



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

Table with 5 columns: APPLICATION NO., FILING DATE, FIRST NAMED INVENTOR, ATTORNEY DOCKET NO., CONFIRMATION NO. Includes details for application 13/450,788, inventor DORON LEVI, and examiner MILLER, VIVA L.

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

- hpe.ip.mail@hpe.com
mkraft@hpe.com
chris.mania@hpe.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Ex parte DORON LEVI

Appeal 2015-006820
Application 13/450,788¹
Technology Center 2100

Before JEAN R. HOMERE, JOHN A. EVANS, and
DANIEL J. GALLIGAN, *Administrative Patent Judges*.

Per Curiam.

DECISION ON APPEAL²

Appellant seeks our review under 35 U.S.C. § 134(a) of the Examiner's final rejection of claims 1–15. We have jurisdiction under 35 U.S.C. § 6(b).

We affirm.

¹ The Appeal Brief identifies as the real party in interest Hewlett-Packard Development Company, LP, a wholly-owned affiliate of Hewlett-Packard Company, having HPQ Holdings, LLC, as its general or managing partner. App. Br. 3.

² Our Decision refers to Appellant's Appeal Brief filed March 4, 2015 ("App. Br."); Appellant's Reply Brief filed July 16, 2016 ("Reply Br."); Examiner's Answer mailed May 21, 2015 ("Ans."); and original Specification filed April 19, 2012 ("Spec").

STATEMENT OF THE CASE

Claims on Appeal

Claims 1, 5, and 15 are independent claims. Claim 1 is reproduced below (with disputed limitations in italics):

1. A testing system for an integrated software system comprising:
a mock object implemented as machine executable instructions on a first non-transitory computer readable medium, *the mock object implemented as a stateless proxy* associated with a corresponding real object in the integrated software system; and
a scenario, implemented as machine executable instructions on one of the first non-transitory computer readable medium and a second non-transitory computer readable medium, to store configuration data for the mock object representing methods associated with the real object.

References

Li et al.	US 2007/0277158 A1	Nov. 29, 2007
Kinnucan et al.	US 2008/0092111 A1	Apr. 17, 2008
Atkin et al.	US 2008/0256517 A1	Oct. 16, 2008

Examiner's Rejections

Claims 1–9 and 11–15 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Atkin and Li. Ans. 3–13.

Claim 10 stands rejected under 35 U.S.C. § 103(a) as unpatentable over Atkin, Li, and Kinnucan. Ans. 13–14.

ANALYSIS

Claim 1

Appellant contends the Examiner erred in finding Li teaches a “mock object implemented as a stateless proxy,” as recited in independent claim 1 and similarly recited in claims 5 and 15. App. Br. 7, 11, 12. In particular,

Appellant argues that the characteristic of a partner stub calling the mock object does not indicate whether the mock object is stateless or stateful. *Id.* Appellant asserts that Li teaches that “the [partner] stub can call a method from the mock [object] without reference to an external data source.” App. Br. 8 (citing Li ¶ 39). Appellant asserts Appellant’s Specification describes that, because the mock object is stateless, its behavior can only be changed by changing the scenario and that the mock object cannot be configured. App. Br. 9 (citing Spec. ¶ 22). Appellant contends that a mock object reacting to input without any external reference indicates that the mock object is stateful having locally stored behaviors. Reply Br. 5.

The Examiner interprets Appellant’s Specification to describe a “mock object implemented as a stateless proxy” to include a mock object whose behavior can be changed. Ans. 16 (citing Spec. ¶ 22). The Examiner finds Li teaches that the behavior of the mock object can be changed using the test cases such that the mock object conforms to the description in Appellant’s Specification of the mock object as a stateless proxy. Ans. 16–17 (citing Li ¶ 77). We agree with the Examiner. Li explains how the test cases change the behavior of the mock object:

Test behavior support: MockObjects may provide some flexible behavior description and verification mechanism. For example, EasyMock has three types of MockControl. The normal one will not check the order of expected method calls. Another strict one will check the order of expected method calls. For these two types, an unexpected method call on the mock object will lead to an Assertion-FailedError. The remaining nice one is a more loose version of the normal one, it will not check the order of expected method calls, and an unexpected method call will return an empty value (0, null, false).

Li ¶ 79. Li teaches types of MockControl that output values to which the MockObjects respond. *See id.* For some types of MockControls, an Assertion-FailedError output prevents the MockObject from being invoked for an out-of-order or unexpected method call; for other types, an empty value output does not prevent the MockObject from being invoked. *See id.* This teaching mirrors Appellant's Specification's description of changing the behavior of a mock object by replacing the scenario with a new scenario. App. Br. 9 (citing Spec. ¶ 22); Ans. 16 (citing Spec. ¶ 22). Thus, the MockControl in Li teaches the claimed scenario. *See Spec. ¶ 11.*

The scenario 22 can include a programmed collection of steps for each unique method signature associated with the mocked real object to model its behavior in response to invocation of the mock object. For example, the programmed behaviors can include return values, output and reference parameter values, exception throwing, event raising, callback execution, and similar behaviors. During execution, the mock object 12 refers to the scenario 22 to determine how it should proceed when an associated method is invoked.

Spec. ¶ 11. In Li, the MockControl outputs, in response to a method call, a return value, an Assertion-FailedError or an empty value, to which the MockObject refers. *See Li ¶ 79.* Li teaches that a MockObject with no apparent locally stored behaviors reacts to input with the external reference of the MockControl. *See Li ¶ 79.* This indicates that the mock object is stateless. *See Reply Br. 5.* Thus, we find the evidence supports the Examiner's finding that Li teaches or suggests a mock object implemented as a stateless proxy. In view of the foregoing, we sustain the rejection of claim 1. Appellant does not present additional persuasive arguments regarding dependent claims 2–4, and, therefore, we sustain the rejection of these claims.

Claims 5 and 15

Appellant argues Atkin does not teach “generating a scenario as a hierarchical data object in which configuration parameters for each of a plurality of methods associated with a mock object are related to an associated method signature,” as recited in claims 5 and 15. App. Br. 11, 12. Specifically, Appellant contends Atkin teaches “the use of a captor code to record methods for the creation of the mock objects themselves, not the recited scenario.” App. Br. 11 (citing Atkin ¶ 84); Reply Br. 6–7. The Examiner finds Atkin teaches the captor code monitors the previous execution to determine how the mock object reacts to certain method calls used to generate unit test cases. Ans. 20–21 (citing Atkin ¶ 84).

The Examiner finds the unit test cases in Atkin act as the recited scenario. Ans. 21. We agree with the Examiner’s findings, which are consistent with how the claimed subject matter is described in Appellant’s Specification. For example, Appellant cites paragraph 19 of Appellant’s Specification as support for the claimed subject matter. App. Br. 5 (citing Spec. ¶ 19). Paragraph 19 provides, in part: “[A] recording component can be used to capture the desired behavior of the mock object and store it in the scenario, which is a complex data structure called that relates the configuration uniquely to type and method signatures associated with the mock object.” The Examiner finds the captor code in Atkins teaches the recording component described in Appellant’s Specification which is used in the generation of unit test cases. *See* Ans. 20–21 (citing Atkin ¶ 84). As Appellant notes, Atkin teaches the mock objects are generated before the unit test cases. Reply Br. 7 (citing Atkin ¶¶ 43, 84). Similarly, Appellant’s

Appeal 2015-006820
Application 13/450,788

Specification describes that the mock object exists before the scenario. *See* Spec. ¶ 19.

As such, we are not persuaded of error in the Examiner's rejection of claims 5 and 15, and we sustain the rejection of claims 5 and 15. Appellant does not present additional persuasive arguments regarding dependent claims 6–14 which depend from claim 5 (App. Br. 11–12), and, therefore, we sustain the rejections of these claims.

DECISION

We affirm the Examiner's rejection of claims 1–15.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED