

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF WASHINGTON
AT SEATTLE

REC SOFTWARE USA, INC.,

Plaintiff,

v.

BAMBOO SOLUTIONS
CORPORATION, et al.,

Defendants.

CASE NO. C11-0554JLR

ORDER ON
CLAIM CONSTRUCTION

I. INTRODUCTION

This is an order on claim construction in a patent infringement action involving a single patent related to what the patent names a “code server,” which maintains and provides information for linking computer code modules that together constitute a computer program. Plaintiff REC Software USA, Inc. (“REC”) sued Defendants Bamboo Solutions Corporation, Microsoft Corporation, SAP America Inc., and SAP AG (collectively, “Defendants”) for infringement of claims 1 and 8 of United States Patent

1 No. 5,854,936 (“the ’936 Patent”) (the “Patent-in-Suit”).¹ The court has considered the
2 parties’ briefing and supporting materials and has heard both oral argument from the
3 parties and expert testimony at a *Markman* hearing, held on January 13, 2011. This order
4 memorializes the court’s construction of the disputed language in the Patent-in-Suit.

5 **II. BACKGROUND**

6 **A. Introduction**

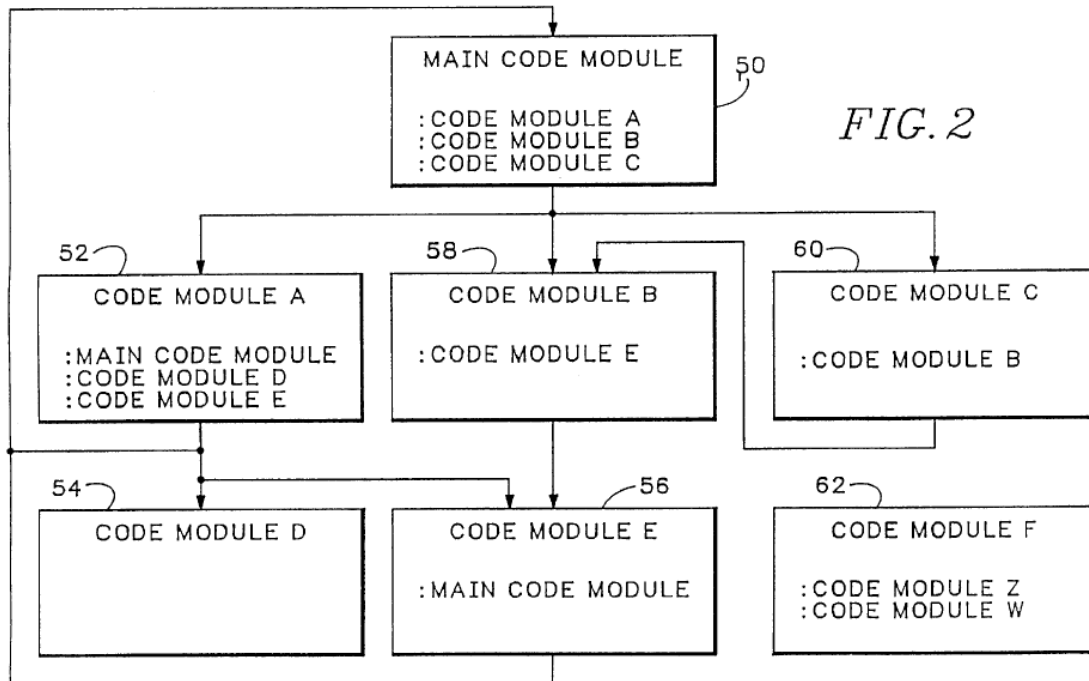
7 Stephen F.B. Pickett is the sole inventor of the Patent-in-Suit, which was assigned
8 to REC. (*See* U.S. Patent No. 5,854,936 (‘936 Patent).) REC contends that Microsoft
9 Corporation’s (“Microsoft”) operating systems that are supported by, or support or
10 utilize, the .NET Framework—namely, Windows 2000, Windows Server 2003, Windows
11 Server 2008, Windows Vista, Windows XP, and Windows 7—infringe the above-stated
12 claims of the Patent-in-Suit. (Joint Claim Construction Statement (Dkt. # 112) at 1-2.)
13 Additionally, REC contends that SAP America, Inc. and SAP AG’s (collectively, “SAP”)
14 “NetWeaver” products infringe the above-stated claims of the Patent-in-Suit. (*Id.* at 2.)

15 **B. Factual Background**

16 The Patent-in-Suit disclose systems and methods related to the behind-the-scenes
17 work done by a computer’s operating system to prepare complex computer programs for
18 execution by a “user” computer. (‘936 Patent at 1:6-10; Defendants Opening Br. (Dkt. #
19 119) at 5.) The central component of the invention is a “code server,” which operates in
20 a multi-module computer operating system to efficiently identify and facilitate the

21
22 ¹ In its complaint, REC asserted claims 1-6, 8-10, 13, 15, 17, 18, and 22 of the ’936
Patent, but by stipulation now only asserts claims 1 and 8. (Stip. (Dkt. # 150).)

1 provision of modules of computer code that constitute a computer program (known by
 2 the Patent-in-Suit as the “second multi-module program”) to a user computer for
 3 execution of that computer program. ('936 Patent at 1:6-10, 2:12-14.) A multi-module
 4 operating system includes computer programs consisting of modules of computer code
 5 that often contain embedded references (also known as linkage information or associative
 6 information) to other modules of computer code. (*Id.* at 1:11-13.) Figure 2 below
 7 provides a schematic diagram of a coded computer program divided into modules, which
 8 in turn reference other modules. (*Id.* at 2:54-57.) Each block of Figure 2 constitutes a
 9 module of computer code, which list other modules referenced by that module. The
 10 arrows in Figure 2 show the connection between a module and the modules that it
 11 references. (*Id.*)



1 Prior to, or at the time of, execution of the computer program by the “user”
2 computer, the embedded references within the modules of the computer program must be
3 linked together (or resolved) by the operating system. (*Id.* at 1:13-15.) Prior to the
4 present invention, there were two traditional ways of linking the modules of computer
5 code together—static linking and dynamic linking. (REC Tutorial (Dkt. # 133-1) at 21.)
6 In static linking, a program called a “linker,” operating on the “user” computer, pulls all
7 the various modules together into one large module with fully linked internal references.
8 (*Id.* at 21-24.) This large module is then placed by the “user” computer on a hard drive.
9 When the “user” computer was ready to run the large module (the fully-linked
10 application), the “user” computer would load the entire module into Random Access
11 Memory (“RAM”) and operate the instructions from there.² (*Id.* at 25.)

12 In dynamic linking, all of the modules of a multi-module program are stored on
13 the hard drive, but unlike static linking, they are not linked together. (*Id.* at 28.) At the
14 time the “user” computer executes the multi-module program, the operating system
15 utilizes a linker program to read through the code modules to identify the other modules
16 that are referenced by each module. (*Id.* at 29.) For instance, in Figure 2 above, the
17 linker program would begin at the “Main Code Module,” which references Modules A,
18 B, and C. (*Id.* at 29-35.) For each referenced module, the linker program constructs in
19 RAM a “dynamic link table” including the identification of Modules A, B, and C as
20 modules referenced by the “Main Code Module,” as well as the RAM address where each

21
22 ² RAM is a faster memory type than memory that makes up the hard drive of a computer.
(REC Tutorial at 5.)

1 Module can be found. (*Id.* at 31.) The information of references between modules (i.e.
2 linkage information) found in the dynamic link table constitutes what the Patent-in-Suit
3 refers to as an “association.” (’936 Patent at 2:14-22.) The dynamic linker may continue
4 to construct an association for the modules referenced by Modules A, B, and C, and so on
5 for the remaining modules of the multi-module program. (REC Tutorial at 31.)

6 In practice, the dynamic link table can be used by another program that utilizes
7 some of the same modules as the first program, such that a specific module is only loaded
8 into RAM once, saving time and memory space. (Microsoft Tutorial (Dkt. # 132-1) at
9 28.) This was an improvement over the static linking process. (*Id.* at 28.) Still, the
10 dynamic linking process was slow in searching for the location of each module
11 referenced by the Main Code Module or other modules containing embedded references.
12 (REC Tutorial at 38.) The present invention purports to improve the efficiency of
13 running a multi-module program when compared to dynamic linking and static linking.
14 (*Id.* at 42.)

15 The present invention adds a “code server” to the described multi-module
16 operating system environment. (’936 Patent at 2:12-14.) The code server includes a
17 “module information table” containing information that links or associates discrete
18 modules of a multi-module program. (*Id.* at 2:14-22; 3:49-50.) Thus, a code server
19 functions to provide linking information for modules of a multi-module program to a
20 “user” computer that desires to run the program. (*Id.* at 28-32.) By storing module
21 linking information in a code server, the need to search through a multi-module program
22

1 and then find the location of referenced modules in memory is greatly reduced. Figure 3
 2 below depicts the interface between the code server and the “user” computer.

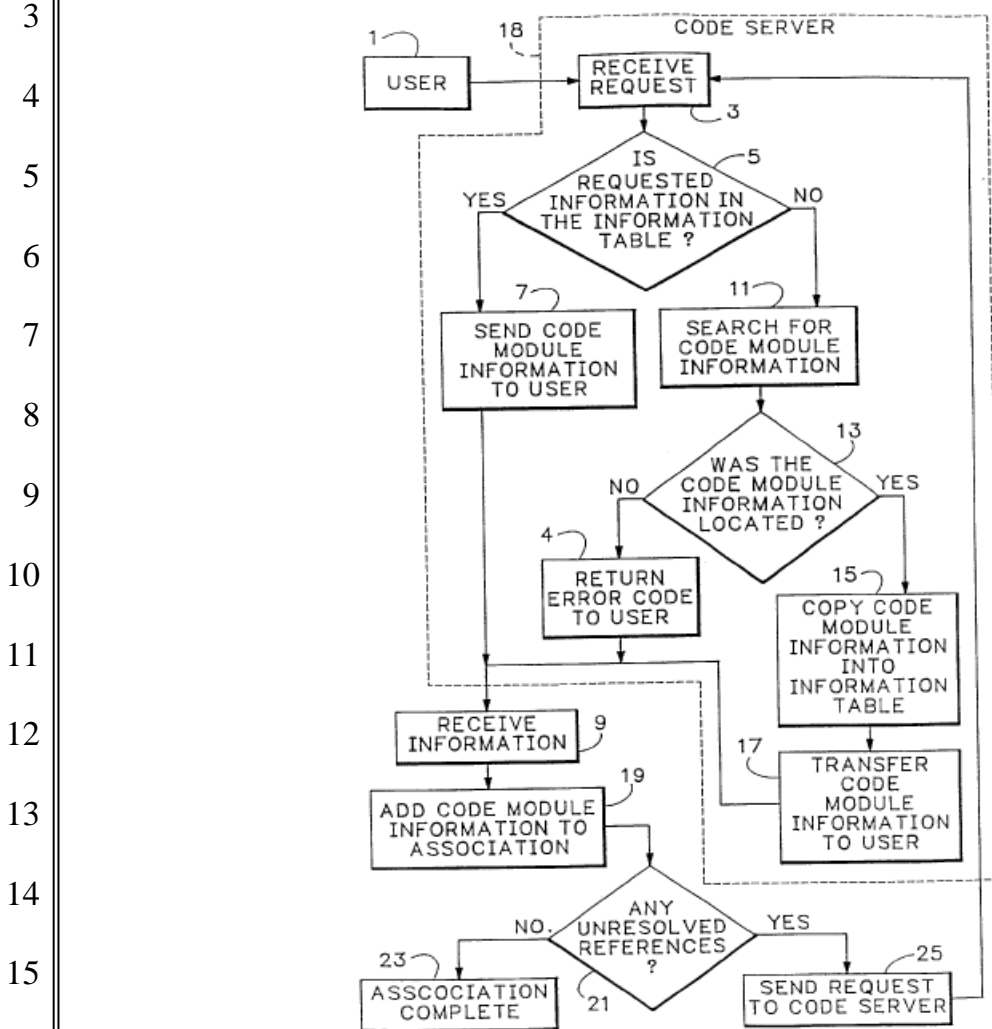


FIG. 3

17 In this matter, REC has asserted claim 1 and 8 of the ‘936 Patent. Claim 1 is an
 18 independent claim and claim 8 is a dependent claim. Claim 1, a method claim, is
 19 provided below:

- 20 1. A method of providing information to a **first program that is executing on a**
 21 **computer** for forming an association for a **second multi-module program** which
 22 includes an embedded reference to a discrete module, said method comprising the
 steps of:

- 1 (a) receiving in a **code server** from said first program, at a point prior to execution-
2 time of said multi-module program being associated, a request associated with said
3 discrete module;
4 (b) **searching a module information table for module information in response to**
5 **said request associated with said discrete module;**
6 (c) in response to finding said module information in said module information table,
7 reading said module information;
8 (d) providing said module information in a response to said first program; and
9 (e) **forming an association of said multi-module program by said first program.**

10 ('936 Patent at Claim 1 (bolding added).) The bolded portions of the quoted claim
11 language represent the claim terms in dispute.

12 III. ANALYSIS

13 In *Markman v. Westview Instruments, Inc.*, the Supreme Court placed sole
14 responsibility for construing patent claims on the court. 517 U.S. 370, 372 (1996). The
15 Federal Circuit later established that the court construes claims purely as a matter of law.
16 *Cybor Corp. v. FAS Tech., Inc.*, 138 F.3d 1448, 1456 (Fed. Cir. 1998) (applying de novo
17 review to all claim construction issues, even “allegedly fact-based questions”).
18 Executing the *Markman* mandate requires a court to interpret claims after giving the
19 appropriate level of consideration to various sources of evidence.

20 Intrinsic evidence, which includes the patent and its prosecution history, is the
21 primary source from which to derive a claim’s meaning. *Phillips v. AWH Corp.*, 415
22 F.3d 1303, 1314 (Fed. Cir. 2005) (en banc). A patent is composed of three parts: (1) a
“written description,” which consists of an often lengthy exposition of the background of

1 the invention, at least one embodiment of the invention, and other written material that
2 assists in understanding how to practice the invention; (2) (in most cases) a set of
3 drawings that illustrates portions of the written description; and (3) the claims, which
4 delimit the scope of the invention. *General Foods Corp. v. Studiengesellschaft Kohle*
5 *mbH*, 972 F.2d 1272, 1274 (Fed. Cir. 1992). Together, these three components make up
6 the patent’s “specification.”³ *Atmel Corp. v. Info. Storage Devices, Inc.*, 198 F.3d 1374,
7 1384 (Fed. Cir. 1999); 35 U.S.C. § 112.

8 The prosecution history exists independently of the patent. It consists of the
9 inventor’s application to the United States Patent and Trademark Office (“PTO”) and all
10 correspondence between the PTO and the inventor documenting the invention’s progress
11 from patent application to issued patent. *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d
12 1576, 1582 (Fed. Cir. 1996).

13 In its review of intrinsic evidence, the court begins with the language of both the
14 asserted claim and other claims in the patent. *Phillips*, 415 F.3d at 1314; *Biagro Western*
15 *Sales, Inc. v. Grow More, Inc.*, 423 F.3d 1296, 1302 (Fed. Cir. 2005) (“It is elementary
16 that claim construction begins with, and remains focused on, the language of the
17 claims.”). The court’s task is to determine the “ordinary and customary meaning” of the
18 terms of a claim through the eyes of a person of ordinary skill in the art on the filing date
19 of the patent. *Phillips*, 415 F.3d at 1313 (quoting *Vitronics*, 90 F.3d at 1582).

20
21 ³ Although 35 U.S.C. § 112 includes the claims as part of the specification, many courts
22 and practitioners use the term “specification” to refer to all portions of a patent except the claims.
In most instances, the context will reveal what portion of the specification is at issue.

1 Sometimes, the ordinary meaning is “readily apparent even to lay judges,” in which case
2 claim construction “involves little more than the application of the widely accepted
3 meaning of commonly understood words.” *Id.* at 1314.

4 The court must read claim language, however, in light of the remainder of the
5 specification. *Id.* at 1316 (“[T]he specification necessarily informs the proper
6 construction of the claims.”). In cases where the ordinary meaning of a claim term seems
7 apparent from its use in the claim, the court must consult the specification either to
8 confirm that meaning or to establish that the inventor intended a different meaning.
9 *Superguide Corp. v. DirecTV Enters., Inc.*, 358 F.3d 870, 875 (Fed. Cir. 2004). If the
10 ordinary meaning is not apparent from its use in the claim, the court looks to the
11 specification to provide meaning. *Johnson Worldwide Assocs., Inc. v. Zebco Corp.*, 175
12 F.3d 985, 990 (Fed. Cir. 1999). The specification acts as a “concordance” for claim
13 terms, and is thus the best source beyond claim language for understanding claim terms.
14 *Phillips*, 415 F.3d at 1315. The inventor is free to use the specification to define claim
15 terms as she wishes, and the court must defer to an inventor’s definition, even if it is
16 merely implicit in the specification. *Id.* at 1316 (“[T]he inventor’s lexicography
17 governs.”), 1320–21 (noting that a court cannot ignore implicit definitions). The court
18 should “rely heavily” on the specification in interpreting claim terms. *Id.* at 1317.

19 When the court relies on the specification, however, it must walk a tightrope
20 between properly construing the claims in light of the written description and the
21 “cardinal sin” of improperly importing limitations from the written description into the
22 claims. *SciMed Life Sys., Inc. v. Advanced Cardiovascular Sys., Inc.*, 242 F.3d 1337,

1 1340 (Fed. Cir. 2001); *Phillips*, 415 F.3d at 1323 (citing *Comark Commc'ns, Inc. v.*
2 *Harris Corp.*, 156 F.3d 1182, 1186-87 (Fed. Cir. 1998)). A patentee often provides
3 examples or “embodiments” of his or her invention in the written description, but courts
4 may not limit the invention to an embodiment absent clear evidence that a patentee
5 “intends for the claims and the embodiments . . . to be strictly coextensive.” *Phillips*, 415
6 F.3d at 1323.

7 Although a patent’s prosecution history is also intrinsic evidence, it is “less useful
8 for claim construction purposes,” because it usually “lacks the clarity of the
9 specification.” *Id.* at 1317. The prosecution history is useful, however, in determining if
10 an inventor has disavowed certain interpretations of his or her claim language. *Id.*

11 Finally, the court can consider extrinsic evidence, “including expert and inventor
12 testimony, dictionaries, and learned treatises.” *Id.* (citing *Markman v. Westview*
13 *Instruments, Inc.*, 52 F.3d 967, 980 (Fed. Cir. 1995)). Extrinsic evidence is usually “less
14 reliable than the patent and its prosecution history” as a source for claim interpretation.
15 *Id.* at 1318. The court thus need not admit extrinsic evidence, but may do so in its
16 discretion if intrinsic evidence does not disclose the meaning of a claim term. *Id.* at
17 1319; *Vitronics*, 90 F.3d at 1583 (“[W]here the public record unambiguously describes
18 the scope of the patented invention, reliance on any extrinsic evidence is improper.”).
19 With this general framework in mind, the court turns to the claim terms in dispute.

1 **A. “First program that is executing on a computer”/ “second multi-module**
2 **program”**

3 The terms-in-dispute are found in asserted claim 1 of the '936 Patent. The parties
4 offer the following proposed constructions:

5 **REC’s Proposed Constructions: “First program that is executing on a**
6 **computer”** means “a set of computer instructions running on a computer that enables the
7 computer to perform a specific operation or operations” **and “second multi-module**
8 **program”** means “a set of computer instructions that comprises two or more modules
9 and enables the computer to perform a specific operation or operations.” (Updated Joint
10 Claim Chart (Dkt. # 134-1) at 1.⁴)

11 **Defendants’ Proposed Constructions:** The terms-in-dispute should be given
12 their ordinary meaning.⁵ (Updated Claim Chart at 1.)

13 The central dispute between the parties is whether the word “program” must be
14 defined. Although the parties agree that the specification is silent with respect the word
15 “program,” they disagree as to how that silence is to be interpreted. Defendants contend
16 that the word has a well-understood meaning in the computer science field and therefore
17 needs no construction. (Microsoft Opening Br. at 9-10.) REC responds that without a

18
19 ⁴ On January 11, 2012, after completion of *Markman* briefing, the parties filed a Joint
20 Claim Chart containing updated proposed constructions for each disputed term. (Dkt. # 134-1.)
The court will rely on the proposed constructions contained in the updated Joint Claim Chart in
this *Markman* order.

21 ⁵ During the *Markman* hearing, for the first time, Microsoft provided the court with the
22 following proposed construction of the word “program”: “The complete set of computer
instructions to perform a process.” (Transcript (Dkt. # 145) at 98-99.)

1 construction, Microsoft is free to enlarge the scope of the word “program,” through
2 argument to the finder of fact, to encompass entire software products, which according to
3 REC constitute multiple “programs.” (REC Opening Br. at 17.) REC seeks to define the
4 word “program” through dictionary definitions. (*Id.*)

5 The court will construe the terms-in-dispute. Here, the parties have a genuine
6 dispute as to the scope of the word “program.” *O2 Micro Intern. Ltd. v. Beyond*
7 *Innovation Technology Co., Ltd.*, 521 F.3d 1351, 1361 (Fed. Cir. 2008) (“In deciding that
8 ‘only if’ needs no construction because the term has a ‘well-understood definition,’ the
9 district court failed to resolve the parties’ dispute because the parties disputed not the
10 *meaning* of the words themselves, but the *scope* that should be encompassed by this
11 claim language.”) (internal quotations omitted).

12 Additionally, although the word “program” may readily be understood by one of
13 skill in the art, to a lay person finder-of-fact, the word has numerous meanings, e.g., a
14 computer program versus a baseball game program versus a television program. Thus,
15 construction of the term is appropriate. *Id.* (“A determination that a claim term ‘needs no
16 construction’ or has the ‘plain and ordinary meaning’ may be inadequate when a term has
17 more than one ‘ordinary’ meaning or when reliance on a term’s ‘ordinary’ meaning does
18 not resolve the parties’ dispute.”).

19 As is the case here, where the specification (and other intrinsic evidence) is silent
20 as to the word “program,” the court may refer to dictionary definitions for guidance. *L.B.*
21 *Plastics, Inc. v. Amerimax Home Prods.*, 499 F.3d 1303, 1308 (Fed. Cir. 2007) (“Since
22 the intrinsic record provides no further guidance to the meaning of the terms ‘weld,’

1 ‘fuse’ or ‘ultrasonic or heat welding,’ the district court properly turned to extrinsic
2 evidence in this case and consulted dictionaries.” REC has provided the court with
3 numerous technical dictionary definitions for the word “program.” (REC Opening Br. at
4 17 (citing *The Illustrated Dictionary of Microcomputers* (3rd ed. 1990) (“A set of
5 instructions . . . directing the computer in performing a desired operation,”); *Wiley Electr.
6 and Electr. Engineering Dictionary* (2004) (“A set of instructions which . . . cause a
7 computer to perform specific operations.”).) The court finds that REC’s proposed
8 constructions for the terms-in-dispute closely align with the technical dictionary
9 definitions apt to define the word “program” in the context of the Patent-in-Suit.

10 Accordingly, the court adopts REC’s proposed constructions. **“First program**
11 **that is executing on a computer”** means “a set of computer instructions running on a
12 computer that enables the computer to perform a specific operation or operations” and
13 **“second multi-module program” means** “a set of computer instructions that comprises
14 two or more modules and enables the computer to perform a specific operation or
15 operations.

16 **B. “code server”**

17 The term “code server” is found in asserted claim 1 of the ’936 Patent and also in
18 numerous other claims of the Patent-in-Suit. As stated, the “code server” constitutes the
19 crux of the patented invention. The parties propose the following constructions for the
20 term-in-dispute:

21 **REC’s Proposed Construction:** “computer software that (i) is an identifiable set
22 of computer instructions other than the first program, and (ii) is capable of maintaining,

1 and providing to the first program upon request, module information for one or more
2 modules of a multi-module program.” (Updated Joint Claim Chart at 1-2.)

3 **Defendants’ Proposed Construction:** “a disjoined task (separate process from
4 the first program) that maintains and provides to the first program upon request module
5 information for one or more modules of a multi-module program.” (Updated Claim
6 Chart at 1.)

7 While the parties agree that the function of the code server is to maintain and
8 provide to the first program module information for one or more modules of a multi-
9 module program, the parties’ competing constructions raise two distinct disputes: (1)
10 whether the code server is computer software that is capable of performing certain tasks
11 or process (REC’s position) or is itself a “task” or “process” (Defendants’ position), and
12 (2) the language to use in describing the distinction between the code server and the first
13 program. These issues are discussed in turn below.

14 First, the court agrees with REC that the claim language and the specification
15 make clear that the code server is a thing, albeit an abstract thing, capable of performing a
16 task or tasks. The code server is found in the claim language of claim 1, a method claim:
17 “receiving in a code server from said first program, . . . a request associated with said
18 discrete module.” The plain claim language indicates that the code server is a thing
19 which receives a request from the first program as opposed to a task or process, as
20 Defendants contend. Consistent with the claim language, the specification describes the
21 code server as a thing, made up of various parts, which can perform a function. For
22 example, the specification states: “The code server, which provides information to users

1 of the data processing system, includes an information storage table containing linkage
2 information needed to form an association between discrete modules of code” (’936
3 Patent at 2:13-16.)

4 Defendants direct the court to the following passage of the specification, which
5 Defendants assert supports their construction:

6 The code server is embodied in a transaction oriented protocol for
7 communication between the user and the code server which allows the
8 users and the code server to be on two remote computer systems or to be
configured as separate disjointed tasks in a multi-processing system on a
single computer.

9 (Microsoft Opening Br. at 10 (citing ’936 Patent at 3:21-26).) As REC correctly points
10 out, this passage from the specification describes a possible configuration of the code
11 server and the user computer(s), but does not describe the code server itself. Thus,
12 Defendants’ reliance on this passage is misplaced. The court concludes that the code
13 server is more aptly described as a thing as opposed to a process or task.

14 With respect to the second dispute, although the parties agree that the code server
15 is distinct from the first program, REC and Defendants disagree as to the language for
16 describing this distinction. REC seeks to describe the distinction as “an identifiable set of
17 instructions other than the first program,” whereas Defendants describe the distinction as
18 a “separate process from the first program.” (REC Opening Br. at 19; Microsoft Opening
19 Br. at 11.) Essentially, the dispute turns on whether the word “other” or the word
20 “separate” should be used to describe the distinction.

21 REC contends that if the court accepts Defendants’ proposed construction,
22 Defendants will be free to argue that for the code server to be separate from the first

1 program either (1) the code server and first program may not interact, or (2) the code
2 server and first program must not reside on the same memory hardware. (REC Opening
3 Br. at 20.) REC contends that both of these arguments, as an extension of Defendants'
4 proposed construction, are incorrect. (*Id.*) The court agrees with REC on both accounts.
5 The claim language and the specification clearly indicate that the first program and the
6 code server will interact. ('936 Patent at Claim 1 ("receiving in a code server from said
7 first program").) Additionally, the code server and the first program may exist on the
8 same hardware. ('936 Patent at 3:21-26 ("the user and the code server [may] be on two
9 remote computer systems or . . . on a single computer.")) This is not to say that the code
10 server and the first program are the same. Indeed, they are certainly distinct from one
11 another. As Defendants explain, Figures 1 and 3 of the specification explicitly depict the
12 code server as distinct from the requesting (or user) program. (Microsoft Opening Br. at
13 11.)

14 Returning to the parties' dispute, the court is concerned that Defendants' choice of
15 "separate" affords Defendants the opportunity to improperly argue, as it does in its brief,
16 that the code server and first program are completely isolated from one another. (*See id.*
17 at 11.) On the other hand, REC's use of the word "other" fails to capture the requisite
18 distinction required by the specification of the '936 Patent. Accordingly, the court
19 believes that the word "different" properly captures the relationship between the code
20 server and the first program.

21 As a final matter, Defendants argue that REC's inclusion of the phrase "an
22 identifiable set of computer instructions" in its definition of code server lacks supporting

1 intrinsic evidence. The court has fully examined the specification of the '936 Patent and
2 finds that the specification fails to provide a clear definition of code server, instead
3 consistently defining the term by its functionality and purpose. (*See generally* '936
4 Patent.) In such a situation, the court will adopt a “construction that stays true to the
5 claim language and most naturally aligns with the patent’s description of the invention.”
6 *Phillips*, 415 F.3d at 1316. Here, the court finds that the code server is a piece of
7 software—a computer program. The specification makes clear that the code server may
8 be a separate piece of hardware, be built into the network, or be a part of the operating
9 system itself. ('936 Patent at 3:17-20.) Logically, functions performed through hardware
10 will be the result of the execution of a computer program. Moreover, the claim language
11 and specification recite interaction between the code server and the first program, which
12 seemingly could only occur if the code server was a computer program itself. Having
13 already defined the word “program” as a set of computer instructions, the court finds
14 REC’s inclusion of the phrase “an identifiable set of computer instructions” accurate and
15 helpful to the finder of fact. Indeed, the word “identifiable” assists the finder of fact in
16 distinguishing the code server from the first program.

17 In conclusion, based on the claim language and the specification, the court gives
18 the term “code server” the following construction: “an identifiable set of computer
19 instructions, different than the first program, which maintains and provides to the first
20 program upon request module information for one or more modules of a multi-module
21 program.”
22

1 **C. “searching a module information table for module information in response to**
2 **said request associated with said discrete module.”**

3 The term-in-dispute is found in asserted claim 1 of the '936. The parties propose
4 the following constructions:

5 **REC’s Proposed Construction:** “in response to a request that includes an
6 identification of the discrete module, the code server searches for module information in
7 the module information table pertaining to the discrete module.” (Updated Joint Claim
8 Chart at 2.)

9 **Defendants’ Proposed Construction:** “in response to a request that includes an
10 identification of the discrete module, the code server searches for module information in
11 the module information table pertaining to the discrete module and any referenced
12 modules.”⁶ (*Id.* at 2.)

13 Here, the parties’ dispute is clear: whether the term-in-dispute requires (1) search
14 for module information pertaining to the discrete module (REC’s position) or (2) search
15 for module information pertaining to the discrete module *and to any modules referenced*
16 *by the discrete module* (Defendants’ position). (*Id.* at 2 (emphasis added).) The parties’
17 proposed constructions mirror one another except that Defendants’ construction requires

18
19 ⁶ In its opening brief, Defendants proposed the following construction for the term-in-
20 dispute: “in response to a request that includes an identification of the discrete module, the code
21 server searches for any and all module information in the module information table pertaining to
22 the discrete module and any referenced modules.” (Microsoft Opening Br. at 14; REC Opening
Br. at 7.) REC disputed the “any and all” limitation contained in Defendants’ construction, and
in its responsive claim construction brief, Defendants removed the “any and all” language from
its original proposal. (Microsoft Resp. Br. (Dkt. # 124) at 13.) Therefore, the court no longer
must resolve this dispute.

1 a search not only for information pertaining to the discrete module, but additionally, for
2 any modules referenced by the discrete module. Both parties cite to intrinsic evidence in
3 support of their constructions. (See REC Opening Br. at 7-10; Microsoft Opening Br. at
4 14-16.) The court discusses the intrinsic evidence below, which it finds dispositive of the
5 proper construction, and for the reasons set forth below, the court agrees with REC's
6 proposed construction.

7 First, REC argues that an ordinary reading of the claim language supports its
8 proposed construction. (REC Opening Br. at 8.) Essentially, REC argues that "searching
9 a module information table for module information" per a request as to a discrete module
10 ordinarily means that a search of that discrete module occurs, and not a search of modules
11 referenced by the discrete module. (*Id.*) REC provides an example of a library which
12 maintains an information table on each of its books. According to REC's example, if the
13 someone was to search the library's information table as to the book *Lincoln the Lawyer*,
14 it would ordinarily mean that a search was conducted for information on *Lincoln the*
15 *Lawyer*, but not necessarily all books referenced by *Lincoln the Lawyer*. (*Id.*)

16 Defendants respond that claim construction should not be performed by isolating the
17 claim language from the specification, and when the term-in-dispute is read in light of the
18 intrinsic evidence, its construction that requires the search to look for "any referenced
19 module" is supported. (Microsoft Reply Br. at 13-14.) Here, the court is persuaded that
20 an ordinary reading of the claim language favors REC's construction, but agrees with
21 Defendants that the term-in-dispute must be read in light of the remaining intrinsic
22 evidence.

1 The court turns to the specification. Both parties principally cite to the same
2 passage: “a request by a user for a particular code module will immediately locate all of
3 the associate information pertaining to that module in the code module information
4 table.” (’936 Patent at 4:31-35; REC Opening Br. at 10; Microsoft Reply Br. at 14.)
5 Unsurprisingly, REC emphasizes the phrase of the passage “pertaining to that module,”
6 to argue that the search is limited to the requested module. In response, Defendants
7 emphasize the phrase “locate all of the associate information” to argue that all of the
8 information associated with the requested module must be searched including all modules
9 referenced by the requested module. The court finds that the natural reading of this
10 passage of the specification supports REC’s construction. Here, as REC asserts, the
11 phrase “pertaining to *that* module” strongly indicates a search of the information table
12 limited to the code module subject to the request. Although Defendants’ interpretation of
13 the passage is plausible, the court finds it forced. It is apparent that had the patentee
14 intended the search to include the requested module as well as all of the referenced
15 modules, it would have stated so in the passage. Instead the patentee chose to describe
16 his invention by limiting the search to “that module.”

17 Indeed, throughout the specification, the patentee makes clear when it is
18 referencing module information (also described as “associative information” or “linkage
19 information”) for a specific module as opposed to module information for any referenced
20 modules. For instance, in describing Figure 2 (shown above), the specification states,
21 “The main code module includes linkage information linking it with code module A,
22 code module B, and code module C.” (’936 Patent at 4:13-15.) Examining Figure 2

1 demonstrates that the main code module links to modules A, B, and C. (*Id.* at Figure 2.)
2 Thus, in accord with REC’s proposed construction, the specification described the
3 linkage information as only the information included in the specific module. For
4 Defendants’ proposed construction to be correct, the specification would have described
5 the linkage information as not only modules A, B, and C, but also the modules referenced
6 by modules A, B, and C. By contrast, when the patentee sought to describe the module
7 and its referenced modules, it did so expressly. For example, when referring to *all* of the
8 linkage information of Figure 2, the specification states, “Collectively the linkage
9 information referred to above forms an *associative set* for the main code module and code
10 modules A through E. (*Id.* at 4:25-27.) In sum, in describing its invention, the patentee
11 differentiated between module information relating to a specific module and module
12 information related to collective or referenced modules. As such, the disputed passage,
13 which recites a discrete module, supports REC’s position that the search is for only the
14 information related to that discrete module.

15 Citing to the Abstract of the specification, Defendants also argue that because the
16 stated purpose of the invention is to “relieve the first program [user] of the task of
17 searching for modules and extracting linkage information,” it would be pointless to
18 search for only a portion of the associative information stored in the module information
19 table. (Microsoft Opening Br. at 14-15.) The court acknowledges that in certain
20 instances a search for information pertaining to the discrete module and the referenced
21 modules may be advantageous. But, as explained above, the claim language and the
22 specification do not place such a limitation on the term-in-dispute. Here, the court

1 declines to read a limitation into the claims based on an advantageous configuration of
2 the patented invention not required by the intrinsic evidence.

3 In conclusion, having found that the claim language and the specification require
4 only that the search be performed for information related to discrete, requested module,
5 the court gives the term-in-dispute the following construction: “in response to a request
6 that includes an identification of the discrete module, the code server searches for module
7 information in the module information table pertaining to the discrete module.”⁷

8 **D. ”forming an association of said multi-module program by said first program”**

9 The disputed term is found in asserted claim 1 of the ‘936 Patent. The parties
10 propose the following constructions:

11 **REC’s Proposed Construction:** “The first program forms a data structure with
12 information concerning how one or more modules of the second multi-module program
13 link to one or more other modules.” (Updated Joint Claim Chart at 2.)

14 **Defendants’ Proposed Construction:** “The first program creates a table
15 containing all dynamic links for every module that the second multi-module program will
16 ever need.” (*Id.* at 2.)

17
18
19 ⁷ With little explanation, Defendants also argue that the specification describes an
20 iterative search process, whereby the search process does not stop until all references pertaining
21 to the requested module are located. (Microsoft Reply at 14.) Defendants cite to Figure 3 of the
22 Patent-in-Suit as well as passages from the specification describing Figure 3. (*Id.*) The court has
examined Defendants’ citations and finds Defendant’s reliance on the citations misplaced in the
context of the term-in-dispute, which relates to the search performed by the code server.
Defendants’ citations relate to an iterative process performed by the user computer, and not to an
iterative process performed by the code server, and therefore are not relevant to the term-in-
dispute.

1 REC aptly describes the parties' disputes as two minor disputes and one primary
2 dispute. (REC Opening Br. at 11.) The first minor dispute is whether the phrase
3 "forming an association" found in the term-in-dispute should be described as "how
4 modules link to other modules" (REC's position) or "dynamic links" (Defendants'
5 position). The second minor dispute is whether the phrase "forms a data structure"
6 (REC's position) or "creates a table" (Defendants' position) properly defines the
7 "forming an association" set forth in the claim language. Finally, the major dispute is
8 over whether "an association" may include linkage information for only part of a
9 complete program (REC's position) or whether "an association" must include linkage
10 information "for every module that the second multi-module program will ever need
11 (Defendants' position). The court discusses the three disputes in turn.

12 **i. "how modules link to other modules" vs. "dynamic links"**

13 REC supports its proposed language by a passage in the specification, which
14 states, "modules contain linkage information which point[s] to additional modules and
15 which collectively forms an association." (REC Opening Br. at 11 (citing '936 Patent at
16 3:14-16).) In response, Defendants assert that during prosecution of the parent patent to
17 the '936 Patent, the patentee described the association in terms of "dynamic links."
18 (Microsoft Opening Br. at 16-18.) Defendants provide various excerpts from the
19 prosecution history, but principally cite to the following portion, which is provided
20 without ellipses for context:

21 **Dynamic links which are used in conjunction with forming an**
22 **association**, as that term is defined by the present invention and used in
 applicant's specification and claims, generally refer to linkages used in

1 operating systems such as OS/2 from IBM or Windows 3.x from Microsoft,
2 for the loading and execution of code modules. In principle, under such
3 systems code modules are simply read into memory by the operating
4 system just prior to the multi-module program's execution in order to
5 extract module information so as to determine the multi-module program's
6 interrelationships between the code modules, and few modifications, if any,
7 are made to the code modules themselves. The code modules, unlike the
8 process of linking, do not address the other code modules directly, **but**
9 **instead address internal tables built within the operating system itself,**
10 **i.e., tables of dynamic links.** These dynamic link tables act under control
11 of the operating system as an intermediary addressing the other code
12 modules. For example, using such dynamic link tables provide the added
13 advantage of not requiring that all the code modules be loaded into memory
14 while determining the interconnections between them. Another advantage
15 of using dynamic link tables, over simply linking and loading as described
16 by Tennant, is that the code modules may be moved without requiring the
17 referencing modules to be changed, because references are effected via the
18 dynamic link tables which, in turn, represent and address the other code
19 modules. When a referenced code module is moved, the dynamic link
20 tables are simply updated to reflect its new location. **As used in the**
21 **specification and claims, applicant refers to the creation by the**
22 **operating system of the dynamic link tables necessary for the execution**
of a multi-mode program as "association". [sic] In contrast, the process
of linking and loading code pieces, as described by Tennant, is not what
applicant's invention refers to as association and Tennant's process is
directed to a system that is much more limited in nature.

(Microsoft Opening Br. at 18 (citing Prosecution History (Dkt. # 112-11) at 6-7 (bolding
and underlining in original).) The patentee made these statements to the United States
Patent and Trademark Office ("PTO") in an effort to overcome a rejection of two pieces
of prior art, "Tennant" and "Terada." (See Prosecution History at 3.)

Without citing the relevant law, Defendants apparently rely on this excerpt to
invoke the doctrine of prosecution disclaimer and argue that REC disavowed a
construction of "forming an association" to include anything other than "dynamic links."
(*Id.* at 17 ("REC distinguished its technique of 'forming an association,' which REC told

1 the PTO in no uncertain terms involved ‘dynamic linking,’ over the prior art Tennant in
2 order to gain allowance. No backsies.”.) “[S]tatements made during prosecution
3 may . . . affect the scope of the invention.” *Rexnord Corp. v. Laitram Corp.*, 274 F.3d
4 1336, 1343 (Fed. Cir. 2001). Specifically, “a patentee may limit the meaning of a claim
5 term by making a clear and unmistakable disavowal of scope during prosecution.”
6 *Purdue Pharma L.P. v. Endo Pharms., Inc.*, 438 F.3d 1123, 1136 (Fed. Cir. 2006). A
7 patentee could do so, for example, by clearly characterizing the invention in a way to try
8 to overcome rejections based on prior art. *See, e.g., Microsoft Corp. v. Multi-Tech Sys.,*
9 *Inc.*, 357 F.3d 1340, 1349 (Fed. Cir. 2004).

10 Having carefully reviewed the portions of the prosecution history cited by
11 Defendants, as well as the entire prosecution record at the PTO placed before the court,
12 the court can find no statements by the patentee clearly disavowing an association of
13 everything but “dynamic links.”⁸ Instead, the patentee distinguished the prior art
14 (Tennant and Terada) by arguing that its invention “assists in the association of a multi-
15 module program,” utilized in dynamic linking, whereas the prior art merely taught linking
16 and loading the modules themselves without the patented code server and information
17 table to assist. (Prosecution History at 8.) Said a different way, the patentee argued that
18 the prior art related to traditional static linking, whereas the novelty of the invention was

19
20 ⁸ As used throughout the specification and in the prosecution history, the term “dynamic
21 link” is a term of art describing the references between discrete modules in a dynamic linking
22 operating system. Limiting the term-in-dispute by a term of art may have the undesirable effect
of narrowing the scope of the claim to accused products which reference themselves by that term
of art.

1 the code server, including a module information table, to assist in creating an association
2 for a multi-module program in a dynamic linking environment. (*Id.* at 3 and 4 (“The
3 executable program thus created by Tennant is a single file that is static in nature,
4 meaning that the one executable linked file contains all the code pieces”)
5 (underlining in original).) Thus, the patentee distinguished its invention through the
6 novelty of assisting in the formation of an “association.” In making this distinction, the
7 patentee described the association as comprised of “dynamic links.” This is not a
8 situation for prosecution disclaimer. *Computer Docking Station Corp. v. Dell, Inc.*, 519
9 F.3d 1366, 1374-75 (Fed. Cir. 2008) (“Prosecution disclaimer does not apply to an
10 ambiguous disavowal. Prosecution disclaimer does not apply, for example, if the
11 applicant simply describes features of the prior art and does not distinguish the claimed
12 invention based on those features.”).

13 Although the court finds that the patentee did not disavow all associations except
14 for dynamic links, “the statements [in the prosecution history] may offer interpretative
15 assistance to the court in construing a particular claim.” *Prima Tek II, L.L.C. v. Polypap,*
16 *S.A.R.L.*, 318 F.3d 1143, 1149 (Fed. Cir. 2003). Here, the patentee’s repeated references
17 to the term “dynamic links” when describing the claimed “association” makes clear that
18 the patentee expected the association to be made up of dynamic links. Accordingly, the
19 court will incorporate the term “dynamic links” into the definition of the term-in-dispute
20 as an example of what may constitute an “association.”

1 **ii. “forms a data structure” vs. “creates a table”**

2 In the second minor dispute, the parties disagree on the proper description for the
3 phrase “forming an association,” found in the term-in-dispute. REC finds support for its
4 construction in dictionary definitions. (REC Opening Br. at 11-12.) REC further argues
5 that its proposed language—“forms a data structure”—will be less likely to confuse a
6 jury than Defendants’ proposed language. (*Id.*) Specifically, REC asserts that the word
7 “table,” found in Defendants’ proposed language, imputes rows and columns to the
8 structure of the “association,” which REC contends has no support in the record. (*Id.*)
9 Defendants disagree that the word “table” is more likely to confuse a jury than the term
10 “data structure,” and further argue that the specification supports a construction including
11 the word “table.” (Microsoft Resp. Br. at 18.) Here, the court agrees with REC.

12 As an initial matter, Defendants reliance on the specification is wholly misplaced.
13 Defendants direct the court to various portions of the specification which reference the
14 word “table.” (*Id.* at 17-18.) Each and every one of Defendants’ citations refer to the
15 code module information table, found in the claim language. (*Id.*) The code module
16 information table is certainly a table, but it is also certainly distinct from the
17 “association,” that is formed. Indeed, it is the code module information table which
18 assists in creating the “association” by the “first program” or user. (*See* ’936 Patent at
19 Claim 1 (“(c) in response to finding said module information in said *module information*
20 *table*, reading said module information; (d) providing said module information in a
21 response to said first program; and (e) *forming an association of said multi-module by*
22 *said first program.*”) (emphases added).) Therefore, that the specification describes the

1 code module information table as a “table” is irrelevant to the correct structure of the
2 “association.”

3 It appears that once again the specification is of little guidance in determining the
4 structure of the association. As stated, in such a situation, the court will adopt a
5 “construction that stays true to the claim language and most naturally aligns with the
6 patent’s description of the invention.” *Phillips*, 415 F.3d at 1316. Here, the claim
7 language leaves little doubt that the “association” is formed by the first program (the user
8 program). Having already defined the word “program” as a set of computer instructions,
9 it is logical that any creation of a computer program will be some sort of “data structure,”
10 as set forth by REC. Moreover, such a description of “association” aligns with the
11 overall nature of the Patent-in-Suit.

12 **iii “one or more modules of the program” vs. “every module that the**
13 **program will ever need”**

14 The major issue between the parties turns on whether an “association,” as recited
15 in the claim language, must include linkage information “for every module that the
16 second multi-module program will ever need,” or may include linkage information for
17 “just parts of” the second multi-module program. REC argues that the claim language
18 and specification support a construction of “association” including linkage information
19 for *only parts of* a complete second multi-module program (as well as linkage
20 information for the entire second multi-module program). (REC Opening Br. at 12.)
21 Defendants respond that the prosecution history dictates a limiting construction of
22 “association” requiring an “association” to include linkage information for every module

1 | the second multi-module program will ever need. (Microsoft Opening Br. at 18-19.) The
2 | court addresses the parties' arguments in turn.

3 | The court agrees with REC that the claim language supports a construction of
4 | “association” including linkage information for only parts of a complete second multi-
5 | module program. Claim 1 of the '936 Patent recites a method “for forming an
6 | association” comprised of five distinct steps, (a) through (e). (*See* '936 Patent at Claim
7 | 1.) The method begins with step (a) and “receiving . . . a request associated with said
8 | discrete module.” The claim language of step (a) describes one discrete module, as
9 | opposed to a request for every module the second multi-module program will ever need.
10 | Next, step (b) is “searching . . . for module information in response to said requested
11 | associated with said discrete module.” The court has already construed this term (*supra* §
12 | III.C) to require only a search for linkage information related to the discrete module (and
13 | not also a search for information of modules referenced by the discrete module). In
14 | response to finding module information through the search of step (b), step (c) reads the
15 | module information. Again, step (c) relates to only reading information related to the
16 | discrete module. Then, step (d) provides the module information to the first program.
17 | Logically, the module information provided by step (d) is only that of the discrete
18 | module. Finally, in step (e) the first program forms an association of the second multi-
19 | module program. Because each of the aforementioned steps (a) through (d) pertain only
20 | to a discrete module, the only conclusion is that the formed “association” also pertains
21 | only to the discrete module. Thus, the claim language supports a construction of
22 |

1 “association” including linkage information for only parts of a complete second multi-
2 module program, indeed, only a discrete module of the second multi-module program.

3 Consistent with the claim language, the specification expressly contemplates an
4 “association” comprised of less than a complete second multi-module program. REC
5 aptly directs the court to the following two passages from the specification (REC
6 Opening Br. at 12.):

7 “The code server . . . includes an information storage table containing
8 linkage information needed to form an association between discrete
modules of code forming at *least parts* of a coded program.”

9 (’936 Patent at 2:12-18 (emphasis added).) And, also:

10 “[T]hese modules contain linkage information which point to additional
11 modules and which collectively forms an *association linking together at
least parts of a complete program.*”

12 (*Id.* at 3:14-16 (emphasis added).) These passages make clear that the “association” need
13 not constitute the entire second multi-module program, but can be something less, such as
14 “parts of a complete program.” Thus, the court finds that the specification supports
15 REC’s position.⁹

17
18 ⁹ The parties partake in a dispute as to the iterative nature of the invention. Defendants
19 argue that the iterative nature of the invention indicates that the “association” created constitutes
20 all the linkage information the second multi-module program will ever need. Principally, the
21 parties argue over Figure 3 of the ’936 Patent. The court has examined Figure 3 and the
22 portions of the specification explaining Figure 3 and finds them ambiguous as the iterative nature
of the invention. On one hand, the flow chart in Figure 3 may be interpreted such that the “user”
program continues a search until it knows all of the linkage information of the second multi-
module program. On the other hand, as the claim language and specification indicate that the
“user” program may request discrete modules or only part of the complete second multi-module
program, the Figure may be interpreted to mean that the “user” program continues a search until
it knows all of the linkage information that it seeks via its current search, which could be less

1 Nevertheless, Defendants assert, again without citing the relevant law, that the
2 prosecution history dictates a limited scope for the term-in-dispute requiring an
3 “association” to include linkage information “for every module that the second multi-
4 module program will ever need.” (Microsoft Opening Br. at 18-19.) Here, again,
5 Defendants seek to limit the scope of a term based on the prosecution history, thereby
6 invoking the doctrine of prosecution disclaimer set forth above. Defendants again
7 principally rely on the same passage from the prosecution history (*supra* § III.D.i), and
8 particularly on the following sentence: “As used in the specification and claims,
9 applicant refers to the creation by the operating system of the dynamic link tables
10 *necessary for the execution of a multi-mode program as ‘association’.*” (Microsoft
11 Opening Br. at 18-19.) Keying on the word “necessary,” Defendants assert that the
12 “association” must include “all dynamic links necessary for the execution of the
13 program.” (*Id.*)

14 The court has examined Defendants’ cited sentence within the context of the
15 prosecution history and does not find that it amounts to a clear disavowal of claim scope.
16 In Defendants cited prosecution reference, the patentee was generally arguing to
17 overcome a rejection of the Tennant and Terada prior art. (*See generally* Prosecution
18 History.) The patentee made two general arguments to overcome that prior art. First (as
19 explained, *supra* § III.D.i), the patentee argued that the prior art did not teach any tool,
20 such as a code server and code module information table, for assisting in the formation of

21
22 than a completed second multi-module program. Thus, because of this ambiguity, the court finds
that Figure 3 has little relevance to its construction of the term “association.”

1 | an association. (*See* Prosecution History at 8 (“Tennant only teaches the linking and
2 | loading of programs . . . distinguished from applicant’s claimed system which assists in
3 | the formation of an association.”).) Second, the patentee argued that in the prior art the
4 | operating system itself read code modules directly in order to extract the module
5 | information for creating the association. (Prosecution History at 6 (in the prior art, “code
6 | modules are simply read into memory by the operating system . . . in order to extract
7 | module information so as to determine the multi-module program’s interrelationships
8 | between the code modules”).) But, in the patented invention, the code modules are not
9 | accessed directly to determine linkage information, but instead through internal tables
10 | which contain information about the relationships between the code modules. (*Id.* at 6-7
11 | (“The code modules, unlike the process of linking do not address other code modules
12 | directly, but instead address internal tables built within the operating system itself, i.e.,
13 | tables of dynamic links.”).) Importantly, at no point did the patentee distinguish the prior
14 | art on the basis that its invention provides for an association containing *all* the linkage
15 | information the second program will ever need, whereas the prior art only taught creating
16 | an association for less than all of the linkage information the second program will ever
17 | need. In fact, at no point in the prosecution history did the patentee even distinguish the
18 | prior art based on the amount of module information that constitutes an “association.”

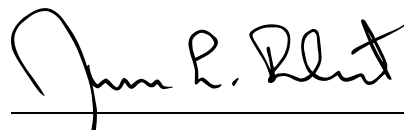
19 | Here, Defendants only argue that the patentee’s description of “association” as
20 | “necessary for the execution of a multi-mode program” limited the scope of the term to
21 | “every module that the second multi-module program will ever need.” The court
22 | disagrees. This statement by the patentee did not disclaim an association that includes

1 | *some* linkage information necessary for execution of the multi-module program.
2 | Certainly, an association including only some of the linkage information for an entire
3 | multi-module program will still be necessary for execution of the multi-module program.
4 | At most this statement is ambiguous as to the amount of module information constituting
5 | an association. *Sandisk Corp. v. Memorex Prods.*, 415 F.3d 1278, 1287 (Fed. Cir 2005
6 | (“There is no ‘clear and unmistakable’ disclaimer if a prosecution argument is subject to
7 | more than on reasonable interpretation.”); *Golight, Inc. v. Wal-Mart Stores, Inc.*, 355
8 | F.3d 1327, 1332 (Fed. Cir. 2004) (holding that statements in the prosecution history
9 | subject to multiple interpretations do not constitute a clear and unmistakable disavowal).
10 | And, as explained, the statement in question was not made to distinguish the invention
11 | from the prior art. Thus, the court cannot find that REC clearly disavowed the scope of
12 | the term “association” as Defendants suggest. *See Computer Docking Station.*, 519 F.3d
13 | at 1374-75.

14 | In accord with the foregoing, the court construes the term “forming an association
15 | of said multi-module program by said first program” to mean “the first program forms a
16 | data structure with information (such as dynamic links) concerning how one or more
17 | modules of a second multi-module program link to one or more other modules.”

18 | IT IS SO ORDERED.

19 | Dated this 1st day of May, 2012.

20 | 
21 | _____

22 | JAMES L. ROBERT
United States District Judge