UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

APPLE, INC.
Petitioner,

v.

EVOLUTIONARY INTELLIGENCE, LLC,
Patent Owner

Case IPR 2014-00086
Patent No. 7,010,536

OBSERVATIONS ON CROSS-EXAMINATION OF HENRY HOUH

As set forth by 77 Fed. Reg. 48756, 48762-3, Patent Owner Evolutionary Intelligence, LLC, hereby files these observations on the deposition testimony of Henry Houh, Ph.D., held on December 2, 2014 at 11:00 a.m. EST in the offices of Sidley Austin LLP, 1501 K Street NW #600, Washington, D.C. 20005 regarding his supplemental declaration (Ex. 1009).

(1) **Reliance on Inherent Disclosure to Establish Anticipation:** When asked whether there was something in Gibbs' disclosure that made its alleged "execution stack" an example of a logically defined data enclosure, Dr. Gibbs admitted that his interpretation of Gibbs is based on reading "necessary components" into Gibb's actual disclosure. *See* Ex. 1010 at 257:15-258:12 ("[O]ne of skill in the art would have understood Gibbs to disclose all the necessary components . . . to support execution of the program.") and 258:22-259:3 (referring to execution stack as something that "people would commonly understand to be there."). His testimony confirms that Dr. Houh is relying on inherent anticipation to supply elements required of claim 2 of U.S. Patent 7,010,536, yet did not address all elements of that theory in his declaration in support of the petition for IPR. *See* P.O. Resp. at 37-41.

(2) **Inability to Establish Inherent Disclosure of All Elements of Claim 2:** Dr. Houh admitted that the execution stack that he was reading into Gibbs in ¶ 29 of his Supplement Declaration did not necessarily function in the manner he relied on in his anticipation analysis (i.e., did not necessarily nest all the elements of the system of Gibbs within one "logically defined data enclosure"). *See* Ex. 1010 at 258:22-259:3 ("It's very hard to make a statement that says everything in the world . . . has this [i.e., an execution stack functioning as described by Dr. Houh] because, you know, ***there probably could be someone who could come up with a system for supporting function calls without this type of execution stack*** . . . ."). This rebuts any attempt by Apple to argue that Gibbs necessarily discloses an execution stack that nests all the objects disclosed as discrete parts of its system into one "container."

(3) Lack of Active Space Register – Dr. Houh admitted that the text in Gibbs he cites as support for use of latitude and longitude as *active space registers* as required by claim 2 actually discloses the latitude and longitude being retrieved *after* the system of Gibbs does a time comparison to determine whether the trains are late. (Rough at 85:8-85:24.) This admission contradicts Dr. Houh's declaration testimony that the latitude and longitude values "trigger" identification of those trains as late. *See* Ex. 1003 at ¶ 127; Ex. 1009 at ¶¶ 49-53.

(4) First Register Having a Unique Container Identification Value– Dr. Houh testified that a "content key register" was a register containing a "keyword" such as "automobiles." Ex. 1010 at 277:3-6 (Q. . . . [C]an you think of an example of what a content key register would be? A. It might be a work like "automobiles," for example."). Dr. Houh further testified that a single "content key register" could identify multiple containers. Ex. 1010 at 278:22-23 ("that key refers to the same ***set of containers*** in either case."); 279:4-7 ("That [i.e., the keyword] would refer to the same set of objects, each of which, you know, had the word "automobiles" in – in the proper field.") As such, the "content key register" cannot be an example of a "first register having a unique container identification value."

(5) Lack of Foundation for Opinions – Dr. Houh admitted that he did not review any material relating to object-oriented programming (i.e., textbooks, research papers, or other "corroborating evidence") in preparation for his deposition testimony, nor in preparing his declarations. (Ex. 1010 at 201:7-203:6; 302:9-306:24.) As such, Dr. Houh has admitted that he has provided no corroborating evidence for his testimony.

(6) Failure to Apply Claim Construction from Perspective of One of Ordinary Skill in the Art – Dr. Houh admitted that he applied a "plain language" definition of "nesting" as "things within things" as his construction for the phrase "logically defined data enclosure." *Id.* at 240:5-243:5 (" . . . generally people understand

[nesting] as things within things"). As such, Dr. Houh applied the wrong legal standard for claim construction, and he applied an incorrect construction of "logically defined data enclosure."

(7) Failure to Establish that Each Object in Gibbs is a "Logically Defined Data Enclosure" that Includes All the Other Objects in Gibbs – When asked for concrete examples of how programmers would "nest" objects within a class interface, polymorphic object, class with inheritance, or "contiguous blocks of memory," Dr. Houh supplied examples which involved a written declaration within the program placing one object within another object. *See e.g.*, Ex. 1010 at 208:7-109:4 (Q. "And how would you create a class interface?" A. ". . . there are generally ways of declaring and defining classes and . . . declaring and defining the public methods of a class. . . . But, you know, ***if you just want to declare a class and define it . . . then you could just start typing***."); 209:6-210:2 (Q. "And is it your understanding that in creating a class interface, you could nest containers within that class interface?" . . . A. "Well, in . . . object oriented programming methods, ***one can declare a class*** and . . . ***declare members of the class***."); 210:4-17 (Q. "So when you talk about declaring things within a class, what do you mean?" . . . . A. ". . . You can create ***classes*** that ***have state***, and states are contained in – ***in a description of the class***."); 213:15-214:20 (Q. "Okay. So if as an object-oriented programmer you wanted to nest one container within another

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.