving Distributed Systems on SAFERlab
 Co-PIs: Yuriy Brun, University of Massachusetts
 Sam Malek, George Mason University
 Contract Number: N66001-11-C-4021
 Duration: 11/1/13 - 2/28/15
 Award Amount: \$300,000

- G17** Intelligence Advanced Research Projects Activity (IARPA)
Lead PI
 OpTile: Private Computing in the Open
 Co-PIs: Yuriy Brun, University of Massachusetts
 Sam Malek, George Mason University
 Contract Number: N66001-13-1-2006
 Duration: 9/27/13 - 9/26/14
 Award Amount: \$300,000
- G18** Northrop Grumman
Sole PI
 Architectural Transformation of Legacy Systems to Future Systems
 Duration: 1/1/14 - 5/15/14
 Award Amount: \$30,000
- G19** Northrop Grumman
Sole PI
 Domain-Specific Modeling and Simulation in Collaborative Software Development Projects
 Duration: 8/16/13 - 5/15/14
 Award Amount: \$10,000
- G20** Infosys Technologies Ltd.
Sole PI
 CoDesign - A Collaborative Software Design Environment
 Duration: 10/1/11 - 8/31/14
 Award Amount: \$200,000
- G21** Northrop Grumman
Sole PI
 Domain-Specific Modeling and Simulation
 Duration: 8/16/12 - 5/15/13
 Award Amount: \$10,000
- G22** Northrop Grumman
Sole PI
 Exploring Design Alternatives in Software-Intensive Projects
 Duration: 8/16/11 - 5/15/12
 Award Amount: \$10,000
- G23** National Science Foundation — Software and Hardware Foundations
Lead PI
 Early Reliability Modeling and Prediction of Embedded Software Systems
 Duration: 3/01/10 - 8/31/12
 Co-PIs: Leana Golubchik, USC
 Gaurav Sukhatme, USC
 Grant Number: 0905665
 Award Amount: \$210,000

- G24** Stevens Institute of Technology — Systems Engineering Research Center
Sole PI
 Requirements for Net-Centric Enterprises (RT-25)
 Duration: 10/1/10 - 12/31/11
 Award Amount: \$270,298
- G25** National Science Foundation — Software for Real-World Systems
Sole PI at USC; collaborative proposal with UC Irvine
 Recombinant Services - Recasting the Web for Continuously Evolving Systems
 Duration: 9/15/08 - 8/31/11
 Grant Number: 0820170
 Award Amount to USC: \$193,670
- G26** Office of Secretary of Defense — UAST
Sole PI; subcontract to MIT
 A Prescriptive and Adaptive Testing Framework (PATFrame) for Unmanned and
 Autonomous Systems of Systems
 Duration: 10/1/09 - 9/30/10
 Subcontract Amount: \$128,649
- G27** Infosys Technologies Ltd.
Sole PI
 An Extensible Infrastructure for Collaborative Architecture-Based Development of
 Distributed Software-Intensive Systems
 Duration: 9/1/08 - 12/31/10
 Grant Number: 0208
 Award Amount: \$200,000
- G28** Jet Propulsion Laboratory — SURP
Co-PI
 Reconfigurable Software for Recovering from Component Failures in Long Duration
 Missions
 Co-PIs: Gaurav Sukhatme, USC (co-PI)
 Duration: 9/1/09 - 5/31/10
 Award Amount: \$75,000
- G29** National Institutes of Health Co-PI
 Mobile Device Biomonitoring to Prevent and Treat Obesity in Underserved Minority
 Youth
 Duration: 8/1/08 - 7/31/10
 Co-PIs: Donna Spruijt-Metz, USC (lead PI)
 Murali Annavaram, USC
 Urbashi Mitra, USC
 Shrikanth Narayanan, USC
 Gaurav Sukhatme, USC
 Award Amount: \$948,348

- G30** Robert Bosch Research and Technology Centers
Sole PI
 Software Architectural Support for Embedded Systems
 Duration: 3/1/06 - 5/31/09
 Award Amount: \$470,000
- G31** National Science Foundation — Information Technology Research
Sole PI
 Software Architectural Support for Programming-in-the-Small-and-Many
 Duration: 9/1/03 - 8/31/09
 Grant Number: 0312780
 Award Amount: \$448,000
- G32** National Science Foundation — Computer Systems Research
Lead PI
 Engineering Reliability into Hybrid Systems: A Compositional and Hierarchical Approach
 Duration: 8/01/07 - 7/31/09
 Co-PIs: Leana Golubchik, USC
 Grant Number: 0720612
 Award Amount: \$40,000
- G33** U.S. Army
Co-PI
 Future Combat Systems Software Definition, Design, Development, and Deployment
 Duration: 1/1/08 - 12/31/08
 Co-PIs: Barry Boehm (lead PI)
 Award Amount: \$425,000
- G34** IBM Real-Time Innovation
Sole PI
 Prism-RT: A Java-Based Architectural Middleware Platform for Real-Time Embedded Systems
 Duration: 12/1/2007 - 12/31/08
 Award Amount: \$20,000
- G35** National Science Foundation — Science of Design
Co-PI
 Value-Based Science of Design
 Duration: 12/15/04 - 11/30/08
 Co-PIs: Barry W. Boehm, USC (lead PI)
 Grant Number: 0438931
 Award Amount: \$400,000
- G36** National Science Foundation — Computer Systems Research
Lead PI
 Engineering Reliability Into Hybrid Systems via Rich Design Models
 Duration: 7/1/05 - 6/30/07
 Co-PIs: Leana Golubchik, USC
 Grant Number: 0509539
 Award Amount: \$100,000

- G37** National Science Foundation — Federal Cyber Service: Scholarship for Service
Co-PI
 An Innovative Information Assurance and Security Technology Capacity Development
 and Outreach Program
 Duration: 8/1/04 - 7/31/07
 Co-PIs: Leana Golubchik, USC (lead PI)
 B. Clifford Neuman, USC
 Gerard Medioni, USC
 Grant Number: 0417274
 Award Amount: \$300,000
- G38** Okawa Research Foundation
Sole PI
 The GridLite DREAM: Bringing the Grid to Your Pocket
 Duration: 10/1/05 - 9/30/06
 Award Amount: \$10,000
- G39** Boeing
Sole PI
 C4ISR SADD Development Assistance (FCS)
 Duration: 10/1/05 - 3/31/06
 Award Amount: \$100,000
- G40** Jet Propulsion Laboratory Sole PI
 Software Connectors for Data Distribution
 Duration: 7/1/05 - 8/31/05
 Award Amount: \$18,144
- G41** NASA High-Dependability Computing Consortium
Co-PI
 Empirical Research on High Dependability Computing
 Duration: 4/1/01 - 3/31/06
 Co-PIs: Barry W. Boehm, USC (lead PI)
 Award Amount: \$2,500,000
- G42** National Science Foundation — CAREER
Sole PI
 Architecture-Based Support for Software Application Families
 Duration: 7/1/00 - 6/30/05
 Grant Number: 9985441
 Award Amount: \$241,000
- G43** Boeing
Sole PI
 Software Architecture-Based Development of Product Lines for the Tactical Radio
 Domain
 Duration: 1/1/04 - 12/31/04
 Award Amount: \$57,500

- G44** Jet Propulsion Laboratory
Sole PI
 Extensible Architectural Technology for Managing Complex Space Data Systems
 Duration: 11/1/03 - 8/31/04
 Contract Number: 579951
 Award Amount: \$50,000
- G45** Defense Advanced Research Projects Agency (DARPA)
Lead PI
 Dynamic Assembly, Assessment, Assurance, and Adaptation via Heterogeneous Software Connectors
 Duration: 7/1/00 - 10/31/03
 Co-PIs: Barry W. Boehm, USC
 Contract Number: F30602-00-2-0615
 Award Amount: \$460,000
- G46** Jet Propulsion Laboratory
Sole PI
 Relating State-Based and Component/Connector-Based Architectural Styles
 Duration: 8/1/00 - 8/31/03
 Contract Number: 1219801
 Award Amount: \$95,672
- G47** U.S. Army Tank-Automotive and Armaments Command
Sole PI
 COTS Components and Connectors
 Duration: 9/1/01 - 3/31/03
 Contract Number: 1400508
 Award Amount: \$68,350
- G48** Xerox Corporation
Sole PI
 Architecting Application Families
 Duration: 9/1/00 - 8/31/02
 Award Amount: \$60,000
- G49** Intel Corporation
Lead PI
 Embedded Systems Lab: A Teaching and Research Facility
 Award Date: July 2001, July 2002
 Co-PIs: Gaurav Sukhatme, USC
 Award Amount: \$95,000 (in equipment and cash)
- G50** Defense Advanced Research Projects Agency (DARPA)
Sole PI; subcontract to UC Irvine
 Architecture-Centric Dynamic Adaptation of Deeply Networked Embedded Systems
 Duration: 9/23/99 - 1/22/01
 Contract Number: F30602-99-C-0174
 Subcontract Amount: \$92,500

8 Patents

- 07/2022 Latency Minimization in Mobile Applications
U.S. Patent 11,379,551
- 11/2020 SEALANT: Security for End-users of Android via Light-weight ANalysis Techniques
U.S. Patent 10,827,349
- 12/2012 Tile Architectural Style for Privacy-Preserved Distributed Computing
U.S. Patent 8,332,457

9 Teaching

University of Southern California

Courses Developed and Taught

- T1** CSCI 699 — Software Forensics
Ph.D.-level seminar
Spring 2020
- T2** CSCI 578 — Software Architectures
Graduate-level course, a core course for the M.S. specialization in Software Engineering
Spring 1999, 2000, 2001, 2002, 2004, 2005, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2015, 2016, 2017, 2018
Fall 2008, 2009, 2010, 2011, 2012, 2014; 2019, 2020
- T3** CSCI 310 — Software Engineering
Upper-division undergraduate course Spring 2021
Fall 2016, 2017
- T4** CSCI 589 — Software Engineering for Embedded Systems
Graduate-level course
Fall 2001, 2002, 2003, 2004, 2006, 2007, 2015
- T5** CSCI 499 — Modeling Software-Intensive Systems
Upper-division undergraduate / lower-division graduate seminar
Spring 2011
- T6** CSCI 377 — Introduction to Software Engineering
Upper-division undergraduate course
Fall 2004, 2006, 2007, 2008, 2009
- T7** CSCI 599 — Formal Methods in Software Architectures
Graduate seminar
Fall 2000
- T8** CSCI 599 — Current Issues in Software Architectures
Graduate seminar
Fall 1999

Courses Developed

- T9** CSCI 568 — Requirements Engineering
Graduate-level course, a core course for the Certificate in Software Architecture

Courses Taught

- T10** CSCI 591 — Computer Science Research Colloquium
M.S.-level research seminar series
Fall 2023
- T11** CSCI 477 — Design and Construction of Large Software Systems
Upper division undergraduate / lower-division graduate course
Spring 2003

Other Universities

- T12** Mining Software Repositories for Architectural Knowledge
International Summer School on Software Engineering, Salerno, Italy
Graduate, half-day course
July 2017
- T13** Software Architecture and Robotics
LASER Summer School
Advanced graduate, one-week course
September 2017
- T14** Software Architectures
Universidad Politécnica de Madrid, Spain
Graduate, one-week course, part of European Master of Software Engineering program
March 2011
- T15** Advanced Software Architectures
Politecnico di Milano, Italy
Graduate course
July 2010
Two-week course, part of PoliMi's Ph.D. program
- T16** Software Architecture
University of Buenos Aires, Argentina
Graduate, one-week course
Part of UBA's Winter School
August 2009
- T17** NSEN 6111 — Software Architectures
Walden University, USA
Graduate online course
Co-taught with Richard N. Taylor and Eric Dashofy
Spring 2008

Industry

- T18** Advanced Topics in Software Architecture
Samsung Electronics Co. Ltd., Suwon, South Korea
2-day professional development course
June 2013

- T19** Software Architecture
 Samsung Electronics Co. Ltd., Suwon, South Korea
 3-day professional development course
 June 2013
- T20** Software Architecture
 Infosys Technologies Ltd., Mysore, India
 3-week professional development course
 December 2009
- T21** Software Architectures
 Jet Propulsion Laboratory, Pasadena, CA
 10-week professional development course
 Spring 2001, 2002; Summer 2001, 2002

Teaching Assistantships

- T22** ICS 121 — Introduction to Software Engineering
 Junior/senior-level undergraduate course (Summer 1994)
 University of California, Irvine
- T23** ICS 141 — Programming Languages
 Junior/senior-level undergraduate course (Spring 1994)
 University of California, Irvine
- T24** ICS 52 — Systematic Software Construction
 Sophomore-level undergraduate course (Fall 1993, Winter 1994)
 University of California, Irvine
- T25** CSE 460/598 — Software Project Management and Development I
 Senior-level undergraduate course (Fall 1992, Spring 1993, Summer 1993)
 Arizona State University

10 Supervised Students

Current Doctoral Students - Advisor

- Satyaki Das
 Start date: August 2021
- Syeda Tasnim Fabiha
 Start date: August 2022
- Abid Hassan
 Start date: August 2023
- Jincheng He
 Start date: August 2020
- Tooraj Helmi
 Start date: August 2020
- Tuan Ngo
 Start date: August 2023

- Ghazal Rafiei
Start date: August 2023

Doctoral Students - Dissertation Committee Member

Current

- Omer Faruk Akgul
- Yazeed Alabdulkarim
- Reem Alfayez
- Greg Harris
- Yannan Li
- Sara Mohammadinejad
- Khiem Ngo
- Xin Qin
- Hao Shi
- Di Wu

Graduated

- Marwan Abi-Antoun, CMU
- Negarsadat Abolhassani
- Mohammad-Mirza Aghatabar
- Abdulmajeed Alameer
- Mohamed Al Said
- Muhammad Ali Amer
- Jesper Andersson, Linkoping University, Sweden
- Pongtip Aroonvatanaporn
- Adi Azar
- Jongmoon Baik
- Sumita Barahmand
- Pooyan Behnamghader
- Jesal Bhuta
- Yue Chen
- Zhihao Chen
- Celia Chen
- Hugo Cheung
- Leslie Cheung
- Paul Chiou
- Kun Young Chung
- Jürgen Cito, University of Zurich
- Eric Dashofy, UC Irvine
- Shirin Ebrahimi-Taghizadeh
- Alexander Egyed

- Ragy Eleish
- Alessio Gambi, University of Lugano, Switzerland
- Jiaping Gui
- Shuai Hao
- Anandi Hira
- I-De Huang
- LiGuo Huang
- Shahdad Irajpour
- Jinwoo Kim
- Hasan Kitapci
- Yoon-Ju Lee
- Ding Li
- In-Young Ko
- Supannika Koolmanojwong
- Alexander Lam
- DeWitt Lattimer
- Joo H. Lee
- Keun Lee
- Qi Li
- Sonal Mahajan
- Ali Malik
- Muhammad Murtaza
- David Naffin
- Vu Nguyen
- Brandon Paulsen
- Luis Pedrosa
- Gustavo Perez
- Monvorath Phongpaibul
- Kan Qi
- Hitesh Sajnani, UC Irvine
- Selva Samuel, Carnegie Mellon University
- Aaron Schram, University of Colorado, Boulder
- Shideh Shahidi
- Doochul Shin
- Chunga Sung
- Giordano Tamburrelli, Politecnico di Milano, Italy
- Thomas Tan
- Shahzad Tiwana
- Marcos Vieira
- Sasha Volokh

- Mian Wan
- Jingbo Wang
- Dan Wu
- Ye Yang
- Jason Yap
- Zhaoxu Zhang

Masters Students

Graduated

- Ian Elston
- John Morse
- Marwan Abi-Antoun
- Raghbir Banwait
- Miheer Bhachech
- Ebru Dincel
- Lei Ding
- Luka Dojcilovic
- Brian D'Souza
- Eder Figueroa
- Prakash Gupta
- Sheryl John
- Himanshu Joshi
- Suhrid Karthik
- Armagan Kilic
- Vishal Kudchadkar
- Chenggang Li
- Mehrdad Mahdavi
- Sean McCleese
- Suhas Mehta
- Daye Nam
- Mahindra Pai
- Sandeep Phadke
- Aliasgar Rampurwala
- Tatsuhiko Tomita
- Vinaynathan Viswanathan
- Michael Wakerly
- Dongwoo Won
- David Woollard

Undergraduate Students

Graduated

- Nels Beckman
- Morgan Brown
- Vanessa Carson
- Justin Chen (Columbia University)
- Charles DeBergh
- Kelly Dickson
- Eric Dong
- Joshua Garcia
- Kevin Jing
- Trevor Johns
- Yi (Nick) Kuang
- Jerry Lin
- Jacob Schaidler
- Anita Singh
- Markus Snelder
- Paul Wat
- Yifeng Yu

11 Formal Presentations

Keynote Presentations

- P1** *Software Archipelagos as Impediments to Scientific Progress*. Montenegrin Academy of Sciences and Arts, Podgorica, Montenegro, November 2022.
- P2** *Software Archipelagos as Impediments to Scientific Progress*. Central European Conference on Information and Intelligent Systems (CECIIS 2022), Dubrovnik, Croatia, September 2022.
- P3** *Mining Architectural Information to Stem Technical Debt*. International Workshop on Mining Software Repositories for Software Architecture (MSR4SA), held with the European Conference on Software Architecture, online, September 2021.
- P4** *Human Archipelagos Have Been Induced by COVID but Architectural Archipelagos Are All Our Own Doing*. IEEE 18th International Conference on Software Architecture (ICSA 2021), online, March 2021.
- P5** *Stemming Architectural Decay in Software Systems*. Fourth International conference on Tools and Methods of Program Analysis (TMPA-2017), Moscow, Russia, March 2017.
- P6** *Software Decay as a Big-Data Problem Viewed Through the Architectural Lens*. 2016 International Simulation Multi-Conference (AsiaSim / SCS AutumnSim 2016), Beijing, China, October 2016.

- P7** *Viewing Software Decay Through the Architectural Lens*. International Symposium on Software Engineering and Applications (SEA 2015), Marina del Rey, CA, October 2015.
- P8** *What Architecture Can Teach Us About When, Where, and Why Software Systems Decay*. 9th European Conference on Software Architecture (ECSA 2015), Dubrovnik/Cavtat, Croatia, September 2015.
- P9** *Adapting Our View of Software Adaptation: An Architectural Perspective*. 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2014), Hyderabad, India, June 2014.
- P10** *PhD Research: What, Where, Why, When, How*. INCOSE/CSER Systems Engineering and Architecting Doctoral Student Network (SEANET), University of Southern California, Los Angeles, CA, March 2014.
- P11** *When, Where, and Why Software Systems Decay and What We Can Do about It*. University of California Irvine (UCI) Institute for Software Research Forum, Irvine, CA, May 2013.
- P12** *Software Architecture Challenges & Opportunities for the 21st Century: Dynamism, Mobility, Autonomy, Services, Grids & Clouds*. 11th Brazilian Symposium on Software Quality, Fortaleza, Brazil, June 2012.
- P13** *Software Architecture Challenges & Opportunities for the 21st Century: Dynamism, Mobility, Autonomy, Services, Grids & Clouds*. 5th India Software Engineering Conference (ISEC 2012), Kanpur, India, February 2012.
- P14** *Software Architecture and Mobility: A Perfect Marriage or an Uneasy Alliance*. International Workshop on Software Architectures and Mobility (SAM 2008), Leipzig, Germany, May 2008.
- P15** *Moving Architectural Description from Under the Technology Lamppost*. Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS 2007), Campinas, Brazil, August 2007.
- P16** *Moving Architectural Description from Under the Technology Lamppost*. 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), Dubrovnik, Croatia, August, 2006.
- P17** *From Dependable Architectures to Dependable Systems*. Workshop on Architecting Dependable Systems (WADS 2004) held in conjunction with the International Conference on Dependable Systems and Networks (DSN 2004), Florence, Italy, June 2004.

Academic Conference Presentations

- P18** *Software Architecture and COVID: A Marriage of (In)Convenience*. New and Emerging Ideas Panel: Software Architecture in the COVID-19 Reality. IEEE 18th International Conference on Software Architecture (ICSA 2021), online, March 2021.
- P19** *Online Sports Betting Through the Prism of Software Engineering*. Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020), November 2020.
- P20** *An Architectural Style for Solving Computationally Intensive Problems on Large Networks*. (with Yuriy Brun) 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2020), Seoul, South Korea, May 2020.

- P21** *ARCADE: A Workbench for Mining Architectural Information and Identifying Technical Debt.* Workshop on the National Java Repository Project, San Diego, CA, November 2019
- P22** *Mining Architectural Information to Stem Technical Debt.* Southern California Software Engineering Symposium, Irvine, CA, June 2016.
- P23** *Viewing Software Decay Through the Architectural Lens.* 2nd Latin American School on Software Engineering, Porto Alegre, Brazil, July 2016.
- P24** *An End-to-End Domain-Specific Modeling and Analysis Platform.* 8th International Workshop on Modeling in Software Engineering (MISE 2016), Austin, TX, May 2016.
- P25** *Extracting the Essence of Software Systems' Architectures Through Unstructured-Data Mining.* Mining and Modeling Unstructured Data in Software - Challenges for the Future, Shonan Village, Japan, March 2016.
- P26** *An Empirical Study of Architectural Decay in Software Systems.* 39th International Conference on Software Engineering (ICSE 2017) Warm-up Workshop, Maceio, Brazil, October 2014.
- P27** *An ARCADE for Architecture Analytics.* Dagstuhl Seminar on Software Development Analytics, Dagstuhl, Germany, June 2014.
- P28** *Component-Aware Triggered Scenarios.* Working IEEE/IFIP Conference on Software Architecture (WICSA 2014), Sydney, Australia, April 2014.
- P29** *When, Where, and Why Software Systems Decay and What We Can Do about It.* International Symposium to Commemorate the 45th Anniversary of the First Software Engineering Conference (SE@45), University of Southern California, Los Angeles, CA, October 2013.
- P30** *Platform Architectures.* Dagstuhl Seminar on Customizing Service Platforms, Dagstuhl, Germany, April 2013.
- P31** *When, Where, and Why Do Software Systems' Architectures Decay?* Workshop on Modularity and Software Architecture, Rio de Janeiro, Brazil, March 2013.
- P32** *Impact Analysis for Distributed Event-Based Systems.* 6th ACM International Conference on Distributed Event-Based Systems (DEBS 2012), Berlin, Germany, July 2012.
- P33** *The Impact of Architectural Design Decisions on Non-Functional Properties of Software Systems.* OOP 2010 - Software Engineering and Management Conference, Munich, Germany, January 2010.
- P34** *A Framework for Modeling Software Product Lines.* OOP 2010 - Software Engineering and Management Conference, Munich, Germany, January 2010.
- P35** *LVC Software Architecture Support for PATFrame.* International Test and Evaluation Association (ITEA) Live Virtual Constructive Conference, El Paso, TX, January 2010.
- P36** *The Anatomy and Physiology of the Grid Revisited.* IEEE/CSSE/ISR Workshop on Software Architecture Challenges for the 21st Century, Los Angeles, CA, June 2009.
- P37** *Center for Systems and Software Engineering Overview.* IEEE/CSSE/ISR Workshop on Software Architecture Challenges for the 21st Century, Los Angeles, CA, June 2009.

- P38** *Architecture-Based Software Development of Mobile and Embedded Systems*. 2nd International ICSE Warm-up Workshop (WUP 2009), Cape Town, South Africa, April 2009.
- P39** *Service-Oriented Architectures*. 13th Ground System Architectures Workshop (GSAW 2009), Torrance, CA, March 2009.
- P40** *Architecture and Mobile/Embedded Systems - An Uneasy Alliance or a Marriage Made in Heaven*. USC-CSSE Annual Research Review Executive Workshop, Los Angeles CA, March 2009.
- P41** *Software Architecture Modeling with Uncertainty*. AADL Workshop, Los Angeles, CA, February 2009.
- P42** *Architecture and Hardware/Software/System Engineering Integration*. USC-CSSE COCOMO Workshop, Los Angeles, CA, October 2008.
- P43** *Runtime Software Adaptation: Framework, Approaches, and Styles*. (with Peyman Oreizy and Richard N. Taylor) 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 2008.
- P44** *Estimating the Energy Consumption in Pervasive Java-Based Systems*. 6th IEEE International Conference on Pervasive Computing and Communications (PerCom 2008), Hong Kong, March 2008.
- P45** *An Introduction to Architecture-Level Reliability Analysis*. Tutorial, with Roshanak Roshandel, Working IEEE/IFIP Conference on Software Architecture (WICSA), Vancouver, Canada, February 2008.
- P46** *Software Architectures and Embedded Systems*. Korean Computer Scientists and Engineers Association in America, University of Southern California, Los Angeles, CA, December 2007.
- P47** *Improving a Distributed Software System's Quality of Service via Architecture-Driven Dynamic Redeployment*. Tutorial, Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS 2007), Campinas, Brazil, August 2007.
- P48** *Construction of Analytic Frameworks for Component-Based Architectures*. Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS 2007), Campinas, Brazil, August 2007.
- P49** *From Dependable Architectures to Dependable Systems*. USC-CSSE Annual Research Review, Los Angeles, CA, February 2007.
- P50** *A Classification of Disconnected Operation Techniques*. 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), Dubrovnik, Croatia, August, 2006.
- P51** *Improving a Distributed Software System's Quality of Service via Redeployment*. Workshop on State of the Art in Software Engineering, Rutgers University, New Brunswick, NJ, June 2006.
- P52** *The GridLite DREAM: Bringing the Grid to Your Pocket*. Monterey Workshop, Laguna Beach, CA, September 22-23, 2005.

- P53** *A View from the Trenches*. ICSE 2005 New Software Engineering Faculty Symposium, St. Louis, MO, May 2005.
- P54** *Modeling Behavior in Compositions of Software Architectural Primitives*. 19th IEEE International Conference on Automated Software Engineering (ASE 2004), Linz, Austria, September 2004.
- P55** *Toward Architecture-Based Reliability Estimation*. Workshop on Architecting Dependable Systems (WADS 2004) held in conjunction with the International Conference on Dependable Systems and Networks (DSN 2004), Florence, Italy, June 2004.
- P56** *Style-Based Software Architectural Compositions as Domain-Specific Models*. Workshop on Directions in Software Engineering Environments (WoDiSEE 2004), Edinburgh, UK, May 2004.
- P57** *Software Architectures and Embedded Systems*. Monterey Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation, Chicago, IL, September 2003.
- P58** *Coupling Architectural Discovery and Recovery to Stem Architectural Erosion*. Dagstuhl Workshop on Software Architecture Recovery and Modeling (SWARM), Dagstuhl, Germany, February 2003.
- P59** *A Family of Software Architecture Implementation Frameworks*. 2002 Working IEEE/IFIP Conference on Software Architectures (WICSA-3), Montreal, Canada, August 2002.
- P60** *On the Role of Middleware in Architecture-Based Software Development*. 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 2002), Ischia, Italy, July 2002.
- P61** *Architecture-Level Support for Software Component Deployment in Resource Constrained Environments*. IFIP/ACM Working Conference on Component Deployment (CD 2002), Berlin, Germany, June 2002.
- P62** *Focus: A Light-Weight, Incremental Approach to Software Architecture Recovery and Evolution*. 2001 Working IEEE/IFIP Conference on Software Architectures (WICSA-2), Amsterdam, the Netherlands, August 2001.
- P63** *Software Model Connectors: Bridging Models across the Software Lifecycle*. 13th International Conference on Software Engineering and Knowledge Engineering (SEKE 2001), Buenos Aires, Argentina, June 2001.
- P64** *What Young Faculty Members Still Want to Know*. ICSE 2001 New Software Engineering Faculty Symposium, Toronto, Canada, May 2001.
- P65** *Exploiting Software Architecture Implementation Infrastructure in Facilitating Component Mobility*. ICSE 2001 Workshop on Software Engineering and Mobility, Toronto, Canada, May 2001.
- P66** *xADL: Enabling Architecture-Centric Tool Integration With XML*. 34th Hawaii International Conference on System Sciences (HICSS-34), Maui, Hawaii, January 2001.
- P67** *Programming in the Small and Many: A Software Engineering Research Agenda for the 21st Century*. 2000 ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-8), Wild Ideas and Fun Flames, San Diego, CA, November 2000.

- P68** *Towards Software Multioperability: Bridging Heterogeneous Software Interoperability Platforms.* Fourth International Software Architecture Workshop (ISAW-4), Limerick, Ireland, June 2000.
- P69** *Using Off-the-Shelf Middleware to Implement Connectors in Distributed Software Architectures.* 21st International Conference on Software Engineering (ICSE'99), Los Angeles, CA, May 1999.
- P70** *A Language and Environment for Architecture-Based Software Development and Evolution.* 21st International Conference on Software Engineering (ICSE'99), Los Angeles, CA, May 1999.
- P71** *Modeling Software Architectures in UML.* Workshop on Software Architectures and the Unified Modeling Language, Denver, CO, April 1999.
- P72** *An Approach to Architecture-Based Software Integration.* Ground System Architectures Workshop (GSAW 99), El Segundo, CA, March 1999.
- P73** *Assessing the Suitability of a Standard Design Method for Modeling Software Architectures.* First Working IFIP Conference on Software Architecture, San Antonio, TX, February 1999.
- P74** *A Comparative Analysis of Architecture Description Languages.* USC-CSE Annual Research Review, Los Angeles, CA, February 1999.
- P75** *Assessing UML's Suitability for Modeling Software Architectures.* USC-CSE Annual Research Review, Los Angeles, CA, February 1999.
- P76** *Employing Off-the-Shelf Connector Technologies in C2-Style Architectures.* California Software Symposium (CSS'98), Irvine, CA, October 1998.
- P77** *An Architecture-Based Approach to Software Evolution.* International Workshop on the Principles of Software Evolution, Kyoto, Japan, April 1998.
- P78** *Domains of Concern in Software Architectures and Architecture Description Languages.* USENIX Conference on Domain-Specific Languages, Santa Barbara, CA, October 1997.
- P79** *A Framework for Classifying and Comparing Architecture Description Languages.* Sixth European Software Engineering Conference together with the Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'97), Zurich, Switzerland, September 1997.
- P80** *Reuse of Off-the-Shelf Components in C2-Style Architectures.* 1997 Symposium on Software Reusability (SSR'97) and 1997 International Conference on Software Engineering (ICSE'97), joint session on Software Reuse, Boston, MA, May 1997.
- P81** *Reusing Off-the-Shelf Components to Develop a Family of Applications in the C2 Architectural Style.* First International Workshop on Development and Evolution of Software Architectures for Product Families, Las Navas del Marqués, Spain, November 1996.
- P82** *ADLs and Dynamic Architecture Changes.* Second International Software Architecture Workshop (ISAW-2), San Francisco, CA, October 1996.

- P83** *Using Object-Oriented Typing to Support Architectural Design in the C2 Style*. Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE4), San Francisco, CA, October 1996.
- P84** *Formal Modeling of Software Architectures at Multiple Levels of Abstraction*. California Software Symposium, Los Angeles, CA, April 1996.
- P85** *A Component- and Message-Based Architectural Style for GUI Software*. Technology Presentation Track, 17th International Conference on Software Engineering (ICSE17), Seattle, WA, April 1995.

University and Research Institute Presentations

- P86** *Mining Architectural Information to Stem Technical Debt*. Georgia Institute of Technology, Atlanta, GA, November 2019
- P87** *Mining Architectural Information to Stem Technical Debt*. University College London, UK, July 2019
- P88** *Loose Coupling in Mobile Systems: Challenges and Opportunities*. Harbin Institute of Technology, Harbin, China, April 2019.
- P89** *Mining Architectural Information to Stem Technical Debt*. Harbin Institute of Technology, Harbin, China, April 2019.
- P90** *Teaching CS and SE at USC*. Harbin Institute of Technology, Harbin, China, April 2019.
- P91** *Publishing Strategies for Top SE Venues*. Harbin Institute of Technology, Harbin, China, April 2019.
- P92** *Ph.D. Research: What, Where, Why, When, How*. Harbin Institute of Technology, Harbin, China, April 2019.
- P93** *Introduction to Software Architecture*. Faculty of Organizational Sciences, University of Belgrade, Serbia, March 2017.
- P94** *Architectural Decay in Software Systems: Symptoms, Causes, and Remedies*. University of Massachusetts, Amherst, MA, November 2013.
- P95** *Architectural Decay in Software Systems: Symptoms, Causes, and Remedies*. McGill University, Montreal, Canada, October 2013.
- P96** *When, Where, and Why Do Software Systems' Architectures Decay?* University of Lugano, Lugano, Switzerland, October 2012.
- P97** *When, Where, and Why Do Software Systems' Architectures Decay?* University of Arizona, Tucson, AZ, October 2012.
- P98** *Architectural Degradation - The Plague of Maturing Software Systems*. University of California Irvine Institute for Software Research Annual Research Review, Irvine, CA, May 2012.
- P99** *Overview of Software Engineering Research at USC*. Computer Science Department Annual Research Review, Los Angeles, CA, March 2012.
- P100** *A Framework for Maintaining a Distributed Software System's QoS*. Computer Science Department Annual Research Review, USC, Los Angeles, March 2010.

- P101** *Academic Career Work-Life Balance - A View from the Trenches*. University of Southern California, April 2011.
- P102** *Overview of Software Engineering Research at USC*. University of Southern California, Computer Science Research Day, March 2011.
- P103** *A Framework for Early Estimation of Software System Reliability*. University of California, San Diego, February 2011.
- P104** *Software Architecture and Mobility: A Perfect Marriage or an Uneasy Alliance?* University College, London, UK, October 2010.
- P105** *A Framework for Early Estimation of Software System Reliability*. University of Zurich, Switzerland, September 2010.
- P106** *The Future of Software Architecture: Challenges and Opportunities*. Federal University of Bahia, Salvador, Brazil, August 2010.
- P107** *Software Architecture and Mobility: A Perfect Marriage or an Uneasy Alliance?* Federal University of Rio de Janeiro, Brazil, August 2010.
- P108** *Software Architecture and Mobility: A Perfect Marriage or an Uneasy Alliance?* Pontificia Universidade Católica do Rio de Janeiro, August 2010.
- P109** *A Framework for Early Estimation of Software System Reliability*. Georgia Institute of Technology, Atlanta, GA, April 2010.
- P110** *Software Architecture and Mobility: A Perfect Marriage or an Uneasy Alliance*. Institute for Software Research, Distinguished Speaker Series, University of California, Irvine, February 2010.
- P111** *Improving a Distributed Software System's Quality of Service via Architecture-Driven Dynamic Redeployment*. Institute for Software Research Forum, University of California, Irvine, June 2007.
- P112** *Dynamic Software Architectures*. Jesper Andersson Dissertation Defense Opponent, Linköping University, Sweden, April 2007.
- P113** *From Dependable Architectures to Dependable Systems*. USC-CSE Annual Research Review, Los Angeles, CA, February 2007.
- P114** *Moving Architectural Description from Under the Technology Lamppost*. USC Center for Systems and Software Engineering Convocation, October 2006.
- P115** *Endowing Legacy Applications with Software Architectural Capabilities*. University of Southern California - Information Sciences Institute, Marina del Rey, CA, October 2006.
- P116** *The Prism Project: Software Architectural Support for Highly Distributed Environments*. Bosch Research and Technology Center, Palo Alto, CA, January 2006.
- P117** *Coupling Architectural Discovery and Recovery to Stem Architectural Erosion*. USC-CSE Annual Research Review, Los Angeles, CA, March 2003.
- P118** *Exploiting Architectural Style, Connectors, and Types in Large-Scale Software Development and Evolution*. University of Texas, Austin, TX, April 2001.

- P119** *An Architecture-Based Approach to Developing Heterogeneous, Highly Distributed Software Systems.* University of Southern California - Information Sciences Institute, Marina del Rey, CA, February 2001.
- P120** *Programming in the Many: Software Engineering Paradigm for the 21st Century.* USC-CSE Annual Research Review, Los Angeles, CA, February 2001.
- P121** *An Architecture-Based Approach to Developing Heterogeneous, Highly Distributed Software Systems.* University of Washington, Seattle, WA, January 2001.
- P122** *From Programming in the Large to Programming in the Small and Many.* University of California, Santa Cruz, CA, October 2000.
- P123** *Exploiting Architectural Style, Connectors, and Types in Large-Scale Software Development and Evolution.* Rice University, Houston, TX, September 2000.
- P124** *SAAGE — An Environment for Software Architecture, Analysis, Generation, and Evolution.* USC-CSE Annual Research Review, Los Angeles, CA, February 2000.
- P125** *A Style-Based Approach to the Design and Evolution of Adaptable Software.* University of Texas, Austin, TX, November 1997.

Professional Organization, Industry, and Funding Agency Presentations

- P126** *On Selecting Datasets for DL-based Vulnerability Detection.* 68th Meeting of the IFIP Working Group 2.4, online, November 2023.
- P127** *On Automated Mobile-App Test Generation and Transfer.* 66th Meeting of the IFIP Working Group 2.4, Salzburg, Austria, July 2022.
- P128** *Protection of Intellectual Property Through the Prism of Software Engineering.* Logate Institute Live Meetup, Podgorica Montenegro, online, May 2022.
- P129** *Classifying and Identifying Noise in Software Patches.* 64th Meeting of IFIP Working Group 2.4, online, May 2021.
- P130** *Events in Mobile Systems: A Blessing and a Curse.* 62nd Meeting of IFIP Working Group 2.4, Port Elizabeth, South Africa, January 2020.
- P131** *Architectural Decay in Open-Source Software,* Google, Irvine, CA, October 2019.
- P132** *Ensuring Security of Android Software via Tailorable Multi-Layer Customization.* Office of Naval Research (ONR) Total Platform Cyber Protection (TPCP) Program Annual Review, Boston, MA, June 2019.
- P133** *Mining Architectural Information to Stem Technical Debt.* Google, Los Angeles, CA, June 2019.
- P134** *Architectural Decay in Large Software Systems.* Google, Los Angeles, CA, November 2017.
- P135** *Correlating Implementation Issues with Architectural Decay in Large Software Systems.* Huawei Vision Forum, Santa Clara, CA, October 2017.
- P136** *Using Architecture Recovery to Stem Software System Decay.* Huawei Vision Forum, Santa Clara, CA, October 2016.

- P137** *Software Architecture Recovery, Event-Based App Analysis, and Mobile Security*. Huawei Technologies Site Visit, University of Southern California, Los Angeles, CA, June 2016.
- P138** *Correctness, Security, and Adaptation in Distributed Event-Based Systems*. Army Research Laboratories Site Visit, University of Southern California, Los Angeles, CA, April 2016.
- P139** *The Flame Project*. Infosys Site Visit, University of Southern California, Los Angeles, CA, February 2016.
- P140** *Architectural Decay in Software Systems: Symptoms, Causes, and Remedies*. United Technologies Research Center, Hartford, CT, October 2013.
- P141** *sTile: Securing the Insecure Cloud*. DARPA Site Visit, Information Sciences Institute, Marina del Rey, CA, May 2012.
- P142** *CoDesign - A Collaborative Software Design Environment*. Infosys Site Visit, Los Angeles, CA, May 2012.
- P143** *Software Architecture Challenges & Opportunities for the 21st Century: Dynamism, Mobility, Autonomy, Services, Grids & Clouds*. Infosys, Mysore, India, March 2012.
- P144** *The Anatomy and Physiology of the Grid Revisited*. Infosys, Bangalore, India, March 2012.
- P145** *Software Architecture Challenges & Opportunities for the 21st Century: Dynamism, Mobility, Autonomy, Services, Grids & Clouds*. IBM Research, Delhi, India, February 2012.
- P146** *sTile: Securing the Insecure Cloud*. CIA, Washington, DC, November 2011.
- P147** *sTile: Securing the Insecure Cloud*. DARPA, Arlington, VA, June 2011.
- P148** *Desiderata and Challenges in Globally-Distributed Software Development*. Infosys Aurora, Las Vegas, NV, August 2011.
- P149** *Modern Software Architectural Challenges: Dynamism, Mobility, Services, Grids, and Clouds*. Aerospace Corporation, El Segundo, CA, November 2010.
- P150** *Software Architecture and Mobility: A Perfect Marriage or an Uneasy Alliance?* Northrop Grumman, Los Angeles, CA, September 2010.
- P151** *Software Architecture and Mobility: A Perfect Marriage or an Uneasy Alliance?* Infosys, Bangalore, India, December 2009.
- P152** *Software Architecture Support for PATFrame*. PATFrame project kick-off meeting, Ft. Hood, TX, August 2009.
- P153** *An Extensible Infrastructure for Collaborative Architecture-Based Development of Distributed Software-Intensive Systems*. Infosys site visit, April 2009.
- P154** *Architecture and Hardware/Software/System Engineering Integration*. Lockheed Martin Site Visit, Los Angeles, CA, December 2008.
- P155** *An Extensible Infrastructure for Collaborative Architecture-Based Development of Distributed Software-Intensive Systems*. Infosys project progress meeting, August 2008.
- P156** *An Extensible Infrastructure for Collaborative Architecture-Based Development of Distributed Software-Intensive Systems*. Infosys project kick-off meeting, August 2008.

- P157** *Architecture-Based Modeling and Simulation*. Boeing Professor Day, Seal Beach, CA, February 2008.
- P158** *Overview of Software Architecture Research at USC*. Bosch Research and Technology Center, Palo Alto, CA, January 2007.
- P159** *Identifying and Addressing Uncertainty in Architecture-Level Software Reliability Modeling*. NSF Next Generation Software Program (NSFNGS) Workshop, Long Beach, CA, February 2007.
- P160** *Multi-Framework Programming via Software Architecture and Architectural Middleware*. DARPA Multi-Framework Programming Workshop, Menlo Park, CA, August 2007.
- P161** *Overview of Software Architecture Research at USC*. USC Computer Science Department Industrial Advisory Board Meeting, Los Angeles, CA, April 2006.
- P162** *Overview of Software Architecture Research at USC*. General Electric Site Visit, Los Angeles, CA, February 2006.
- P163** *Software Architecture for DREAM Environments*. Jet Propulsion Laboratory, Pasadena, CA, January 2005.
- P164** *DeSi - An Environment for Modeling, Analyzing, and Simulating Deployment Architectures*. Boeing, Anaheim, November 2004.
- P165** *Overview of Software Architecture Research at USC*. Northrop Grumman Site Visit, Los Angeles, CA, April 2004.
- P166** *Overview of Software Architecture Research at USC*. Microsoft Site Visit, Los Angeles, CA, March 2004.
- P167** *Overview of Software Architecture Research at USC*. Airbus Site Visit, Los Angeles, CA, February 2004.
- P168** *DeSi - A Tailorable Environment for Assessing the Quality of Distributed Software Architectures*. Jet Propulsion Laboratory, Pasadena, CA, January 2004.
- P169** *Overview of Software Engineering Research at USC*. Boeing Site Visit, Los Angeles, CA, November 2003.
- P170** *Software Architecture Research at USC-CSE: An Overview*. Jet Propulsion Laboratory Site Visit, Los Angeles, CA, September 2003.
- P171** *Software Architectures and Self-Healing Systems*. Boeing, Anaheim, CA, May 2003.
- P172** *Software Architectures and Self-Healing Systems*. Boeing-DARPA Future Combat Systems Meeting, Boeing, Anaheim, CA, September 2002.
- P173** *Software Connectors for Programming-in-the-Many*. Boeing, Seattle, WA, November 2001.
- P174** *Software Architecture Collaboration between JPL and USC*. Jet Propulsion Laboratory, Pasadena, CA, September 2001.
- P175** *Relating State-Based and Component/Connector-Based Architectural Styles: Technical Project Review*. Jet Propulsion Laboratory, Pasadena, CA, August 2001.
- P176** *Architectural Support for Programming-in-the-Many*. U.S. Army Tank-Automotive and Armaments Command (TACOM), Detroit, MI, July 2001.

- P177** *Software Architecture Research at USC: Programming in the Large and Many*. Xerox Corporation, El Segundo, CA, January 2001.
- P178** *From Programming in the Large to Programming in the Small and Many*. Microsoft USC Site Visit, Los Angeles, December 2000.
- P179** *Software Architecture at USC and JPL*. Jet Propulsion Laboratory, Pasadena, CA, October 2000.
- P180** *Programming in the Small, Large, and Many*. Xerox Corporation, El Segundo, CA, October 2000.
- P181** *Programming in the Small, Large, and Many*. Schlumberger Corporation, Houston, TX, September 2000.
- P182** *Dynamic Assembly, Assessment, Assurance, and Adaptation via Heterogeneous Software Connectors*. DARPA/ISO DASADA Project Kick-off Meeting, Santa Fe, NM, September 2000.
- P183** *UML and Software Architectures*. DARPA/ISO DASADA Project Kick-off Meeting, Santa Fe, NM, September 2000.
- P184** *An Approach to Architecture-Based Software Development and Evolution*. Jet Propulsion Laboratory, Pasadena, CA, May 2000.
- P185** *SAAGE — An Environment for Software Architecture, Analysis, Generation, and Evolution*. Lockheed Martin USC Site Visit, Los Angeles, CA, May 2000.
- P186** *On the Role of UML in Modeling Software Architectures*. Southern California Software Process Improvement Network, Long Beach, CA, April 2000.
- P187** *Software Architecture Research and Practice*. Jet Propulsion Laboratory, Pasadena, CA, April 2000.
- P188** *Architecture-Centric Dynamic Adaptation of Deeply Networked Embedded Systems*. DARPA/ITO RENES Project Kick-off Meeting, Arlington, VA, November 1999.
- P189** *Software Architecture Research and Practice*. NSF Software Engineering Strategies Workshop Briefing, National Science Foundation, Arlington, VA, September 1999.
- P190** *Assessing the Suitability of UML for Modeling Software Architectures*. Bay Area Round Table (BART), Palo Alto, CA, July 1999.
- P191** *A Classification and Comparison Framework for Software Architecture Description Languages*. Second DARPA EDCS Architecture/Generation Workshop, Santa Fe, NM, April 1997.
- P192** *Component-Based GUI Architecture*. Hughes Aircraft Corp., El Segundo, CA, January 1997.
- P193** *What's New with C2?* Arcadia Research Meeting, Portland, OR, August 1996.
- P194** *Chiron-2: A Component- and Message-Based Architectural Style for GUI Software*. Sun Microsystems Laboratories, Palo Alto, CA, June 1995.
- P195** *Chiron-2: A Component- and Message-Based Architectural Style for GUI Software*. Bay Area Round Table (BART), Palo Alto, CA, June 1995.

12 Professional Service

Professional Organization and Journal Leadership

- S1** Editor-in-Chief, IEEE Transactions on Software Engineering (1/18 - 12/22)
- S2** Chair, ACM Software System Award Committee (7/21 - 6/22)
- S3** Deputy Chair, ACM Software System Award Committee (7/20 - 6/21)
- S4** Chair, Editor-in-Chief Search Committee, ACM Transactions on Autonomous and Adaptive Systems (6/20 - 9/20)
- S5** Chair, ACM Publications Board Task Force on Improving Peer-Review Incentives (1/19 - 7/20)
- S6** Chair, ACM SIGSOFT (7/15 - 6/18)
- S7** Chair, Committee on Capture/Streaming at ACM Conferences, ACM SIG Governing Board (10/16 - 6/18)
- S8** Chair, Steering Committee, ACM SIGSOFT Symposium on the Foundations of Software Engineering (7/15 - 9/17)
- S9** Chair, Steering Committee, International Conference on Software Engineering (11/13 - 10/15)

Editorial Boards and Steering Committee Membership

- S10** Member, Joint ICSE/FSE/ASE Conference Consolidation Committee (7/20 - present)
- S11** Member, Executive Committee, ACM Special Interest Group on Software Engineering (SIGSOFT) (7/18 – 6/24)
- S12** Member, Steering Committee, International Conference on Software Engineering (5/08 - present)
- S13** Member, Steering Committee, ACM International Conference on the Foundations of Software Engineering (FSE) (5/17 – 6/24)
- S14** Member, ICSE Journal-First Task Force (5/16 - 12/23)
- S15** Member, ACM SIG Governing Board (7/15 - 6/20)
- S16** Member, ACM Publications Board (7/18 - 6/20)
- S17** Guest Editor, IEEE Software, 50th Anniversary of Software Engineering (6/17 - 10/18)
- S18** Associate Editor, ACM Transactions on Software Engineering and Methodology (3/14 - 12/17)
- S19** Associate Editor, Journal of Software Engineering for Robotics (12/08 - 12/17)
- S20** Editorial Board Member, Journal of Software Engineering Research and Development (4/12 - 12/17)
- S21** Editorial Board Member, Springer Computing Journal (2/11 - present)
- S22** Editorial Board Member, Elsevier Journal of Systems and Software (1/11 - 12/17)

- S23** Editorial Board Member, Elsevier Information and Software Technology Journal (1/08 - 12/17)
- S24** Associate Editor, IEEE Transactions on Software Engineering (1/10 - 12/13)
- S25** Guest Editor, Elsevier Journal of Systems and Software, The Future of Software Engineering FOR/IN the Cloud (7/11 - 7/13)
- S26** Guest Editor, The Science of Computer Programming Journal Best Papers of the CompArch 2012 Conference (7/12 - 9/14)
- S27** Member, Steering Committee, Working IEEE/IFIP Conference on Software Architecture (1/05 - 2/08)

Advisory Boards and Award Committees

- S28** Member, Graduate Program Review Committee, Computer Science, University of California Riverside (2024 - 2025)
- S29** Co-Chair, ICSE 2011 Most Influential Paper Award (3/19 - 5/21)
- S30** Member, ACM Software System Award Committee (9/18 - 7/20)
- S31** Member, Selection Committee, 2016 ACM SIGSOFT Impact Paper Award (12/16 - 5/17)
- S32** Member, Selection Committee, 2015 ACM SIGSOFT Impact Paper Award (12/15 - 5/16)
- S33** Member, Selection Committee, 2014 ACM SIGSOFT Outstanding Research Award (8/13 - 5/14)
- S34** Member, International Selection Committee, Bower Award for Achievement in Science (8/13 - 11/13)
- S35** Member, President's Advisory Board, Carnegie Mellon University - Silicon Valley (1/13 - 4/13)

Conference/Workshop Chairmanship

- S36** General Co-Chair, 21st IEEE International Conference on Software Architecture (ICSA 2024), Hyderabad, India, 2024
- S37** Chair, Barry Boehm Memorial Symposium, Los Angeles, CA, September 2014
- S38** Co-Chair, ICSE Showcase, 45th International Conference on Software Engineering (ICSE 2023), Melbourne, Australia, May 2023
- S39** Co-Chair, Doctoral Symposium, 44th International Conference on Software Engineering (ICSE 2022), Pittsburgh, PA, May 2022
- S40** Co-Chair, 2nd International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE 2019), Montreal, Canada, May 2019
- S41** Co-Chair, 50 Years of Software Engineering Track, 40th International Conference on Software Engineering (ICSE 2018), Gothenburg, Sweden, May 2018
- S42** Co-Chair, 2nd International Workshop on Infrastructures and Instruments For Software Architecture (REINFORCE), Urbana-Champaign, IL, November 2017

- S43** Co-Chair, 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE 2017), Buenos Aires, Argentina, May 2017
- S44** Program Co-Chair, Innovations in Software Engineering Conference (ISEC 2017), Jaipur, India, February 2017
- S45** Co-Chair, 1st Workshop on Infrastructures and Instruments For Software Architecture (REINFORCE), Los Angeles, CA, January 2017
- S46** Co-Chair, Workshops, 36th International Conference on Software Engineering (ICSE 2014), Hyderabad, India, June 2014
- S47** Chair, International Symposium to Commemorate the 45th Anniversary of the First Software Engineering Conference (SE@45), University of Southern California, Los Angeles, CA, October 2013
- S48** Program Co-Chair, 15th International ACM SIGSOFT Symposium on Component Based Software Engineering (CBSE-2012), Bertinoro, Italy, June 2012
- S49** Program Co-Chair, 33rd International Conference on Software Engineering (ICSE 2011), Honolulu, HI, May 2011
- S50** Program Co-Chair, Workshop on Software Engineering for Cloud Computing (SECLOUD 2011), Honolulu, HI, May 2011
- S51** Program Co-Chair, 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2009), San Francisco, CA, September 14-18, 2009
- S52** Workshop Co-Chair, IEEE/CSSE/ISE Workshop on Software Architecture Challenges for the 21st Century, Los Angeles, CA, June 2009
- S53** Program Co-Chair, 2nd International ICSE Warm-up Workshop (WUP 2009), Cape Town, South Africa, April 2009
- S54** Workshop Chair, USC-CSSE Annual Research Review and Executive Workshop, Los Angeles, CA, March 2009
- S55** Program Chair, Doctoral Symposium, 16th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 16), Atlanta, GA, November 2008
- S56** Program Chair, Special Session on Software Architecture for Pervasive Systems (SAPS), 34th Euromicro Conference in Parma, Italy, September 3-5, 2008
- S57** Program Co-Chair, 2005 Working IEEE/IFIP Conference on Software Architecture (WICSA 2005), Pittsburgh, PA, November 2005
- S58** Chair, Executive Workshop on Model-Driven Architecture, Annual Research Review, Center for Software Engineering, University of Southern California, Los Angeles, CA, March 2005
- S59** Chair, Workshops, 26th International Conference on Software Engineering (ICSE 2004), Edinburgh, UK, May 2004
- S60** Co-Chair, Tutorials, 9th European Software Engineering Conference and 11th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2003), Helsinki, Finland, September 2003

- S61** Chair, State-of-the-Art Track, 24th International Conference on Software Engineering (ICSE 2002), Orlando, FL, May 2002
- S62** Co-Chair, First International Workshop on Evaluating Software Architecture Solutions (WESAS), Irvine, CA, May 2000

Conference/Workshop Organization

- S63** Session Co-Chair, Computing Research Association (CRA) Summit, Affinity Group Meeting, R1 Private Universities, San Francisco, CA, July 2025
- S64** Organizing Committee Member, First International Workshop on Bringing Architecture Design Thinking into Developers' Daily Activities (Bridge 2016), Austin, TX, May 2016
- S65** Organizing Committee Member, 39th International Conference on Software Engineering (ICSE 2017) Warm-up Workshop, Maceio, Brazil, October 2014
- S66** Technical Paper Session Chair, 22nd ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 2014), Hong Kong, November 2014
- S67** Workshops Selection Committee Member, 8th Joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2011), Szeged, Hungary, September 2011
- S68** Technical Paper Session Chair, 12th European Software Engineering Conference and 17th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2009), Amsterdam, the Netherlands, August 2009
- S69** Student Volunteer Coordinator, 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 2008
- S70** Technical Paper Session Chair, Working IEEE/IFIP Conference on Software Architecture (WICSA 2008), Vancouver, Canada, February 2008
- S71** E-Publicity Chair, 29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, May 2007
- S72** Session Chair, USC Center for Systems and Software Engineering Convocation, Los Angeles, CA, October 2006
- S73** Panelist, New Software Engineering Faculty Symposium (NSEFS 05), 27th International Conference on Software Engineering (ICSE 2005), St. Louis, MO, May 2005
- S74** Technical Paper Session Chair, 19th IEEE International Conference on Automated Software Engineering (ASE 2004), Linz, Austria, September 2004
- S75** Session Organizer and Chair, Annual Research Review, Center for Software Engineering, University of Southern California, Los Angeles, CA, March 2004
- S76** Session Organizer and Chair, Annual Research Review, Center for Software Engineering, University of Southern California, Los Angeles, CA, March 2003
- S77** Session Chair, Dagstuhl Workshop on Software Architecture Recovery and Modeling (SWARM), Dagstuhl, Germany, February 2003
- S78** Chair, Tutorials, The Third Working IEEE/IFIP Conference on Software Architecture (WICSA-3), Montreal, Canada, August 2002

- S79** Technical Paper Session Chair, The Working IEEE/IFIP Conference on Software Architecture (WICSA-3), Montreal, Canada, August 2002
- S80** Technical Paper Session Chair, 24th International Conference on Software Engineering (ICSE 2002), Orlando, FL, May 2002
- S81** Session Organizer and Chair, Annual Research Review, Center for Software Engineering, University of Southern California, Los Angeles, CA, March 2002
- S82** Technical Paper Session Chair, Eighth European Software Engineering Conference and Ninth ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2001), Vienna, Austria, September 2001
- S83** Technical Paper Session Chair, 23rd International Conference on Software Engineering (ICSE 2001), Toronto, Canada, May 2001
- S84** Session Organizer and Chair, Annual Research Review, Center for Software Engineering, University of Southern California, Los Angeles, CA, March 2001
- S85** Technical Paper Session Chair, Eighth ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE8), San Diego, CA, November 2000
- S86** Session Organizer and Chair, Annual Research Review, Center for Software Engineering, University of Southern California, Los Angeles, CA, March 2000
- S87** Organizing Committee Member, NSF Workshop on Software Engineering Research Strategies, Los Angeles, CA, August 1999
- S88** Student Volunteer Coordinator, 21st International Conference on Software Engineering (ICSE'99), Los Angeles, CA, May 1999
- S89** Co-Chair, DARPA/ITO EDCS Workshop on Architecture Description Language Toolkits, San Diego, CA, October 1998
- S90** Chair, DARPA/ITO EDCS Workshop on the UML, Austin, TX, November 1997
- S91** Webmaster, 1997 International Conference on Software Engineering (ICSE'97), Boston, MA, May 1997

Program Committee Membership

- S92** Program Committee Member, 45th International Conference on Software Engineering (ICSE 2023), Melbourne, Australia, May 2023
- S93** Program Committee Member, 44th International Conference on Software Engineering (ICSE 2022), Pittsburgh, PA, May 2022
- S94** Program Committee Member, 5th IEEE International Conference on Software Architecture (ICSA 2021), Stuttgart, Germany, March 2021
- S95** Program Committee Member, Journal-First Track, International Conference on Software Engineering (ICSE 2020), Seoul, South Korea, May 2020
- S96** Program Committee Member, 4th IEEE International Conference on Software Architecture (ICSA 2020), Salvador, Brazil, March 2020

- S97** Program Committee Member, ACM SIGSOFT Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2019), Tallinn, Estonia, August 2019
- S98** Program Committee Member, Journal-First Track, International Conference on Software Engineering (ICSE 2019), Montreal, Canada, May 2019
- S99** Program Committee Member, ACM SIGSOFT Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018), Lake Buena Vista, FL, November 2018
- S100** Program Committee Member, 40th International Conference on Software Engineering (ICSE 2018), Gothenburg, Sweden, May 2018
- S101** Program Committee Member, International Workshop on Software Fairness (FairWare 2018), Gothenburg, Sweden, May 2018
- S102** Program Committee Member, 2nd IEEE International Conference on Software Architecture (ICSA 2018), Seattle, WA, April 2018
- S103** Program Committee Member, Innovations in Software Engineering Conference (ISEC 2018), Hyderabad, India, February 2018
- S104** Symposium Committee Member, Doctoral Symposium, 11th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2017), Paderborn, Germany, September 2017
- S105** Symposium Committee Member, Doctoral Symposium, 39th International Conference on Software Engineering (ICSE 2017), Buenos Aires, Argentina, May 2017
- S106** Program Board Member, 39th International Conference on Software Engineering (ICSE 2017), Buenos Aires, Argentina, May 2017
- S107** Symposium Committee Member, Doctoral Symposium, 39th International Conference on Software Engineering (ICSE 2017), Buenos Aires, Argentina, May 2017
- S108** Program Committee Member, 1st IEEE International Conference on Software Architecture (ICSA 2017), Gothenburg, Sweden, April 2017
- S109** Mentoring Committee Member, 24th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 2016), Seattle, WA, November 2016
- S110** Program Committee Member, 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2016), Austin, TX, May 2016
- S111** Program Committee Member, Joint 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), 19th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE), and 12th International ACM SIGSOFT Conference on the Quality of Software Architectures (QoSA), Venice, Italy, April 2016
- S112** Program Committee Member, 9th Annual India Software Engineering Conference (ISEC 2016), Goa, India, February 2016
- S113** Program Board Member, 38th International Conference on Software Engineering (ICSE 2016), Austin, TX, May 2016

- S114** Program Committee Member, 5th International Workshop on the Twin Peaks of Requirements and Architecture, Florence, Italy, May 2015
- S115** Symposium Committee Member, Doctoral Symposium, 37th International Conference on Software Engineering (ICSE 2015), Florence, Italy, May 2015
- S116** Program Committee Member, Technical Briefings, 37th International Conference on Software Engineering (ICSE 2015), Florence, Italy, May 2015
- S117** Program Committee Member, 22nd ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 2014), Hong Kong, November 2014
- S118** Program Committee Member, 8th European Conference on Software Architecture (ECSA 2014), Vienna, Austria, August 2014
- S119** Expert Reviewer Panel Member, 28th IEEE/ACM International Conference on Automated Software Engineering (ASE 2013), Palo Alto, CA, November 2013
- S120** Program Committee Member, 9th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2013), Saint Petersburg, Russia, August 2013
- S121** Symposium Committee Member, Doctoral Symposium, 9th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2013), Saint Petersburg, Russia, August 2013
- S122** Program Committee Member, 7th European Conference on Software Architecture (ECSA 2013), Montpellier, France, July 2013
- S123** Program Committee Member, 7th ACM International Conference on Distributed Event-Based Systems (DEBS 2013), Arlington, TX, June-July 2013
- S124** Program Committee Member, 16th International ACM SIGSOFT Symposium on Component-Based Software Engineering, Vancouver, Canada, June 2013
- S125** Program Committee Member, 4th International ACM SIGSOFT Symposium on Architecting Critical Systems (ISARCS), Vancouver, Canada, June 2013
- S126** Program Committee Member, 33rd International Conference on Software Engineering - New Ideas and Emerging Results Track, San Francisco, CA, May 2013
- S127** Program Committee Member, 2nd International Workshop on the Twin Peaks of Requirements and Architecture, San Francisco, CA, May 2013
- S128** Program Committee Member, 1st International Workshop on the Twin Peaks of Requirements and Architecture, Chicago, IL, September 2012
- S129** Program Committee Member, 6th ACM International Conference on Distributed Event-Based Systems, Berlin, Germany, July 2012
- S130** Program Committee Member, 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2012), Zurich, Switzerland, June 2012
- S131** Program Committee Member, 3rd International Workshop on Software Engineering for Sensor Network Applications, Zurich, Switzerland, June 2012

- S132** Mentoring Committee Member, 34th International Conference on Software Engineering (ICSE 2012), Zurich, Switzerland, June 2012
- S133** Program Committee Member, 22nd IEEE International Symposium on Software Reliability Engineering, Hiroshima, Japan, November-December 2011
- S134** Program Committee Member, 9th IEEE International Conference on Software Engineering and Formal Methods, Montevideo, Uruguay, November 2011
- S135** Expert Reviewer Panel Member, 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, November 2011
- S136** Workshops Committee Member, The joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2011), Szeged, Hungary, September 2011
- S137** Program Committee Member, 5th Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS), Sao Paulo, Brazil, September 2011
- S138** Program Committee Member, 9th IEEE/IFIP Working Conference on Software Architecture (WICSA), Boulder, CO, June 2011
- S139** Program Committee Member, 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Honolulu, HI, May 2011
- S140** Program Committee Member, IEEE Globecom 2010 Workshop on Pervasive Group Communications (PerGroup), Miami, FL, December 2010
- S141** Program Committee Member, 18th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 2010), Santa Fe, NM, November 2010
- S142** Program Committee Member, International Symposium on Architecting Critical Systems (ISARCS 2010), Prague, Czech Republic, June 2010
- S143** Program Committee Member, 32nd International Conference on Software Engineering, Cape Town, South Africa, May 2010
- S144** Program Committee Member, 5th International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Cape Town, South Africa, May 2010
- S145** Program Committee Member, Workshop on Flexible Modeling Tools (FlexiTools), Cape Town, South Africa, May 2010
- S146** Program Committee Member, Workshop on Quantitative Stochastic Models in the Verification and Design of Software Systems (QUOVADIS), Cape Town, South Africa, May 2010
- S147** Expert Panel Member, 24th IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), Auckland, New Zealand, November 2009
- S148** Program Committee Member, 8th Working IEEE/IFIP Conference on Software Architecture (WICSA) and 3rd European Conference on Software Architecture (ECSA), Cambridge, UK, September 2009
- S149** Program Committee Member, 12th European Software Engineering Conference and 17th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2009), Amsterdam, the Netherlands, August 2009

- S150** Program Committee Member, 8th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA'09), Rhodes, Grece, July 2009
- S151** Program Committee Member, DSN 2009 Workshop on Architecting Dependable Systems (WADS 2007), Lisbon, Portugal, June 2009
- S152** Program Committee Member, 5th International Conference on the Quality of Software Architecture (QoSA 2009), East Stroudsburg, PA, June 2009
- S153** Program Committee Member, 6th International Conference on Autonomic Computing (ICAC), Barcelona, Spain, June 2009
- S154** Program Committee Member, 4th International Workshop on UML and AADL, Potsdam, Germany, June 2009
- S155** Program Committee Member, 31st International Conference on Software Engineering, Vancouver, Canada, May 2009
- S156** Program Committee Member, 4th International Conference on the Quality of Software Architectures (QoSA 2008), Karlsruhe, Germany, October 2008
- S157** Program Committee Member, 11th International Symposium on Component-Based Software Engineering (CBSE 2008), Karlsruhe, Germany, October 2008
- S158** Program Committee Member, 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), L'Aquila, Italy, September 2008
- S159** Program Committee Member, 2nd Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS 2008), Porto Alegre, Brazil, August 2008
- S160** Program Committee Member, 7th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA'08), Reykjavik, Iceland, July 2008
- S161** Program Committee Member, DSN 2008 Workshop on Architecting Dependable Systems (WADS 2007), Anchorage, AK, June 2008
- S162** Program Committee Member, 30th International Conference on Software Engineering, Leipzig, Germany, May 2008
- S163** Symposium Committee Member, Doctoral Symposium, 30th International Conference on Software Engineering, Leipzig, Germany, May 2008
- S164** Program Committee Member, 3rd International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2008), Leipzig, Germany, May 2008
- S165** Program Committee Member, International Workshop on Software Architectures and Mobility (SAM 2008), Leipzig, Germany, May 2008
- S166** Program Committee Member, 22nd IEEE/ACM International Conference on Automated Software Engineering, Atlanta, GA, November 2007
- S167** Program Committee Member, International Workshop on the Engineering of Software Services for Pervasive Environments (ESSPE), Dubrovnik, Croatia, September 2007
- S168** Program Committee Member, 10th International Symposium on Component-Based Software Engineering (CBSE 2007), Boston, MA, July 2007

- S169** Program Committee Member, 3rd International Conference on the Quality of Software Architectures (QoSA 2007), Boston, MA, July 2007
- S170** Program Committee Member, First IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007), Boston, MA, July 2007
- S171** Program Committee Member, DSN 2007 Workshop on Architecting Dependable Systems (WADS 2007), Edinburgh, UK, June 2007
- S172** Program Committee Member, 2nd International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2007), Minneapolis, MN, May 2007
- S173** Program Committee Member, 2nd Workshop on Sharing and Reusing Architectural Knowledge - Architecture Rationale and Design Intent (SHARK/ADI), Minneapolis, MN, May 2007
- S174** Tutorials Committee Member, 29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, May 2007
- S175** Program Committee Member, Workshop on Tools, Operating Systems and Programming Models for Developing Reliable Systems (TOPMoDeLS), Long Beach, CA, March 2007
- S176** Program Committee Member, 6th International Workshop on Software and Performance (WOSP 2007), Buenos Aires, Argentina, February 2007
- S177** Program Committee Member, 6th Working IEEE/IFIP Conference on Software Architecture, Mumbai, India, January 2007
- S178** Program Committee Member, 14th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 14), Portland, OR, November 2006
- S179** Symposium Committee Member, Doctoral Symposium, 14th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 14), Portland, OR, November 2006
- S180** Expert Reviewer Panel Member, 21st IEEE/ACM International Conference on Automated Software Engineering (ASE 2006), Tokyo, Japan, September 2006
- S181** Doctoral Symposium Committee Member, 21st IEEE/ACM International Conference on Automated Software Engineering (ASE 2006), Tokyo, Japan, September 2006
- S182** Program Committee Member, 9th International Symposium on Component-Based Software Engineering (CBSE 2006), Västerås, Sweden, June/July 2006
- S183** Program Committee Member, ICSE 2006 Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Shanghai, China, May 2006
- S184** Tutorials Committee Member, 28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 2006
- S185** Program Committee Member, 20th IEEE International Conference on Automated Software Engineering (ASE 2005), Long Beach, CA, November 2005
- S186** Program Committee Member, 10th European Software Engineering Conference and 13th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2005), Lisbon, Portugal, September 2005

- S187** Program Committee Member, 9th International Software Product Line Conference (SPLC- EUROPE 2005), Rennes, France, September 2005
- S188** Program Committee Member, 5th International Workshop on Software Engineering and Middleware (SEM 2005), Lisbon, Portugal, 5-6 September 2005
- S189** Program Committee Member, Workshop on Architecture-Centric Evolution (ACE 2005), Glasgow, UK, July 2005
- S190** Program Committee Member, 4th International Workshop on Architecting Dependable Systems (WADS 2004), St. Louis, MO, May 2005
- S191** Program Committee Member, 8th International Symposium on Component-Based Software Engineering (CBSE8), St. Louis, MO, May 2005
- S192** Program Committee Member, Workshop on Software Composition (SC 2005), Edinburgh, UK, April 2005
- S193** Program Committee Member, Hawaii International Conference on System Sciences, Mini- Track on Strategic Software Engineering, Hawaii, January 2005
- S194** Program Committee Member, 2004 Workshop on Self-Managed Systems, Newport Beach, CA, November 2004
- S195** Program Co-Chair, Workshop on Software Architecture Description & UML, Lisbon, Portugal, October 2004
- S196** Program Committee Member, 19th IEEE International Conference on Automated Software Engineering (ASE 2004), Linz, Austria, September 2004
- S197** Program Committee Member, 4th IEEE/IFIP Working Conference on Software Architecture (WICSA-4), Oslo, Norway, June 2004
- S198** Program Committee Member, Workshop on Capturing Experience in Software Architecture Best Practices, Oslo, Norway, June 2004
- S199** Program Committee Member, Twin Workshops on Architecting Dependable Systems (WADS 2004), Edinburgh, UK, May 2004 and Florence, Italy, June 2004
- S200** Program Committee Member, 2nd International Working Conference on Component Deployment (CD 2004), Edinburgh, UK, May 2004
- S201** Program Committee Member, 7th International Symposium on Component-Based Software Engineering (CBSE7), Edinburgh, UK, May 2004
- S202** Program Committee Member, International Workshop on Incorporating COTS into Software Systems: Tools and Techniques (IWICSS), Redondo Beach, CA, February 2004
- S203** Program Committee Member, 10th Working Conference on Reverse Engineering (WCRE 2003), Victoria, BC, Canada, November 2003
- S204** Program Committee Member, 5th International Workshop on Product Family Engineering (PFE-5), Siena, Italy, November 2003
- S205** Program Committee Member, 2nd International Workshop on Architecting Dependable Systems (WADS 2003), Portland, OR, May 2003

- S206** Program Committee Member, Conference on Generative and Component-Based Software Engineering (GCSE), Pittsburgh, PA, October 2002
- S207** Program Committee Member, The Working IEEE/IFIP Conference on Software Architecture (WICSA-3), Montreal, Canada, August 2002
- S208** Program Committee Member, 2nd Software Product Line Conference (SPLC-2), San Diego, CA, August 2002
- S209** Program Committee Member, International Workshop on Mobile Teamwork, Vienna, Austria, July 2002
- S210** Program Committee Member, ECOOP 2002 Workshop on Concrete Communication Abstractions Of The Next 701 Distributed Object Systems, Malaga, Spain, June 2002
- S211** Program Committee Member, 24th International Conference on Software Engineering (ICSE 2002), Orlando, FL, May 2002
- S212** Symposium Committee Member, Doctoral Symposium, 24th International Conference on Software Engineering (ICSE 2002), Orlando, FL, May 2002
- S213** Program Committee Member, 1st International Workshop on Architecting Dependable Systems (WADS 2002), Orlando, FL, May 2002
- S214** Program Committee Member, 4th International Workshop on Product Family Engineering (PFE-4), Bilbao, Spain, October 2001
- S215** Program Committee Member, 8th European Software Engineering Conference and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2001), Vienna, Austria, September 2001
- S216** Program Committee Member, The Second Working IEEE/IFIP Conference on Software Architecture (WICSA-2), Amsterdam, The Netherlands, August 2001
- S217** Program Committee Member, 23rd International Conference on Software Engineering (ICSE 2001), Toronto, Canada, May 2001
- S218** Program Committee Member, Second International Workshop on Engineering Distributed Objects (EDO 2000), Davis, CA, November 2000
- S219** Program Committee Member, Eighth ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE8), San Diego, CA, November 2000
- S220** Program Committee Member, Fourth International Software Architecture Workshop (ISAW-4), Limerick, Ireland, June 2000
- S221** Program Committee Member, Third International Workshop on Software Architectures for Product Families (IW-SAPF-3), Las Palmas de Gran Canaria, Spain, March 2000

Referee and Reviewer Service

- S214** Journal of Empirical Software Engineering 2017
- S222** National Science Foundation, Review Panel 1/2000, 10/2003, 11/2013, 1/2016, 3/2018, 10/2021, 10/2024
- S223** ACM Transactions on Autonomous and Adaptive Systems 2015

- S224** ACM Transactions on Software Engineering and Methodology 1998, 1999, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013
- S225** IEEE Transactions on Software Engineering 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2025
- S226** Elsevier Information and Software Technology Journal 2007, 2008, 2011, 2014, 2015, 2016
- S227** Journal of Software Engineering for Robotics 2009
- S228** Architecting Dependable Systems VI, LNCS, Springer Verlag 2009
- S229** IEEE Software 2008
- S230** IEEE Transactions on Industrial Informatics 2008
- S231** Journal of Systems and Software 2005, 2006, 2007, 2008, 2011, 2015
- S232** University of Wisconsin-Milwaukee, Research Growth Initiative 2006
- S233** Journal of Automated Software Engineering 1999, 2000, 2005
- S234** 28th International Conference on Software Engineering, Outside Reviewer 2005
- S235** Architecting Dependable Systems III, LNCS, Springer Verlag 2005
- S236** Architecting Dependable Systems II, LNCS, Springer Verlag 2004
- S237** IEEE Transactions on Parallel and Distributed Systems 2003, 2004
- S238** National Science Foundation, Proposal Reviewer (PASI) 2/2011
- S239** IEEE Computer, Special Issue on Handheld Computing 2003
- S240** Architecting Dependable Systems, LNCS, Springer Verlag 2003
- S241** International Journal on Software Maintenance and Evolution 2002
- S242** Science Foundation Ireland 2002
- S243** IEE Proceedings - Software Engineering 1998
- S244** Conference on User Interface Software and Technology, Outside Reviewer 1996
- S245** 19th International Conference on Software Engineering, Outside Reviewer 1997

Other

- S246** Member, IFIP Working Group 2.4 – Software Implementation Technologies (2020 - present)
- S247** Participant, IFIP Working Group 2.10 - Software Architecture (2002 - 2007)

13 University Service

- S248** Chair, Thomas Lord Department of Computer Science, USC (7/22 – present)
- S249** Vice-Chair for Tenure Affairs, Department of Computer Science, USC (7/21 – 6/22)
- S250** Chair, Diversity, Equity, and Inclusion (DEI) Committee, Computer Science Department, USC (2/21 – 6/22)

- S251** Chair, Hiring Committee, Computer Science Department, USC (10/21 – 5/22)
- S252** Chair, Hiring Committee, Computer Science Department, USC (10/20 - 5/21)
- S253** Member, Ad-Hoc Promotion and Tenure Committee, Computer Science Department, USC (1/20 - 3/21)
- S254** Appointments, Promotions, and Tenure (APT) Committee, Viterbi School of Engineering, USC (8/20 - 7/21)
- S255** APT Executive Committee, Viterbi School of Engineering, USC (8/20 - 7/21)
- S256** Area Lead, Computer Systems Area, Computer Science Department, USC (8/19 - 8/20)
- S257** Member, Faculty Hiring Committee - Software Engineering, Computer Science Department, USC (12/19 - 5/20)
- S258** Member, Faculty Hiring Committee - Programming Languages, Computer Science Department, USC (12/19 - 5/20)
- S259** Chair, Ad-Hoc Promotion and Tenure Committee, Computer Science Department, USC (12/19 - 3/20)
- S260** Member, Research Ethics Committee, University of Southern California (2/19 - 5/20)
- S261** Committee on Academic Policies and Procedures, University of Southern California (8/17 - 8/21)
- S262** Chair, Ad-Hoc Promotion and Tenure Committee, Computer Science Department, USC (8/19 - 1/20)
- S263** Member, Hiring Committee, Computer Science Department, USC (10/18 - 5/19)
- S264** Member, Ad-Hoc Promotion and Tenure Committee, Computer Science Department, USC (12/18 - 5/19)
- S265** Chair, Teaching Faculty Merit Review Committee, Computer Science Department, USC (1/18 - 5/18)
- S266** Chair, Ad-Hoc Promotion and Tenure Committee, Computer Science Department, USC (8/17 - 12/17)
- S267** Member, Joint-Appointment Review Committee, Computer Science Department, USC (9/17 - 11/17)
- S268** Member, Ad-Hoc Promotion and Tenure Committee, Computer Science Department, USC (3/17 - 5/17)
- S269** Chair, Faculty Hiring Committee - Programming Languages, Computer Science Department, USC (12/16 - 5/17)
- S270** Member, Merit Review Committee, Computer Science Department, USC (2/17 - 3/17)
- S271** Chair, Merit Review Process Committee, Computer Science Department, USC (8/16 - 2/17)
- S272** Chair, Merit Review Committee, Computer Science Department, USC (2/16 - 3/16)
- S273** Chair, Research Faculty Promotion Committee Computer Science Department, USC (1/16 – present)

- S274** Chair, Research Faculty Promotion Committee, Computer Science Department, USC (11/15 - 11/16)
- S275** Chair, Faculty Hiring Committee - Software Engineering, Computer Science Department, USC (12/15 - 5/16)
- S276** Chair, Faculty Hiring Committee - Programming Languages, Computer Science Department, USC (12/15 - 5/16)
- S277** Chair, Ad-Hoc Promotion and Tenure Committee, Computer Science Department USC (3/15 - 5/16)
- S278** Ad-Hoc Promotion Committee, Computer Science Department, USC (5/15 - 5/16)
- S279** Associate Chair for Ph.D. Affairs, Computer Science Department, USC (7/11 - 6/15)
- S280** Chair, PhD Program Reboot Committee, Computer Science Department, USC (8/13 - 6/15)
- S281** Chair, Ph.D. Admissions Committee, Computer Science Department, USC (7/11 - 6/15)
- S282** Ph.D. Council, Viterbi School of Engineering, USC (1/12 - 6/15)
- S283** Research Committee, Viterbi School of Engineering, USC (7/12 - 8/13)
- S284** Transformative Faculty Hiring Committee, Computer Science Department, USC (6/12 - 5/13)
- S285** Director, Center for Systems & Software Engineering, Viterbi School of Engineering USC (1/09 - 2/13)
- S286** Joint Faculty Appointment Committee, Computer Science Department, USC (11/12 - 12/12)
- S287** Department Chair Search Committee, Computer Science Department, USC (9/11 - 5/12)
- S288** Chair Performance Evaluation Committee, Computer Science Department, USC (3/12 - 5/12)
- S289** Department Ranking Committee, Computer Science Department (2/12 - 5/12)
- S290** Engineering Faculty Council (EFC), Viterbi School of Engineering, USC (1/02 - 5/05, 8/06 - 5/10)
- S291** New Building Faculty Advisory Committee, Viterbi School of Engineering, USC (1/10 - 3/14)
- S292** EFC Academic Senate Alternate, USC (8/08 - 8/09)
- S293** Appointments, Promotions, and Tenure (APT) Committee, Viterbi School of Engineering, USC (8/08 - 5/10)
- S294** Ph.D. Admissions Committee, Computer Science Department, USC (8/04 - 5/05, 8/07 - 8/15)
- S295** Ph.D. Fellowships Committee, Computer Science Department, USC (8/07 - 8/15)
- S296** Group Leader, Software Systems and Engineering, Computer Science Department, USC (8/08 - 8/10)
- S297** Chair, Faculty Hiring Committee, Computer Science Department, USC (8/08 - 5/09)

- S298** Department Chair's Advisory Committee, Computer Science Department, USC (8/07 - 7/08)
- S299** Merit Review Committee, Computer Science Department, USC (2000, 2004, 2008, 2010)
- S300** Provost/Academic Senate Committee on Information Services, USC (8/06 - 5/08)
- S301** Dean/EFC Standing Committee on Best Practices, USC (1/08 - 8/08)
- S302** Dean/EFC Standing Committee on Faculty Recruitment and Retention, Viterbi School of Engineering, USC (8/07 - 8/08)
- S303** EFC Research Committee, Viterbi School of Engineering, USC (1/07 - 8/07, 1/10 - 5/10)
- S304** Chair, Interaction Group, Computer Science Department, USC (8/06 - 8/07)
- S305** Chair, EFC Merit Review Committee, Viterbi School of Engineering, USC (8/06 - 8/08)
- S306** Qualifying Exam Timeline Committee, Computer Science Department, USC (4/06 - 5/07)
- S307** M.S. Program Review Committee, Computer Science Department, USC (12/03 - 5/05)
- S308** Research Faculty Hiring Committee, Computer Science Department, USC (2002, 2003, 2006, 2007, 2008)
- S309** Dean's Evaluation Committee, School of Engineering, USC (11/04 - 1/05)
- S310** M.S. Program Review Committee, School of Engineering, USC (9/03 - 12/03)
- S311** Faculty Hiring Committee, Computer Science Department, USC (2001, 2004)
- S312** Interview Panel, Trustee Scholarship, School of Engineering, USC (3/02, 3/04)
- S313** EFC Committee on Research Faculty Rights and Responsibilities, School of Engineering, USC (11/02 - 5/03)
- S314** Chair, Computer Systems Group, Computer Science Department, USC (12/99 - 5/02)
- S315** College of Engineering Commencement Marshal, USC (2000, 2002)
- S316** Industrial Liaison Committee, Computer Science Department, USC (2000)

14 Professional Associations

- Association for Computing Machinery (ACM)
- ACM Special Interest Group on Software Engineering (SIGSOFT)
- Institute of Electrical and Electronics Engineers (IEEE)
- IEEE Computer Society (CS)

15 Consulting

- 2010 - present Expert witness in multiple legal matters involving software technology
- 2/11 - present Aerospace Corporation, El Segundo, CA
- 6/00 - 6/03 Jet Propulsion Laboratory (JPL), Pasadena, CA