EXHIBIT B

MATTHEW D. POWERS (Bar No. 104795)
matthew.powers@weil.com
STEVEN S. CHERENSKY (Bar No. 168275)
steven.cherensky@weil.com
WEIL, GOTSHAL & MANGES LLP
Silicon Valley Office
201 Redwood Shores Parkway
Redwood Shores, CA 94065
Telephone: (650) 802-3000
Facsimile: (650) 802-3100

TIMOTHY DEMASI
tim.demasi@weil.com
STEVEN KALOGERAS
steven.kalogeras@weil.com
JULIAN MOORE
julian.moore@weil.com
WEIL, GOTSHAL & MANGES LLP
New York Office
767 5th Avenue
New York, NY 10153
Telephone: (212) 310-8000
Facsimile: (212) 310-8007

Attorneys for Plaintiffs
SAMSUNG ELECTRONICS CO., LTD.
SAMSUNG ELECTRONICS AMERICA, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

CV11 0189 MEJ

SAMSUNG ELECTRONICS CO., LTD. and SAMSUNG ELECTRONICS AMERICA, INC., Plaintiffs, v. VERTICAL COMPUTER SYSTEMS, INC., Defendant.	Case No. **COMPLAINT FOR DECLARATORY JUDGMENT** **DEMAND FOR JURY TRIAL**

Plaintiffs Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc. (collectively "Samsung"), by and through their attorneys, bring this action against Vertical Computer Systems, Inc. ("Vertical") and allege as follows:

INTRODUCTION

1. Samsung seeks a declaratory judgment that (i) Samsung has not infringed any claim of United States Patent Nos. 6,826,744 ("the '744 patent") or 7,716,629 ("the '629 patent); and (ii) each and every claim of the '744 and '629 patents is invalid and/or unenforceable. The '744 and '629 patents are attached as Exhibit A and Exhibit B, respectively.

2. The Northern District of California is already presiding over a related declaratory judgment action brought by Interwoven, Inc. ("Interwoven") against Vertical, Civil Case No. 10-CV-4645 ("the California action"), which similarly seeks a judgment of invalidity, unenforceability and non-infringement of the '744 and '629 patents. The California Action was filed on October 14, 2010.

3. Over one month later, on November 15, 2010, Vertical filed suit in the Eastern District of Texas, Civil Case No. 2:10-CV-490 (the "Texas action"), accusing Interwoven, Samsung, and two LG Electronics entities of infringing the same '744 and '629 patents.

4. Vertical's initiation of the Texas action has resulted in a duplicative and wasteful litigation in a new forum of the same issues being litigated in the California action. Samsung has given notice to this Court that the present declaratory judgment action is related to the first-filed California action so that the two actions may be consolidated and that the parties may proceed in a manner that promotes judicial economy, avoids the potential for conflicting rulings, and leads to an efficient resolution of these overlapping disputes.

PARTIES

5. Samsung Electronics Co., Ltd. is a corporation organized under the laws of Korea, with a principal place of business at Samsung Electronics Building, 1320-10, Seocho-2-dong, Seochu-gu, Seoul 137-857, Korea.

6. Samsung Electronics America, Inc. is a corporation organized under the laws of New York, with a principal place of business at 105 Challenger Road, Ridgefield Park, New Jersey 07660. Samsung Electronics America, Inc. is a wholly owners subsidiary of Samsung Electronics Co., Ltd.

7. On information and belief, Vertical is a company organized under the laws of Delaware, with a principal place of business at 101 W. Renner Road, Suite 300, Richardson, Texas 75082.

EXISTENCE OF AN ACTUAL CONTROVERSY

8. There is an actual controversy within the jurisdiction of this Court under 28 U.S.C. §§ 2201 and 2202.

9. Vertical purports to be the owner of the '744 and '629 patents (collectively, "the Patents-in-Suit").

10. On November 15, 2010, Vertical filed the Texas action, alleging that Samsung infringes the Patents-in-Suit. Specifically, Vertical alleges that Samsung infringes claims 1, 3-5, 9, 17, 21, 23 and 25 of the '744 patent and claims 1, 4, 8, 10, 12, 21, 24, 28, 30, and 32 of the '629 patent, both directly and indirectly, through the commercialization of the Samsung Galaxy Tab™ Android tablet and Samsung's Captivate™, Fascinate™, Epic™ and Mesmerize™ i500 Android cell phones.

11. Samsung denies that it infringes any of the Patents-in-Suit, and disputes their validity. Samsung further denies that it needs a license to any of the Patents-in-Suit in order to continue its activities. Thus, an actual and justiciable controversy exists between Vertical and Samsung as to whether the Patents-in-Suit are infringed and/or invalid.

JURISDICTION AND VENUE

12. This Court has subject matter jurisdiction over Samsung's Declaratory Judgment claims pursuant to 28 U.S.C. §§ 2201 and 2202.

13. This Court also has original subject matter jurisdiction over the claims asserted herein pursuant to 28 U.S.C. §§ 1331 and 1338(a).

14. On information and belief, Vertical is subject to personal jurisdiction in this district arising out of its contacts with this district as well as Vertical's acquiescence to this district's jurisdiction in the California action. In particular, Vertical has filed a motion to dismiss and/or transfer in the California action without raising the defense of lack of personal jurisdiction, thereby waiving any objection to the propriety of personal jurisdiction in this district.

15. Venue is proper in this district pursuant to 28 U.S.C. § 1391 because, for the reasons set forth above, Vertical is subject to personal jurisdiction within this judicial district.

FIRST CLAIM – THE '744 PATENT

16. Samsung hereby restates and realleges the allegations set forth in paragraphs 1 through 15 above and incorporates them by reference.

17. No claim of the '744 patent has been or is infringed, either directly or indirectly, or either literally or under the doctrine of equivalents, by Samsung.

18. The claims of the '744 patent are invalid for failure to comply with the requirements of the Patent Laws of the United States, including but not limited to the provisions of 35 U.S.C. §§ 101, 102, 103, and/or 112.

19. The claims of the '744 patent are unenforceable as a result of inequitable conduct before the United States Patent and Trademark Office ("PTO"). One or more of the people substantively involved in the prosecution of the application leading to the '744 patent, including inventor Aubrey McAuley, were aware of information material to the patentability of the '744 patent, but withheld that information from the PTO with the intent to deceive, and made false and misleading statements to the PTO during the prosecution of the '744 patent, as set forth herein.

20. Aubrey McAuley, the named inventor of the '744 patent, was a founder and president of Adhesive Media, Inc. ("Adhesive").

21. On information and belief, Adhesive offered for sale and sold software products and/or services based on its "WebOS" technology more than one year before October 1, 1999, the filing date of the application leading to the '744 patent. Those software products and/or services included Adhesive's "NewsFlash" product, as well as a number of websites that Adhesive designed for particular customers, purportedly using Adhesive's WebOS technology.

22. On information and belief, Adhesive also published information relating to its software products and/or services based on its "WebOS" technology more than one year before October 1, 1999. For example, more than one year before October 1, 1999, Adhesive posted on the Internet a diagram of the Web Object Management Facility of its WebOS

technology, which is attached hereto as Exhibit C.

23. Exhibit C is nearly identical to Figure 5 of the '744 patent.

24. The specification of the '744 patent describes Figure 5 as an alleged embodiment of "the present invention." See '744 patent at 5:3-17.

25. Therefore, Adhesive's commercial offer for sale and sale of products and services based on its WebOS technology, and Adhesive's publication of information relating to its WebOS technology, all of which occurred more than one year before October 1, 1999, constitute material prior art.

26. Upon information and belief, prior to issuance of the '744 patent, Mr. McAuley had knowledge of Adhesive's offer for sale and sale of its WebOS technology and Adhesive's publication of information relating to its WebOS technology.

27. None of the persons involved in the prosecution of the '744 patent, including but not limited to Mr. McAuley, disclosed to the PTO Adhesive's offer for sale or sale of it WebOS technology or the publication of information relating to its WebOS technology.

28. Information regarding Adhesive's offer for sale and sale of WebOS technology, and publications relating thereto, was withheld from the PTO with intent to deceive.

29. This withholding of information material to patentability with intent to deceive the PTO constitutes inequitable conduct, which renders the '744 patent unenforceable.

SECOND CLAIM – THE '629 PATENT

30. Samsung hereby restates and realleges the allegations set forth in paragraphs 1 through 29 above and incorporates them by reference.

31. No claim of the '629 patent has been or is infringed, either directly or indirectly, or either literally or under the doctrine of equivalents, by Samsung.

32. The claims of the '629 patent are invalid for failure to comply with the requirements of the Patent Laws of the United States, including but not limited to the provisions of 35 U.S.C. §§ 101, 102, 103, and/or 112.

33. The claims of the '629 patent are unenforceable as a result of inequitable conduct before the PTO. On information and belief, one or more of the people substantively

involved in the prosecution of the application leading to the '629 patent, including inventor Aubrey McAuley and patent agent Jack D. Stone Jr., were aware of information material to the patentability of the '629 patent, but withheld that information from the PTO with the intent to deceive, and made false and misleading statements to the PTO during the prosecution of the '629 patent, as set forth herein.

34. During the prosecution of the '629 patent, Vertical initiated a patent infringement suit against Microsoft Corporation in the Eastern District of Texas, Civil Action No. 2:07-CV-144 ("the Microsoft litigation"), alleging infringement of the '744 patent.

35. During the course of the Microsoft litigation, material information regarding the patentability of the '744 patent was disclosed by Microsoft to Vertical and Vertical's attorneys. For example, Microsoft raised inequitable conduct allegations regarding the '744 patent in its Answer to Vertical's complaint, Microsoft served invalidity contentions explaining how numerous prior art references anticipated and/or rendered obvious the claims of the '744 patent, Microsoft produced copies of the underlying prior art references, and Microsoft filed a claim construction brief arguing that numerous claims of the '744 patent were invalid under 35 U.S.C. § 112. However, this material information was not properly disclosed to the PTO during the prosecution of the '629 patent.

36. Because the application leading to the '629 patent is a continuation of the '744 patent, and because the claims and specifications of the '629 and '744 patents are substantially similar, Microsoft's inequitable conduct allegations, invalidity contentions and arguments, and the invalidating prior art references it produced in the Microsoft litigation are also material to the patentability of the '629 patent.

A. Microsoft's Inequitable Conduct Allegations

37. On July 13, 2007, Microsoft filed its Answer, Affirmative Defenses, and Counterclaims ("Microsoft's Answer") in the Microsoft litigation. Microsoft alleged that the '744 patent was unenforceable due to the inequitable conduct of Mr. McAuley in failing to disclose material information to the PTO. In particular, Microsoft alleged that Mr. McAuley, with intent to deceive, failed to disclose Adhesive's offer for sale and sale of products and

services based on Adhesive's WebOS technology, and publications relating thereto, more than one year prior to October 1, 1999.

38. Microsoft's inequitable conduct allegations disclose critical information expressly challenging the validity and enforceability of the related '744 patent, and thus constitute material prior art.

39. None of the persons involved in the prosecution of the '629 patent, including Mr. McAuley and Mr. Stone, disclosed to the PTO either Microsoft's Answer or the existence or substance of Microsoft's inequitable conduct allegations.

40. Further, during prosecution of the '629 patent, the applicants disclosed certain prior art documents relating to Adhesive's prior art WebOS technology relied on by Microsoft during the Microsoft litigation, but failed to disclose the critical facts that Mr. McAuley was the founder and president of Adhesive and other information indicating that the WebOS technology qualified as prior art under 35 U.S.C. § 102(b).

41. The knowledge that Mr. McAuley is both a named inventor of the '629 patent and the founder and president of Adhesive, as well as the date of the WebOS materials, is essential for the PTO to fully understand the relevance and applicability of Adhesive's prior art WebOS technology.

42. During prosecution of the '629 patent, the applicants selectively disclosed to the PTO only certain information and prior art materials from the Microsoft litigation.

43. The selective disclosure to the PTO of information arising out of the Microsoft litigation demonstrates that Mr. McAuley and Mr. Stone were aware of the Microsoft litigation and the existence of material regarding the patentability of the '629 patent information arising out of that litigation.

44. This selective disclosure to the PTO also demonstrates that Mr. McAuley and Mr. Stone made a deliberate decision to withhold material information from the PTO, and thus demonstrates an intent to deceive.

45. The withholding of information material to patentability with intent to deceive constitutes inequitable conduct, which renders the '629 patent unenforceable.

B. Microsoft's Invalidity Contentions and Claim Construction Brief

46. On January 18, 2008, Microsoft served its Invalidity Contentions in the Microsoft litigation.

47. Microsoft's Invalidity Contentions identified 58 prior art references that anticipated and/or rendered obvious claims 1-5, 9, 11, 17-19, 21, 23, 25-29, 33, 39-41, 43, 45, and 47-48 of the '744 patent and provided over 50 pages of narrative analysis of how the identified prior art anticipated and/or rendered obvious the asserted claims. The Invalidity Contentions also included over 250 pages of claim charts mapping the prior art references to each limitation of the asserted claims. Further, the Invalidity Contentions include an analysis of the '744 patent's invalidity based on lack of enablement, lack of written description, and indefiniteness.

48. On July 18, 2008, Microsoft served its First Amended Invalidity Contentions in the Microsoft litigation.

49. Microsoft's First Amended Invalidity Contentions added three prior art references to Microsoft's prior Invalidity Contentions, identifying a total of 61 prior art references that anticipated and/or rendered obvious claims 1-5, 9, 11, 17-19, 21, 23, 25-29, 33, 39-41, 43, 45, 47-48, and 53 of the '744 patent. As with the initial Invalidity Contentions, Microsoft First Amended Invalidity Contentions provided over 50 pages of narrative analysis of how the identified prior art anticipated and/or rendered obvious the asserted claims and included over 250 pages of claim charts mapping the prior art references to each limitations of the asserted claims. Further, the First Amended Invalidity Contentions include an analysis of the '744 patent's invalidity based on lack of enablement, lack of written description, and indefiniteness.

50. On June 6, 2008, Microsoft filed its Claim Construction Brief in the Microsoft litigation.

51. Microsoft argued in its Claim Construction Brief that the term "arbitrary object framework" is fatally indefinite.

52. The term "arbitrary object framework" is found in all independent claims of both the '744 and '629 patents, making Microsoft's indefiniteness argument material to the patentability of the '629 patent.

53. Microsoft's Invalidity Contentions, First Amended Invalidity Contentions, and Claim Construction Brief disclose critical information expressly challenging the validity of the related '744 patent, and thus constitute material prior art.

54. None of the persons involved in the prosecution of the '629 patent, including Mr. McAuley and Mr. Stone, disclosed to the PTO Microsoft's Invalidity Contentions, First Amended Invalidity Contentions, or Claim Construction Brief. Further, none of the persons involved in the prosecution of the '629 patent, including Mr. McAuley and Mr. Stone, disclosed to the PTO the existence of Microsoft's Invalidity Contentions, First Amended Invalidity Contentions, or Claim Construction Brief or the substance of the invalidity arguments set forth therein.

55. Only some prior art references relied on during the Microsoft litigation were selectively disclosed to the PTO during prosecution of the '629 patent.

56. The selective disclosure to the PTO of information arising out of the Microsoft litigation demonstrates that Mr. McAuley and Mr. Stone were aware of the Microsoft litigation and the existence of material information regarding the patentability of the '629 patent arising out of that litigation.

57. This selective disclosure to the PTO also demonstrates that Mr. McAuley and Mr. Stone made a deliberate decision to withhold material information from the PTO, and thus demonstrates an intent to deceive.

58. The withholding of information material to patentability with intent to deceive constitutes inequitable conduct, which renders the '629 patent unenforceable.

C. Prior Art References Produced by Microsoft

59. In an Information Disclosure Statement, the '629 patent applicants disclosed to the PTO 24 of the 61 prior art references (or excerpts thereof) that were identified by Microsoft in its Invalidity Contentions and First Amended Invalidity Contentions as anticipating and/or rendering obvious certain claims of the '744 patent.

60. On information and belief, none of the persons involved in the prosecution of the '629 patent, including Mr. McAuley and Mr. Stone, disclosed to the PTO any references

identified in Microsoft's Invalidity Contentions or First Amended Invalidity Contentions that pertain to the prior art Borland Delphi technology. Specifically, none of the persons involved in the prosecution of the '629 patent disclosed to the PTO: (i) *Borland Delphi 3 for Windows 95 & Windows NT, User's Guide*, Borland International, Inc. (1997); (ii) *Borland's Official No-Nonsense Guide to Delphi 2*, Sams Publishing (1996); (iii) Osier et al., *Teach Yourself Delphi 3 in 14 Days*, Sams Publishing (1997); (iv) Reisdorph, *Sams Teach Yourself Borland Delphi 4 in 21 Days*, Sams Publishing (1998); (v) Swan, *Delphi 4 Bible*, IDG Books Worldwide, Inc., Tom Swan (1998); (vi) Teixeira et al., *Borland Delphi 4 Developer's Guide*, Sams Publishing (1998).

61. On information and belief, none of the persons involved in the prosecution of the '629 patent, including Mr. McAuley and Mr. Stone, disclosed to the PTO any references identified in Microsoft's Invalidity Contentions or First Amended Invalidity Contentions that pertain to the prior art Microsoft Visual J++ technology. Specifically, none of the persons involved in the prosecution of the '629 patent disclosed to the PTO: (i) Doss, *DCOM Networking with Visual J++ 6.0*, Wordware Publishing, Inc. (1999); (ii) Morgan et al., *Visual J++ Unleashed*, Sams.net Publishing (1997); (iii) Mulloy, *Using Visual J++ 6*, Que Corporation (1998); (iv) Wood, *Visual J++ 6 Secrets*, IDG Books Worldwide, Inc. (1998).

62. On information and belief, none of the persons involved in the prosecution of the '629 patent, including Mr. McAuley and Mr. Stone, disclosed to the PTO any references identified in Microsoft's Invalidity Contentions or First Amended Invalidity Contentions that pertain to the prior art ASP technology. Specifically, none of the persons involved in the prosecution of the '629 patent disclosed to the PTO: (i) Fedorchek et al., *ASP: Active Server Pages*, IDG Books Worldwide, Inc. (1997); (ii) Fedorov et al., *ASP 2.0 Programmer's Reference*, Wrox Press (1998).

63. On information and belief, none of the persons involved in the prosecution of the '629 patent, including Mr. McAuley and Mr. Stone, disclosed to the PTO any references identified in Microsoft's Invalidity Contentions or First Amended Invalidity Contentions that pertain to the prior art Lotus Notes and Domino 4.5 technology. Specifically, none of the persons involved in the prosecution of the '629 patent disclosed to the PTO: (i) Forlini et al., *Lotus Notes*

and Domino 4.5 Professional Reference, New Riders Publishing (1997); (ii) Krantz, *Building Intranets with Lotus Notes & Domino*, Maximum Press (1997).

64. On information and belief, none of the persons involved in the prosecution of the '629 patent, including Mr. McAuley and Mr. Stone, disclosed to the PTO any references identified in Microsoft's Invalidity Contentions or First Amended Invalidity Contentions that pertain to the prior art Paradox 7 technology. Specifically, none of the persons involved in the prosecution of the '629 patent disclosed to the PTO: (i) Karim et al., *Paradox 7 Projects for Windows 95*, The Benjamin/Cummings Publishing Company, Inc. (1997); (ii) Weingarten et al., *Paradox 7 for Windows 95 Illustrated Brief Edition*, CTI (1997).

65. Just as was the case with the 24 prior art references from the Microsoft litigation that the applicants did disclose to the PTO, the narrative and claim charts submitted with Microsoft's Invalidity Contentions and First Amended Invalidity contentions demonstrate how the undisclosed Borland Delphi, Microsoft Visual J++, ASP, Lotus Notes and Domino 4.5, and Paradox 7 prior art references listed in Paragraphs 60-64 above anticipate and/or render obvious the asserted claims in the Microsoft litigation. Therefore, the undisclosed Borland Delphi, Microsoft Visual J++, ASP, Lotus Notes and Domino 4.5, and Paradox 7 prior art references listed in Paragraphs 60-64 above constitute material prior art.

66. The selective disclosure to the PTO of prior art references identified by Microsoft during the Microsoft litigation demonstrates that Mr. McAuley and Mr. Stone were aware of the Microsoft litigation and the existence of material information regarding the patentability of the '629 patent arising out of that litigation.

67. This selective disclosure to the PTO also demonstrates that Mr. McAuley and Mr. Stone made a deliberate decision to withhold material information from the PTO, and thus demonstrates an intent to deceive.

68. The withholding of information material to patentability with intent to deceive constitutes inequitable conduct, which renders the '629 patent unenforceable.

PRAYER FOR RELIEF

WHEREFORE, Samsung prays for the following relief:

A. A declaration that Samsung has not infringed and is not infringing, either directly or indirectly, or either literally or under the doctrine of equivalents, any claim of the '744 or '629 patent;

B. A declaration that each claim of the '744 and '629 patents is invalid;

C. A declaration that each claim of the '744 and '629 patents is unenforceable.

D. An order that Defendant and each of its officers, employees, agents, alter egos, attorneys, and any persons in active concert or participation with them are restrained and enjoined from further prosecuting or instituting any action against Samsung claiming that either the '744 or '629 patent is valid, enforceable, or infringed, or from representing that Samsung's products or services infringe the '744 patent or the '629 patent;

E. A declaration that this case is exceptional under 35 U.S.C. § 285 and awarding Samsung its attorneys' fees and costs in connection with this case;

F. Such other and further relief as the Court deems just and proper.

DEMAND FOR JURY TRIAL

Samsung demands a trial by jury on all issues so triable.

Dated: January 12, 2011

WEIL, GOTSHAL & MANGES LLP

By: Steven S. Cherensky

MATTHEW D. POWERS (Bar No. 104795)
matthew.powers@weil.com
STEVEN S. CHERENSKY (Bar No.168275)
steven.cherensky@weil.com
WEIL, GOTSHAL & MANGES LLP
Silicon Valley Office
201 Redwood Shores Parkway
Redwood Shores, CA 94065
Telephone: (650) 802-3000
Facsimile: (650) 802-3100

TIMOTHY DEMASI
tim.demasi@weil.com
STEVEN KALOGERAS
steven.kalogeras@weil.com
JULIAN MOORE
julian.moore@weil.com
WEIL, GOTSHAL & MANGES LLP
New York Office
767 5th Avenue
New York, NY 10153
Telephone: (212) 310-8000
Facsimile: (212) 310-8007

Attorneys for Plaintiffs
SAMSUNG ELECTRONICS CO., LTD.
SAMSUNG ELECTRONICS AMERICA, INC.

EXHIBIT A

US006826744B1

(12) **United States Patent**
McAuley

(10) **Patent No.: US 6,826,744 B1**
(45) **Date of Patent: Nov. 30, 2004**

(54) **SYSTEM AND METHOD FOR GENERATING WEB SITES IN AN ARBITRARY OBJECT FRAMEWORK**

(75) Inventor: **Aubrey McAuley**, Austin, TX (US)

(73) Assignee: **Vertical Computer Systems, Inc.**, Austin, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/410,334**

(22) Filed: **Oct. 1, 1999**

(51) **Int. Cl.**[7] **G06F 9/45**
(52) **U.S. Cl.** **717/108**
(58) **Field of Search** ..... 717/108; 707/103 R; 715/522

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,544,302	A *	8/1996	Nguyen	345/837
5,555,365	A *	9/1996	Selby et al.	345/765
5,894,554	A	4/1999	Lowery et al.	707/104.1
5,903,894	A *	5/1999	Reneris	707/100
5,930,512	A *	7/1999	Boden et al.	717/102
5,956,736	A *	9/1999	Hanson et al.	345/760
6,026,433	A *	2/2000	D'Arlach et al.	707/10
6,028,998	A *	2/2000	Gloudeman et al.	717/108
6,052,670	A *	4/2000	Johnson	705/27
6,199,082	B1 *	3/2001	Ferrel et al.	715/522
6,219,680	B1 *	4/2001	Bernardo et al.	707/501.1
6,226,648	B1 *	5/2001	Appleman et al.	707/102
6,247,032	B1 *	6/2001	Bernardo et al.	345/733
6,253,282	B1 *	6/2001	Gish	709/203
6,308,188	B1 *	10/2001	Bernardo et al.	707/501.1

OTHER PUBLICATIONS

Lewandowski, Framework for Component–Based Client/Server Computing, Mar. 1998, ACM, pp. 3–27.*

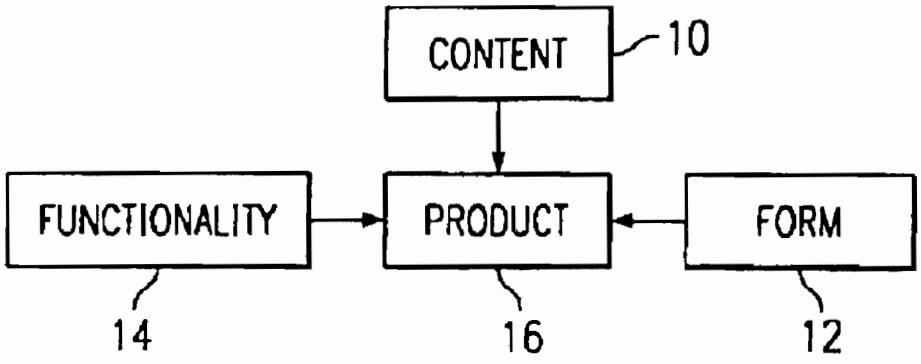
* cited by examiner

Primary Examiner—John Chavis
(74) *Attorney, Agent, or Firm*—Brown Raysman Millstein Felder & Steiner LLP

(57) **ABSTRACT**

A system and method for generating computer applications in an arbitrary object framework. The method separates content, form, and function of the computer application so that each may be accessed or modified separately. The method includes creating arbitrary objects, managing the arbitrary objects throughout their life cycle in an object library, and deploying the arbitrary objects in a design framework for use in complex computer applications.

53 Claims, 2 Drawing Sheets



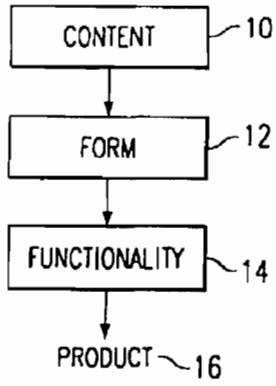


FIG. 1
(PRIOR ART)

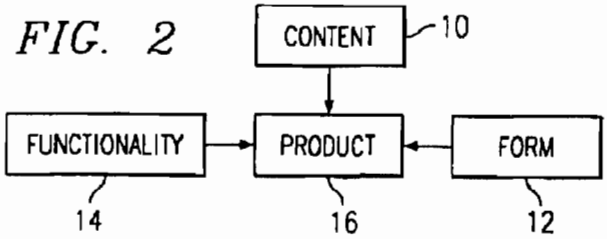


FIG. 2

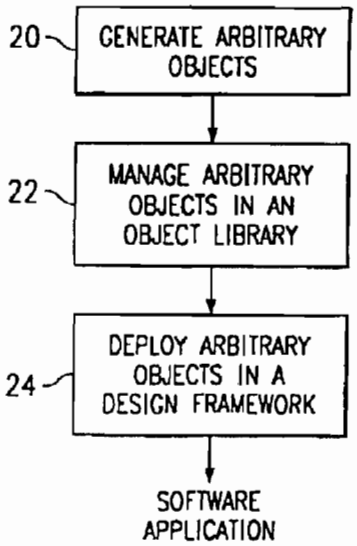


FIG. 3

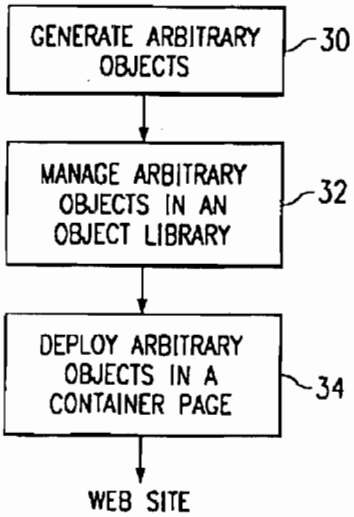


FIG. 4

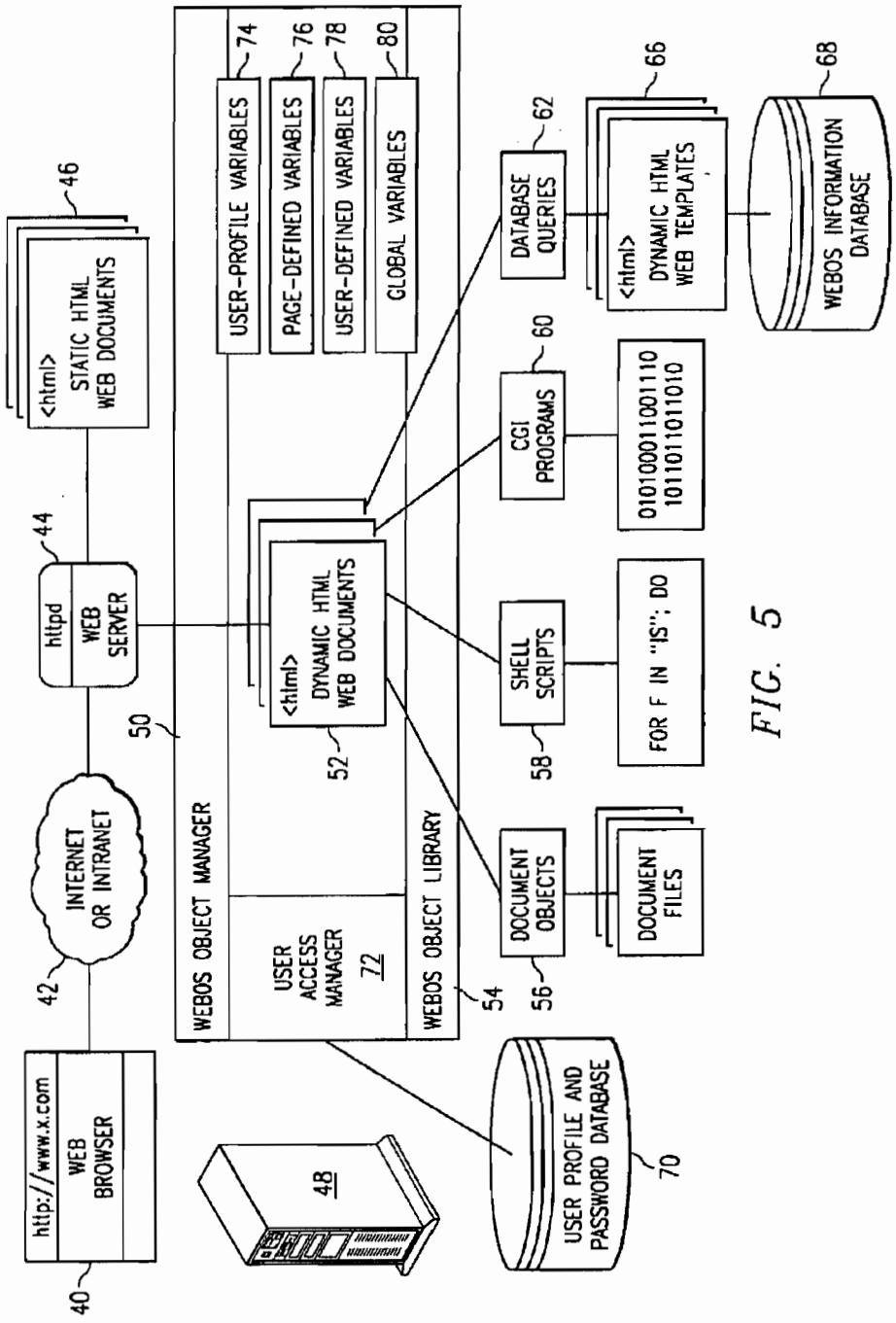

http://www.x.com
WEB BROWSER
40
INTERNET OR INTRANET
42
httpd
WEB SERVER
44
<html>
STATIC HTML WEB DOCUMENTS
46
48
WEBOS OBJECT MANAGER
50
<html>
DYNAMIC HTML WEB DOCUMENTS
52
USER ACCESS MANAGER
72
WEBOS OBJECT LIBRARY
54
USER-PROFILE VARIABLES
74
PAGE-DEFINED VARIABLES
76
USER-DEFINED VARIABLES
78
GLOBAL VARIABLES
80
DOCUMENT OBJECTS
56
DOCUMENT FILES
SHELL SCRIPTS
58
FOR F IN "IS"; DO
CGI PROGRAMS
60
01010001100111010111011011010
DATABASE QUERIES
62
<html>
DYNAMIC HTML WEB TEMPLATES
66
WEBOS INFORMATION DATABASE
68
USER PROFILE AND PASSWORD DATABASE
70

FIG. 5

SYSTEM AND METHOD FOR GENERATING WEB SITES IN AN ARBITRARY OBJECT FRAMEWORK

TECHNICAL FIELD OF THE INVENTION

This invention relates generally to systems and methods for generating software applications in an arbitrary object framework, and more specifically to systems and methods for generating web sites in an arbitrary object framework.

BACKGROUND OF THE INVENTION

Three processes used to create complex software applications such as web sites are form, function, and content. Form includes graphic designs, user interfaces, and graphical representations created by a designer or a group of designers. Function includes logical functionality, which can be software code created by a programmer or group of programmers. Form includes informative content. Informative content can include written, recorded, or illustrated documentation, such as photographs, illustrations, product marketing material, and news articles. Content can be created by writers, photographers, artists, reporters, or editors.

Currently, typical workflows dictate a serial approach to integrating the form, function, and content to create complex software applications such as a web site. The serial approach is illustrated in FIG. **1**. In FIG. **1**, content **10** for a complex software application can be chosen or created. Form **12** for the presentation of content **10** can then be created. Functionality **14** can then be generated using code to create the complex software application (product **16**) with the desired information (content **10**) and style (form **12**). Using the method illustrated in FIG. **1**, every final component of the complex software application must be manipulated by a programmer before it is ready to be used. The exact workflow may vary from industry to industry or business to business, but the basic restrictions are generally the same.

A traditional approach such as that illustrated in FIG. **1**, may create unwanted bottlenecks in the production process. Each upstream revision, such as a change of content **10** or design **12**, forces a repetition of the entire process. As an example, consider a web site for a large newspaper. The web site may have a function that can include a file into the web site. The marketing department may decide to change the appearance of the header on the web site depending on the browser of a user. In this case, a programmer may need to invoke an external script or embed some specific logic within the web site. Unfortunately, if there is a large web site with thousands of pages of information stored on a server, the programmer may have to change every one of the thousands of pages. Therefore, a small change by the marketing department can cause a large burden on the programming department.

Prior art solutions have succeeded in partially separating some of these functions. Notably, content management databases and digital repositories provide a means of separating content from form and function. Likewise, sophisticated software development teams frequently employ internal code structuring techniques that can help to minimize dependencies between interface designs and the functions they access. However, content management tools typically fail to address form/function issues. Therefore, there can still be production slow-downs due to changes in form that require a subsequent change in functionality.

SUMMARY OF THE INVENTION

Therefore a need exists for a method of generating complex software applications that reduces or eliminates

production delays and the workload for programmers due to changes in content and/or form. This method should separate form, content and function so that each area can be independently changed.

The present invention provides a system and method for generating software applications that substantially eliminates or reduces disadvantages and problems associated with previously developed systems and methods used for generation of software applications. More specifically, the present invention provides a method for generating software applications in an arbitrary object framework. The method of the present invention separates content, form, and function of the computer application so that each may be accessed or modified independently. The method of this invention includes creating arbitrary objects, managing the arbitrary objects throughout their life cycle, and deploying the arbitrary objects in a design framework for use in complex computer applications.

The present invention provides an important technical advantage in that content, form, and function are separated from each other in the generation of the software application. Therefore, changes in design or content do not require the intervention of a programmer. This advantage decreases the time needed to change various aspects of the software application. Consequently, cost is reduced and versatility is increased.

The present invention provides another technical advantage in that users are not required to use a proprietary language to encode. These arbitrary objects may include encapsulated legacy data, legacy systems and custom programming logic from essentially any source in which they may reside. Any language supported by the host system, or any language that can be interfaced to by the host system, can be used to generate an object within the application.

The present invention provides yet another technical advantage in that it can provide a single point of administrative authority that can reduce security risks. For instance, a large team of programmers can work on developing a large group of arbitrary objects within the object library. If one object has a security hole, an administrator can enter the object library and disable that arbitrary object.

Still another technical advantage of the present invention is that it enables syndication of the software application. As noted above, functionality is separate from form and content. Consequently, a user can easily introduce a new look for the application or syndicate the content and functionality of the application to another group without having to recode all of the objects needed to access content.

Another technical advantage of the present invention is that it allows for personalization and profiling. With personalization, the web presentation is tailored to the specific needs of the web user based on the user's past history. Profiling also enables tailoring a web site or presentation. Profiling is dependent on environmental variables such as browser type or IP address.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the present invention and the advantages thereof may be acquired by referring to the following description, taken in conjunction with the accompanying drawings in which like reference numbers indicate like features and wherein:

FIG. **1** illustrates a prior art workflow diagram for generating a software product;

FIG. **2** is a hierarchical workflow diagram for one embodiment of the present invention;

FIG. **3** is a flow diagram for one embodiment of the present invention;

FIG. **4** is a flow diagram for the embodiment illustrated in FIG. **4**.

FIG. **5** is a diagram illustrating the components of one embodiment of the present invention used to generate web sites; and

DETAILED DESCRIPTION OF THE INVENTION

Preferred embodiments of the present invention are illustrated in the FIGURES, like numerals being used to refer to like and corresponding parts of various drawings.

The present invention provides a system and method for using a hierarchical, arbitrary object framework for generating software applications. The method separates content, form, and function of the software application so that each can be accessed or modified independently. The method of this invention includes creating arbitrary objects, managing the arbitrary objects in an object library, and deploying the arbitrary objects in a design framework for use in computer applications.

FIG. **2** is a hierarchical workflow diagram for the present invention. Product **6** includes three contributing groups: content **10**, form **12**, and functionality **14**. Content **10** can include written, recorded, or illustrated collateral such as documentation, photographic illustrations, product marketing material, and articles. Form **12** can include graphic designs such as user interfaces and graphical presentations. Function **14** can include the logical functionality of software code and scripts. The hierarchical framework separates content **10**, form **12**, and functionality **14** to generate product **16**. Product **16** may be a computer software application such as a web site. Since content **10**, design **12**, and functionality **14** are separate entities independent of each other, modification in one group does not require corresponding modifications in another group. Each group can contribute to product **16** directly.

FIG. **3** is a flow diagram of one embodiment of the present invention. At step **20**, arbitrary objects can be generated. Arbitrary objects may include any combination of application logic and data desired by a developer. Arbitrary objects can include text file pointers, binary file pointers, compiled executables, scripts, data base queries, shell commands, remote procedure calls, global variables, and local variables. The arbitrary object framework allows arbitrary objects to be referenced in a consistent manner regardless of the type. Also, the arbitrary object framework allows local arbitrary objects to either override global parent arbitrary objects or inherit capabilities and data from the global parent, regardless of the type of the local arbitrary object.

At step **22**, these arbitrary objects can be managed in an object library. The life cycle of these objects may be managed in a consistent manner using revision tracking, roll back, and sign off. At step **24**, objects can be deployed from the object library into a design framework to create the software application. Because the object pointers are not tied in any way to the functionality of the object, an object of one type can be easily replaced with another object of another type. This eliminates a common problem in content management systems of the inability to preview content within its appropriate location on the site or within the system. Normally, a special system made for the purpose of previewing a piece of content would have to be hard-coded to view the current approved live content for all other pieces except the piece in question. This multiplies the design problem, because changes in the design in the main site change all previous templates. In the method of the present invention, since all that exists within the framework is an arbitrary object, the arbitrary object can be swapped for another object that pulls the current piece content in question.

Using one embodiment of this invention, for example, the Features or Editorials page of a newspaper can be dynamically replaced. The present invention can execute all the normal objects that can be placed on the page to show the content as it would appear, and then take the one piece in question and replace it with a second object to be examined. Objects may be deployed globally across an entire system or locally within a specific area or sub-areas of a system.

FIG. **4** represents a flow diagram of another embodiment of the present invention. At step **30**, arbitrary objects can be generated. At step **32**, the arbitrary objects can be managed in an object library. Arbitrary objects can be deployed in a container page at step **34** to generate a web site.

Arbitrary objects may include any combination of application logic and data desired by a developer. Arbitrary objects can include text file pointers, binary file pointers, compiled executable scripts, database queries, shell commands, remote call procedures, global variables and local variables. Arbitrary objects may also include cached data queries and executables. The arbitrary object framework allows arbitrary objects to be referenced in a consistent manner regardless of the type of object. Also, the arbitrary object framework allows local arbitrary objects to either override global parent arbitrary objects or inherit capabilities and data from the global parent arbitrary object.

Arbitrary objects can execute any function that can be run or understood by the host computer system so that any underlying functionality of the operating system used by the host system can be defined as an object within the arbitrary framework. Legacy data, document objects, CPI programs, and database queries can all be encapsulated as objects within the arbitrary framework. The arbitrary object can be accessed by an arbitrary object name. Arbitrary objects are not tied to their functionality. One arbitrary object can be easily replaced with another arbitrary object of another type.

Arbitrary objects can be managed in an object library. The life cycle of the arbitrary objects may be managed in a consistent manner using revision tracking, roll-back, and sign-off. The object library can include separate specialized object libraries that can be administered separately by different developers in each area. For instance, for a web site used to generate a newspaper, there may be an advertising object library that is physically distinguished from other object libraries, such as an object library for sports or an object library for news. Therefore, queries for advertising can be created without impacting any other area of the web site.

Arbitrary objects can be deployed from the object library into a container page to generate the web site. The container page is a truly dynamic page. Unlike prior art methods, where a static copy of information is often pushed over a firewall to a live web site, the present invention incorporates object caching. An arbitrary object can be cached, rather than caching an entire page. When the arbitrary object is cached, certain elements of the arbitrary object can be specified as dynamic elements while others can be specified as static elements. Therefore, a web site can contain multiple dynamic web pages wherein objects used to construct the form, function, and content of the web page can contain dynamic elements and static elements. This provides flex-

ibility for what needs to be computed or processed at the time that someone, such as a web user, accesses the web page.

FIG. **5** shows the components of one embodiment of the present invention used to generate web sites. A user with web browser **40** can connect to web server **44** through internet or intranet **42**. Web server **44** can access static HTML web documents **46** as well as dynamic HTML documents **52**. Dynamic HTML web documents **52** can be created using Web OS Object Manager **50**. Dynamic HTML Web document **52** can include document objects **56**, shell scripts **58**, CGI programs **60**, and database queries **62**. Document objects **56**, shell scripts **58**, CGI programs **60**, and database queries **62** can be stored in WebOS object library **54**. Database queries **62** can result from extracting information from WebOS Information Database **68** and inputting the information into Dynamic HTML Web Template **66**.

User Profile and Password Database **70** can provide web sites or systems with a means to take advantage of customer profiles to look at customer preferences or history, and dynamically replace a website object with another object that contains content information matching the user profile or preferences. Thus, the web site or system can dynamically allocate the correct content for a customer. This is important in commerce applications. A customer's buying history can be examined for trend items and the customer presented products that match his or her profile. Present personalization systems are written purely in custom code and require an inordinately large amount of time to construct the custom applications necessary to interpret the preferences of an individual user.

The method of present invention can perform object caching. This means that an object can be cached instead of caching an entire page. Object caching permits specifying elements of an object to be dynamic and elements of the object to be static. A system user can thus have the flexibility of specifying what needs to be computed or processed at the time a user accesses the system versus trying to anticipate and calculate in advance and cache and post the object over to a server.

Many functions are stored within an object library on an arbitrary object framework such that those functions can be accessed by name arbitrarily. This is in contrast to a traditional model where the function must be explicitly invoked with all its parameters included. Objects may execute any function that can be run or understood by the host computer system so that any underlying functionality of the host's operating system can be defined as an object within the framework of the method of the present invention. The object library can contain legacy data, document objects, CTI programs, and/or database queries, that can all be encapsulated as objects within a framework and accessed from within a design. All that is needed is the name of the function in order to access the function.

Objects can be controlled to perform functions based on a profile of an individual and environmental variables, such as the type of browser, the country of the individual or the individual's IP address. A specific competitor may be blocked from seeing certain objects on a web page created using the method of the present invention.

A critical distinction between the present invention and previous object oriented development systems is the need to know how a function can be called and what to expect it to return, rather than just knowing the function's name. This means that typically the system administrator calls the name of an object and passes parameters to the object. Any and all variable information or environmental information can be available to every object. The environment space can be available to all objects executed and an object can arbitrarily take advantage of any of the environmental information, depending on the design of the object.

Different areas of a web site can be administered separately by different developers in each of these areas. An advertising object library can be physically distinguished from other object libraries, such as those for sports and news. An advertising programmer can create new queries for the advertising section of a site without having to worry about affecting other areas of the site.

The present invention allows different object types to be interchangeable. The object name is essentially just another variable in the environment. Also different variables can also be interchangeable. The object framework can be designed such that objects and variables can be kept in the same name space, every object can have access to all the environmental settings, and every object pointer can potentially be another name in the name space.

Object caching, rather than page caching can be implemented with the present invention. These objects can be stored in an object library. An object in the object library can be a file, a global variable, an executable script, a database query, a cached executable or a cached database query. This means that the results of a query can be stored in a static file using the object name as long as the static file has not expired. This is important if the query is a lengthy query.

A technical advantage of the present invention is that it allows for syndication. Syndication enables the content and function of a particular web site to be syndicated to another web site or web presentation. For instance, if a company would like to roll out a new look or syndicate its content and functionality to another business, this can be easily accomplished using the present invention. Since there is no application code resident in a web page itself, the same data can be repackaged in a number of different ways across multiple sites. There is no need to recode the design elements or design pages on the web site or recode any functions that are needed to access the content of the website. The present invention enables electronic store fronts to sell from a single source with a unique interface design. Also, newspaper chains can distribute international and national content from a single source and add local content themselves.

Another technical advantage of the present invention is that it allows for a single point of control when developing a web site. Therefore, if a large team of developers are working on a site, and multiple persons are contributing arbitrary objects to the overall arbitrary framework, then if one of the arbitrary objects has a security hole in it, the arbitrary object can be easily accessed in the object library and disabled. This security feature can immediately shut down that function across the entire web site and patch the security hole.

The present invention provides still another technical advantage in that it allows for personalization. Personalization enables companies that want to take advantage of a customer profile to look at the customer's preferences or histories and deploy information to the web site specific to the customer.

Another technical advantage of the present invention allows for profiling. Profiling enables control over the arbitrary objects presented in a web site based on a profile of the individual accessing the web site. Profiling entails determining different environmental variables such as the type of browser hitting the site, the country of the individual access-

ing the site, and/or the individual's IP address. This can enable a company to present specific information to the individual based on the individual's environmental variables.

Although the present invention has been described in detail herein with reference to the illustrative embodiments, it should be understood that the description is by way of example only and is not to be construed in a limiting sense. It is to be further understood, therefore, that numerous changes in the details of the embodiments of this invention and additional embodiments of this invention will be apparent to, and may be made by, persons of ordinary skill in the art having reference to this description. It is contemplated that all such changes and additional embodiments are within the spirit and true scope of this invention as claimed below.

What is claimed is:

1. A method for generating a computer application on a host system in an arbitrary object framework that separates a content of said computer application, a form of said computer application and a functionality of said computer application, said method comprising:

creating arbitrary objects with corresponding arbitrary names of various object types for generating said content of said computer application, said form of said computer application, and said functionality of said computer application;

managing said arbitrary objects in an object library; and

deploying said arbitrary objects from said object library into a design framework to create said computer application.

2. The method of claim 1, wherein said computer application is a web site.

3. The method of claim 1, wherein said various object types comprise text file pointers.

4. The method of claim 1, wherein said various object types comprise binary file pointers.

5. The method of claim 1, wherein said various object types comprise compiled executables.

6. The method of claim 1, wherein said various object types comprise shell commands.

7. The method of claim 1, wherein said various object types comprise remote procedure calls.

8. The method of claim 1, wherein said various object types comprise global variables.

9. The method of claim 1, wherein said various object types comprise cached executables.

10. The method of claim 1, wherein said various object types comprise cached database queries.

11. The method of claim 1, wherein said various object types comprise local variables.

12. The method of claim 1, wherein said various object types comprise local objects and global parent objects.

13. The method of claim 12, wherein said local objects can override said global parent objects.

14. The method of claim 12, wherein said local objects inherit data from said global parent objects.

15. The method of claim 12, wherein said local objects inherit capabilities from said global parent objects.

16. The method of claim 1, further comprising deploying arbitrary objects globally.

17. The method of claim 1, further comprising deploying arbitrary objects locally.

18. The method of claim 1, wherein the step of managing said arbitrary objects further comprises using revision tracking.

19. The method of claim 1, wherein the step of managing said arbitrary objects further comprises using rollback.

20. The method of claim 1, wherein the step managing further comprises using signoff.

21. The method of claim 1, wherein said arbitrary objects can be accessed and deployed into said design framework using said corresponding arbitrary names.

22. The method of claim 1, further comprising swapping an arbitrary object of one type with an arbitrary object of another type.

23. The method of claim 1, further comprising caching objects.

24. The method of claim 23, wherein the step of caching objects further comprises specifying some elements of an arbitrary object to be dynamic elements and specifying some elements of said arbitrary object to be static elements.

25. The method of claim 1, further comprising generating arbitrary objects in a programming language that is compatible or supported by said host system.

26. A method for generating a web site on a host system in an arbitrary object framework that separates a content of said web site, a form of said web site, and a functionality of said web site, said method comprising:

creating arbitrary objects with corresponding arbitrary names of various object types for generating said content of said web site, said form of said web site, and said functionality of said web site; managing said arbitrary objects in an object library; and

deploying said arbitrary objects from said object library to a container page to create said web site.

27. The method of claim 26, wherein said various object types comprise text file pointers.

28. The method of claim 26, wherein said various object types comprise binary file pointers.

29. The method of claim 26, wherein said various object types comprise compiled executables.

30. The method of claim 26, wherein said various object types comprise shell commands.

31. The method of claim 26, wherein said various object types comprise remote procedure calls.

32. The method of claim 26, wherein said various object types comprise global variables.

33. The method of claim 26, wherein said various object types comprise local variables.

34. The method of claim 26, wherein said various object types comprise local objects and global parent objects.

35. The method of claim 34, wherein said local objects can override said global parent objects.

36. The method of claim 34, wherein said local objects inherit data from said global parent objects.

37. The method of claim 34, wherein said local objects inherit capabilities from said global parent objects.

38. The method of claim 26, further comprising deploying arbitrary objects globally.

39. The method of claim 26, further comprising deploying arbitrary objects locally.

40. The method of claim 26, wherein the step of managing said arbitrary objects further comprises using revision tracking.

41. The method of claim 26, wherein the step of managing said arbitrary objects further comprises using rollback.

42. The method of claim 26, wherein the step managing said arbitrary objects further comprises using signoff,.

43. The method of claim 26, wherein said arbitrary objects can be accessed and deployed into said container page using said corresponding arbitrary names.

44. The method of claim 26, further comprising swapping an arbitrary object of one type with an arbitrary object of another type.

45. The method of claim **26**, further comprising caching objects.

46. The method of claim **45**, wherein the step of caching objects further comprises specifying some elements of an arbitrary object to be dynamic elements and specifying some elements of said arbitrary object to be static elements.

47. The method of claim **26**, further comprising generating arbitrary objects in a programming language that is compatible or supported by said host system.

48. The method of claim **26**, wherein said various object types comprise cached executable.

49. The method of claim **25**, wherein said various object types comprise cached database queries.

50. The method of claim **26**, further comprising profiling of a user accessing said web site.

51. The method of claim **26**, further comprising personalization of said web site for a user accessing said web site.

52. The method of claim **26**, wherein said container page comprises arbitrary objects with both dynamic and static elements.

53. The method of claim **26**, wherein said content of said web site and said function of said web site can be syndicated.

* * * * *

EXHIBIT B

US007716629B2

(12) **United States Patent**
McAuley

(10) **Patent No.: US 7,716,629 B2**
(45) **Date of Patent: *May 11, 2010**

(54) **SYSTEM AND METHOD FOR GENERATING WEB SITES IN AN ARBITRARY OBJECT FRAMEWORK**

(75) Inventor: **Aubrey McAuley**, Austin, TX (US)

(73) Assignee: **Vertical Computer Systems, Inc.**, Fort Worth, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 613 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/999,911**

(22) Filed: **Nov. 29, 2004**

(65) **Prior Publication Data**

US 2005/0154486 A1 Jul. 14, 2005

Related U.S. Application Data

(63) Continuation of application No. 09/410,334, filed on Oct. 1, 1999, now Pat. No. 6,826,744.

(51) **Int. Cl.**
G06F 9/45 (2006.01)

(52) **U.S. Cl.** **717/100**

(58) **Field of Classification Search** 717/106, 717/100, 149

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,544,302	A *	8/1996	Nguyen	715/837
5,555,365	A *	9/1996	Selby et al.	715/765
5,894,554	A *	4/1999	Lowery et al.	709/203
5,895,476	A	4/1999	Orr et al.	
5,903,894	A *	5/1999	Reneris	707/100
5,930,512	A *	7/1999	Boden et al.	717/102
5,956,736	A *	9/1999	Hanson et al.	715/513
6,026,433	A *	2/2000	D'Arlach et al.	709/217
6,028,998	A *	2/2000	Gloudeman et al.	717/108
6,052,670	A *	4/2000	Johnson	705/27
6,199,082	B1 *	3/2001	Ferrel et al.	715/522
6,219,680	B1 *	4/2001	Bernardo et al.	715/501.1
6,226,648	B1 *	5/2001	Appleman et al.	707/102
6,247,032	B1 *	6/2001	Bernardo et al.	715/530

(Continued)

FOREIGN PATENT DOCUMENTS

CA	2110970	6/1995

OTHER PUBLICATIONS

Adhesive Media, Inc., Dynamic Web Management Tools, WebOS/NewsFlash/SiteFlash, Assorted Documents, pp. MSVERT002 through MSVERT126.

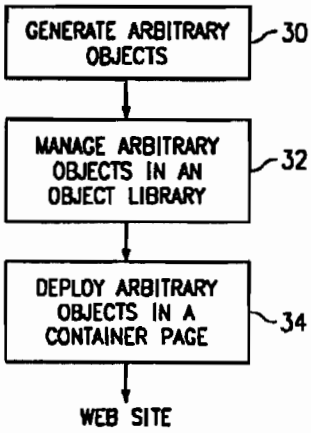
(Continued)

Primary Examiner—John Chavis
(74) *Attorney, Agent, or Firm*—Scheef & Stone, L.L.P.; Jack D. Stone, Jr.

(57) **ABSTRACT**

A method and system for generating a computer application is disclosed. The computer application is generated on a host system in an arbitrary object framework that separates a content of said computer application, a form of said computer application and a functionality of said computer application. Arbitrary objects are created with corresponding arbitrary names of various object types for generating said content of said computer application, said form of said computer application, and said functionality of said computer application. The arbitrary objects are managed in an object library. The arbitrary objects are deployed from said object library into a design framework to create said computer application.

32 Claims, 2 Drawing Sheets



U.S. PATENT DOCUMENTS

6,253,282	B1 *	6/2001	Gish	711/113
6,308,188	B1 *	10/2001	Bernardo et al.	715/530
6,553,563	B2	4/2003	Ambrose et al.	
6,574,635	B2	6/2003	Stauber et al.	
7,284,193	B1	10/2007	Lindhorst	
2002/0147805	A1 *	10/2002	Leshem et al.	709/223
2002/0161734	A1	10/2002	Stauber et al.	

OTHER PUBLICATIONS

Apple Computer, Inc., Inc., "The WebObjects Dynamic Elements Reference," http://developer.apple.com/ Documentation/ WebObjects/Reference/DynamicElements/DynamicElements.pdf, Jan. 10, 2006, WebObjects and the Enterprise Objects Framework, Chapt. 16.

Banick, et al.., "Using Microsoft FrontPage 98," Que Corporation (1998), Microsoft FrontPage, Chapt. 3.

Burke, "Web Databases with Cold Fusion 3," The McGraw-Hill Companies, Inc. (1998), Chapts. 1, 6, 9, and 15.

Buyens, "Running Microsoft FrontPage 98," Microsoft Press, Jim Buyens (1997), Microsoft FrontPage 98, Chapt. 4.

Darnell, et al., Using Macromedia Dreamweaver 1.2, Que (1998), Macromedia Dreamweaver, Chapts. 8 and 12.

Forta, "The ColdFusion 4.0 Web Application Construction Kit," Que (1998), ColdFusion 4.0, Chapts. 10 through 14.

Gray, "Web Publishing with Adobe PageMill 2," Ventana Communications Group, Inc., (1997), Adobe PageMill1, Chapts. 1 and 3.

Howell, "The Complete Idiot's Guide to Microsoft Visual InterDev," Que Corporation (1997), Microsoft Visual InterDev, Chapt. 4.

Jones, et al.,, "Cascading Style Sheets: A Primer," MIS:Press, Big Tent Media Labs, LLC (1998), Dynamic HTML and Cascading Style Sheets, Chapt. 4.

Martin, et al., "Object-Oriented Methods: A Foundation," P T R Prentice Hall, (1995), Object-Oriented Programming Art, Chapts. 1 through 3.

Meyer, "Object-Oriented Software Construction," 2nd Ed., Prentice Hall PTR, (1997), Object-Oriented Programming Art, Chapt. 2.

Pollizi, "Separating Form from Function: The StarView Experience," Astronomical Data Analysis Software and Systems III, ASP Conference Series, vol. 61, pp. 88-91 (1994).

Prague, et al., "Access 97 Bible," IDG Books Worldwide, Inc. (1997), Microsoft Access 97, Chapts. 1 through 9, (Part 1).

Prague, et al., "Access 97 Bible," IDG Books Worldwide, Inc. (1997), Microsoft Access 97, Chapts. 10 through18, (Part 2).

Prague, et al., "Access 97 Bible," IDG Books Worldwide, Inc. (1997), Microsoft Access 97, Chapts. 19 through 26, (Part 3).

Prague, et al., "Access 97 Bible," IDG Books Worldwide, Inc. (1997), Microsoft Access 97, Chapts. 27 through 33, and Appendices (Part 4).

Webster, "NetObjects Fusion Handbook," Hayden Books (1996), Webster, NetObjects Fusion, Chapts. 9 through 13.

* cited by examiner

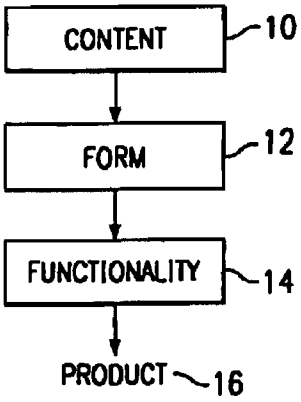


FIG. 1
(PRIOR ART)

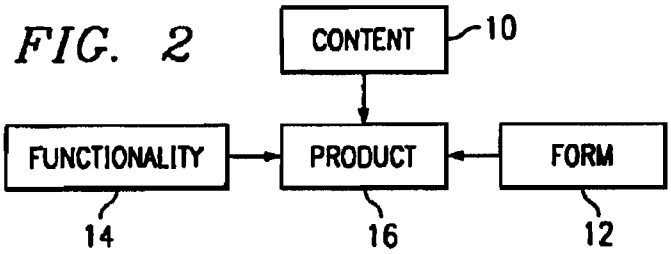


FIG. 2

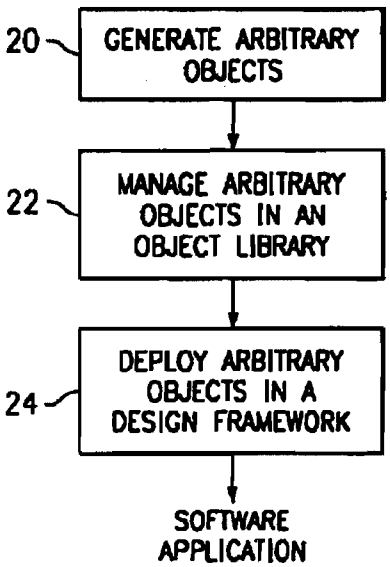


FIG. 3

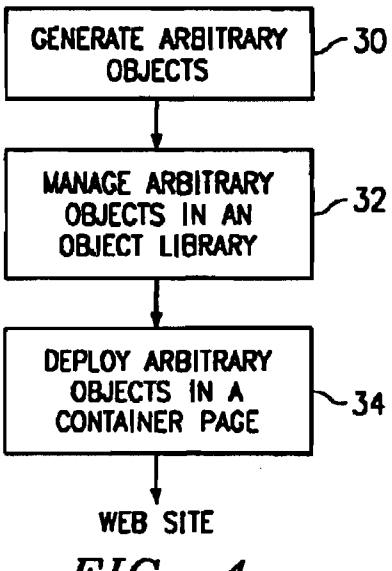


FIG. 4

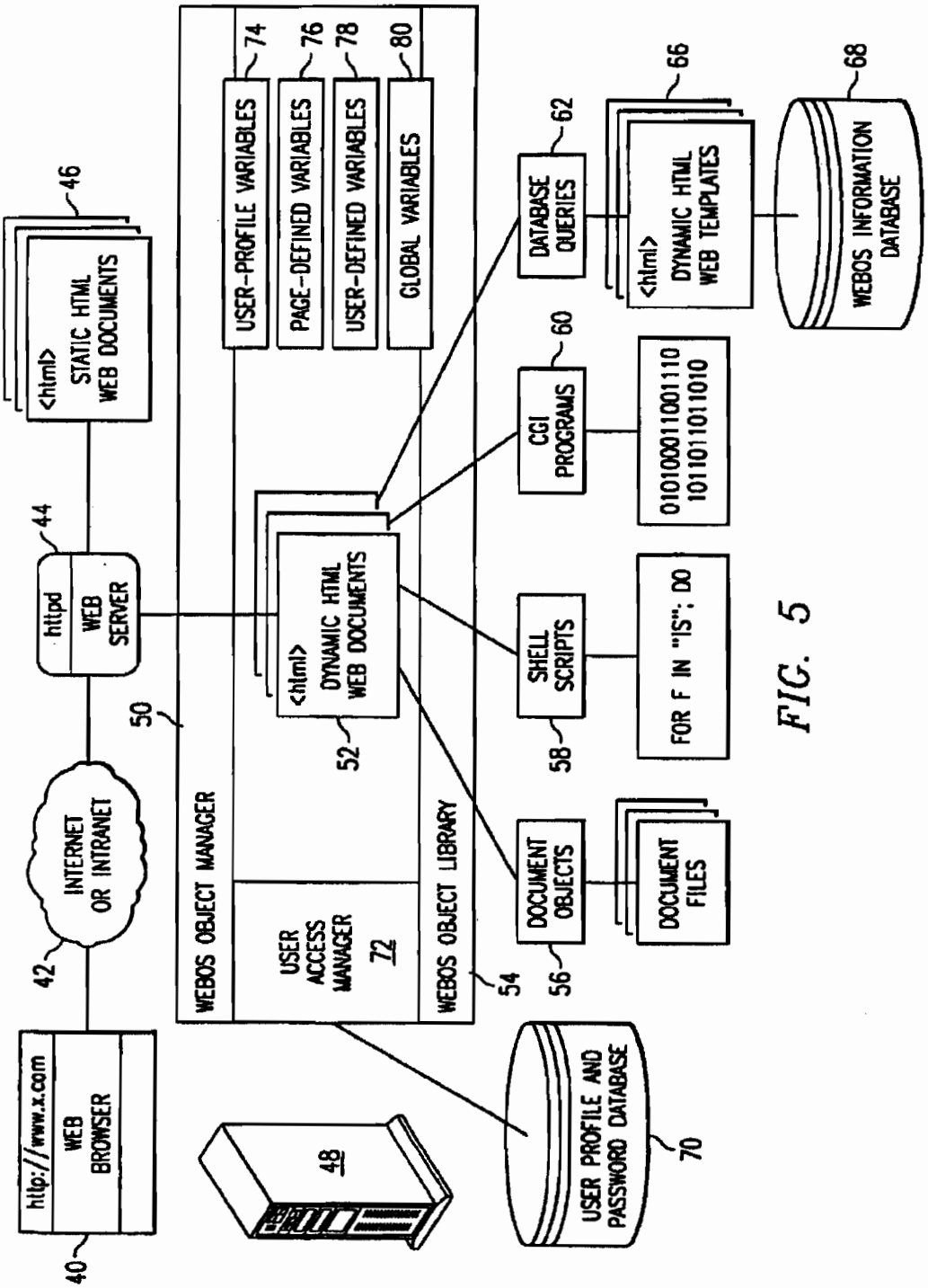

http://www.x.com
WEB BROWSER
40
INTERNET OR INTRANET
42
httpd WEB SERVER
44
<html> STATIC HTML WEB DOCUMENTS
46
48
WEBOS OBJECT MANAGER
50
USER ACCESS MANAGER 72
<html> DYNAMIC HTML WEB DOCUMENTS
52
WEBOS OBJECT LIBRARY
54
USER-PROFILE VARIABLES
74
PAGE-DEFINED VARIABLES
76
USER-DEFINED VARIABLES
78
GLOBAL VARIABLES
80
DOCUMENT OBJECTS
56
DOCUMENT FILES
SHELL SCRIPTS
58
FOR F IN "IS"; DO
CGI PROGRAMS
60
0101000111001110 1011011011010
DATABASE QUERIES
62
<html> DYNAMIC HTML WEB TEMPLATES
66
WEBOS INFORMATION DATABASE
68
USER PROFILE AND PASSWORD DATABASE
70

FIG. 5

SYSTEM AND METHOD FOR GENERATING WEB SITES IN AN ARBITRARY OBJECT FRAMEWORK

RELATED APPLICATIONS

This application is a continuation of U.S. application Ser. No. 09/410,334, filed Oct. 1, 1999, now U.S. Pat. No. 6,826,744.

TECHNICAL FIELD OF THE INVENTION

This invention relates generally to systems and methods for generating software applications in an arbitrary object framework, and more specifically to systems and methods for generating web sites in an arbitrary object framework.

BACKGROUND OF THE INVENTION

Three processes used to create complex software applications such as web sites are form, function, and content. Form includes graphic designs, user interfaces, and graphical representations created by a designer or a group of designers. Function includes logical functionality, which can be software code created by a programmer or group of programmers. Form includes informative content. Informative content can include written, recorded, or illustrated documentation, such as photographs, illustrations, product marketing material, and news articles. Content can be created by writers, photographers, artists, reporters, or editors.

Currently, typical workflows dictate a serial approach to integrating the form, function, and content to create complex software applications such as a web site. The serial approach is illustrated in FIG. **1**. In FIG. **1**, content **10** for a complex software application can be chosen or created. Form **12** for the presentation of content **10** can then be created. Functionality **14** can then be generated using code to create the complex software application (product **16**) with the desired information (content **10**) and style (form **12**). Using the method illustrated in FIG. **1**, every final component of the complex software application must be manipulated by a programmer before it is ready to be used. The exact workflow may vary from industry to industry or business to business, but the basic restrictions are generally the same.

A traditional approach such as that illustrated in FIG. **1**, may create unwanted bottlenecks in the production process. Each upstream revision, such as a change of content **10** or design **12**, forces a repetition of the entire process. As an example, consider a web site for a large newspaper. The web site may have a function that can include a file into the web site. The marketing department may decide to change the appearance of the header on the web site depending on the browser of a user. In this case, a programmer may need to invoke an external script or embed some specific logic within the web site. Unfortunately, if there is a large web site with thousands of pages of information stored on a server, the programmer may have to change every one of the thousands of pages. Therefore, a small change by the marketing department can cause a large burden on the programming department.

Prior art solutions have succeeded in partially separating some of these functions. Notably, content management databases and digital repositories provide a means of separating content from form and function. Likewise, sophisticated software development teams frequently employ internal code structuring techniques that can help to minimize dependencies between interface designs and the functions they access.

However, content management tools typically fail to address form/function issues. Therefore, there can still be production slow-downs due to changes in form that require a subsequent change in functionality.

SUMMARY OF THE INVENTION

Therefore a need exists for a method of generating complex software applications that reduces or eliminates production delays and the workload for programmers due to changes in content and/or form. This method should separate form, content and function so that each area can be independently changed.

The present invention provides a system and method for generating software applications that substantially eliminates or reduces disadvantages and problems associated with previously developed systems and methods used for generation of software applications. More specifically, the present invention provides a method for generating software applications in an arbitrary object framework. The method of the present invention separates content, form, and function of the computer application so that each may be accessed or modified independently. The method of this invention includes creating arbitrary objects, managing the arbitrary objects throughout their life cycle, and deploying the arbitrary objects in a design framework for use in complex computer applications.

The present invention provides an important technical advantage in that content, form, and function are separated from each other in the generation of the software application. Therefore, changes in design or content do not require the intervention of a programmer. This advantage decreases the time needed to change various aspects of the software application. Consequently, cost is reduced and versatility is increased.

The present invention provides another technical advantage in that users are not required to use a proprietary language to encode. These arbitrary objects may include encapsulated legacy data, legacy systems and custom programming logic from essentially any source in which they may reside. Any language supported by the host system, or any language that can be interfaced to by the host system, can be used to generate an object within the application.

The present invention provides yet another technical advantage in that it can provide a single point of administrative authority that can reduce security risks. For instance, a large team of programmers can work on developing a large group of arbitrary objects within the object library. If one object has a security hole, an administrator can enter the object library and disable that arbitrary object.

Still another technical advantage of the present invention is that it enables syndication of the software application. As noted above, functionality is separate from form and content. Consequently, a user can easily introduce a new look for the application or syndicate the content and functionality of the application to another group without having to recode all of the objects needed to access content.

Another technical advantage of the present invention is that it allows for personalization and profiling. With personalization, the web presentation is tailored to the specific needs of the web user based on the user's past history. Profiling also enables tailoring a web site or presentation. Profiling is dependent on environmental variables such as browser type or IP address.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the present invention and the advantages thereof may be acquired by referring to

the following description, taken in conjunction with the accompanying drawings in which like reference numbers indicate like features and wherein:

FIG. **1** illustrates a prior art workflow diagram for generating a software product;

FIG. **2** is a hierarchical workflow diagram for one embodiment of the present invention;

FIG. **3** is a flow diagram for one embodiment of the present invention;

FIG. **4** is a flow diagram for another embodiment of the present invention; and

FIG. **5** is a diagram illustrating the components of one environment of the present invention used to generate web sites.

DETAILED DESCRIPTION OF THE INVENTION

Preferred embodiments of the present invention are illustrated in the FIGUREs, like numerals being used to refer to like and corresponding parts of various drawings.

The present invention provides a system and method for using a hierarchical, arbitrary object framework for generating software applications. The method separates content, form, and function of the software application so that each can be accessed or modified independently. The method of this invention includes creating arbitrary objects, managing the arbitrary objects in an object library, and deploying the arbitrary objects in a design framework for use in computer applications.

FIG. **2** is a hierarchical workflow diagram for the present invention. Product **16** includes three contributing groups: content **10**, form **12**, and functionality **14**. Content **10** can include written, recorded, or illustrated collateral such as documentation, photographic illustrations, product marketing material, and articles. Form **12** can include graphic designs such as user interfaces and graphical presentations. Function **14** can include the logical functionality of software code and scripts. The hierarchical framework separates content **10**, form **12**, and functionality **14** to generate product **16**. Product **16** may be a computer software application such as a web site. Since content **10**, design **12**, and functionality **14** are separate entities independent of each other, modification in one group does not require corresponding modifications in another group. Each group can contribute to product **16** directly.

FIG. **3** is a flow diagram of one embodiment of the present invention. At step **20**, arbitrary objects can be generated. Arbitrary objects may include any combination of application logic and data desired by a developer. Arbitrary objects can include text file pointers, binary file pointers, compiled executables, scripts, data base queries, shell commands, remote procedure calls, global variables, and local variables. The arbitrary object framework allows arbitrary objects to be referenced in a consistent manner regardless of the type. Also, the arbitrary object framework allows local arbitrary objects to either override global parent arbitrary objects or inherit capabilities and data from the global parent, regardless of the type of the local arbitrary object.

At step **22**, these arbitrary objects can be managed in an object library. The life cycle of these objects may be managed in a consistent manner using revision tracking, roll back, and sign off. At step **24**, objects can be deployed from the object library into a design framework to create the software application. Because the object pointers are not tied in any way to the functionality of the object, an object of one type can be easily replaced with another object of another type. This eliminates a common problem in content management systems of the inability to preview content within its appropriate location on the site or within the system. Normally, a special system made for the purpose of previewing a piece of content would have to be hard-coded to view the current approved live content for all other pieces except the piece in question. This multiplies the design problem, because changes in the design in the main site change all previous templates. In the method of the present invention, since all that exists within the framework is an arbitrary object, the arbitrary object can be swapped for another object that pulls the current piece content in question.

Using one embodiment of this invention, for example, the Features or Editorials page of a newspaper can be dynamically replaced. The present invention can execute all the normal objects that can be placed on the page to show the content as it would appear, and then take the one piece in question and replace it with a second object to be examined. Objects may be deployed globally across an entire system or locally within a specific area or sub-areas of a system.

FIG. **4** represents a flow diagram of another embodiment of the present invention. At step **30**, arbitrary objects can be generated. At step **32**, the arbitrary objects can be managed in an object library. Arbitrary objects can be deployed in a container page at step **34** to generate a web site.

Arbitrary objects may include any combination of application logic and data desired by a developer. Arbitrary objects can include text file pointers, binary file pointers, compiled executable scripts, database queries, shell commands, remote call procedures, global variables and local variables. Arbitrary objects may also include cached data queries and executables. The arbitrary object framework allows arbitrary objects to be referenced in a consistent manner regardless of the type of object. Also, the arbitrary object framework allows local arbitrary objects to either override global parent arbitrary objects or inherit capabilities and data from the global parent arbitrary object.

Arbitrary objects can execute any function that can be run or understood by the host computer system so that any underlying functionality of the operating system used by the host system can be defined as an object within the arbitrary framework. Legacy data, document objects, CGI programs, and database queries can all be encapsulated as objects within the arbitrary framework. The arbitrary object can be accessed by an arbitrary object name. Arbitrary objects are not tied to their functionality. One arbitrary object can be easily replaced with another arbitrary object of another type.

Arbitrary objects can be managed in an object library. The life cycle of the arbitrary objects may be managed in a consistent manner using revision tracking, roll-back, and sign-off. The object library can include separate specialized object libraries that can be administered separately by different developers in each area. For instance, for a web site used to generate a newspaper, there may be an advertising object library that is physically distinguished from other object libraries, such as an object library for sports or an object library for news. Therefore, queries for advertising can be created without impacting any other area of the web site.

Arbitrary objects can be deployed from the object library into a container page to generate the web site. The container page is a truly dynamic page. Unlike prior art methods, where a static copy of information is often pushed over a firewall to a live web site, the present invention incorporates object caching. An arbitrary object can be cached, rather than caching an entire page. When the arbitrary object is cached, certain elements of the arbitrary object can be specified as dynamic elements while others can be specified as static elements. Therefore, a web site can contain multiple dynamic

web pages wherein objects used to construct the form, function, and content of the web page can contain dynamic elements and static elements. This provides flexibility for what needs to be computed or processed at the time that someone, such as a web user, accesses the web page.

FIG. **5** shows the components of one environment of the present invention used to generate web sites. A user with web browser **40** can connect to web server **44** through internet or intranet **42**. Web server **44** can access static HTML web documents **46** as well as dynamic HTML documents **52**. Dynamic HTML web documents **52** can be created using WebOS Object Manager **50**. Dynamic HTML Web document **52** can include document objects **56**, shell scripts **58**, CGI programs **60**, and database queries **62**. Document objects **56**, shell scripts **58**, CGI programs **60**, and database queries **62** can be stored in WebOS object library **54**. Database queries **62** can result from extracting information from WebOS Information Database **68** and inputting the information into Dynamic HTML Web Template **66**.

User Profile and Password Database **70** can provide web sites or systems with a means to take advantage of customer profiles to look at customer preferences or history, and dynamically replace a website object with another object that contains content information matching the user profile or preferences. Thus, the web site or system can dynamically allocate the correct content for a customer. This is important in commerce applications. A customer's buying history can be examined for trend items and the customer presented products that match his or her profile. Present personalization systems are written purely in custom code and require an inordinately large amount of time to construct the custom applications necessary to interpret the preferences of an individual user.

The method of present invention can perform object caching. This means that an object can be cached instead of caching an entire page. Object caching permits specifying elements of an object to be dynamic and elements of the object to be static. A system user can thus have the flexibility of specifying what needs to be computed or processed at the time a user accesses the system versus trying to anticipate and calculate in advance and cache and post the object over to a server.

Many functions are stored within an object library on an arbitrary object framework such that those functions can be accessed by name arbitrarily. This is in contrast to a traditional model where the function must be explicitly invoked with all its parameters included. Objects may execute any function that can be run or understood by the host computer system so that any underlying functionality of the host's operating system can be defined as an object within the framework of the method of the present invention. The object library can contain legacy data, document objects, CGI programs, and/or database queries, that can all be encapsulated as objects within a framework and accessed from within a design. All that is needed is the name of the function in order to access the function.

Objects can be controlled to perform functions based on a profile of an individual and environmental variables, such as the type of browser, the country of the individual or the individual's IP address. A specific competitor may be blocked from seeing certain objects on a web page created using the method of the present invention.

A critical distinction between the present invention and previous object oriented development systems is the need to know how a function can be called and what to expect it to return, rather than just knowing the function's name. This means that typically the system administrator calls the name of an object and passes parameters to the object. Any and all variable information or environmental information can be available to every object. The environment space can be available to all objects executed and an object can arbitrarily take advantage of any of the environmental information, depending on the design of the object.

Different areas of a web site can be administered separately by different developers in each of these areas. An advertising object library can be physically distinguished from other object libraries, such as those for sports and news. An advertising programmer can create new queries for the advertising section of a site without having to worry about affecting other areas of the site.

The present invention allows different object types to be interchangeable. The object name is essentially just another variable in the environment. Also different variables can also be interchangeable. The object framework can be designed such that objects and variables can be kept in the same name space, every object can have access to all the environmental settings, and every object pointer can potentially be another name in the name space.

Object caching, rather than page caching can be implemented with the present invention. These objects can be stored in an object library. An object in the object library can be a file, a global variable, an executable script, a database query, a cached executable or a cached database query. This means that the results of a query can be stored in a static file using the object name as long as the static file has not expired. This is important if the query is a lengthy query.

A technical advantage of the present invention is that it allows for syndication. Syndication enables the content and function of a particular web site to be syndicated to another web site or web presentation. For instance, if a company would like to roll out a new look or syndicate its content and functionality to another business, this can be easily accomplished using the present invention. Since there is no application code resident in a web page itself, the same data can be repackaged in a number of different ways across multiple sites. There is no need to recode the design elements or design pages on the web site or recode any functions that are needed to access the content of the website. The present invention enables electronic store fronts to sell from a single source with a unique interface design. Also, newspaper chains can distribute international and national content from a single source and add local content themselves.

Another technical advantage of the present invention is that it allows for a single point of control when developing a web site. Therefore, if a large team of developers are working on a site, and multiple persons are contributing arbitrary objects to the overall arbitrary framework, then if one of the arbitrary objects has a security hole in it, the arbitrary object can be easily accessed in the object library and disabled. This security feature can immediately shut down that function across the entire web site and patch the security hole.

The present invention provides still another technical advantage in that it allows for personalization. Personalization enables companies that want to take advantage of a customer profile to look at the customer's preferences or histories and deploy information to the web site specific to the customer.

Another technical advantage of the present invention allows for profiling. Profiling enables control over the arbitrary objects presented in a web site based on a profile of the individual accessing the web site. Profiling entails determining different environmental variables such as the type of browser hitting the site, the country of the individual accessing the site, and/or the individual's IP address. This can

enable a company to present specific information to the individual based on the individual's environmental variables.

Although the present invention has been described in detail herein with reference to the illustrative embodiments, it should be understood that the description is by way of example only and is not to be construed in a limiting sense. It is to be further understood, therefore, that numerous changes in the details of the embodiments of this invention and additional embodiments of this invention will be apparent to, and may be made by, persons of ordinary skill in the art having reference to this description. It is contemplated that all such changes and additional embodiments are within the spirit and true scope of this invention as claimed below.

The invention claimed is:

1. A system for generating a computer application on a host system in an arbitrary object framework that separates a content of said computer application, a form of said computer application, and a functionality of said computer application, said system including a computer comprising a processor and a memory operably coupled to said processor, said memory being configured for storing a computer program executable by said processor, said computer program comprising:

a first set of executable instructions for creating arbitrary objects with corresponding arbitrary names of content objects used in generating said content of said computer application, form objects used in defining said form of said computer application, and function objects used in executing said functionality of said computer application each arbitrary object being separate from each other arbitrary object;

a second set of executable instructions for managing said arbitrary objects in an arbitrary object library; and

a third set of executable instructions for deploying said arbitrary objects from said arbitrary object library into a design framework to create said computer application.

2. The system of claim **1**, wherein said computer application is a web site.

3. The system of claim **1**, wherein each of said various object types include a type selected from the group consisting of: text file pointers; binary file pointers;

compiled executables; shell commands; remote procedure calls; global variables; cached executables; cached database queries; local variables; and local objects and global parent objects, wherein said local objects are capable of overriding said global parent objects, and wherein said local objects are capable of inheriting data from said global parent objects.

4. The system of claim **1**, wherein the third set of executable instructions are for deploying arbitrary objects locally.

5. The system of claim **1**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for revision tracking.

6. The system of claim **1**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using rollback.

7. The system of claim **1**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using signoff.

8. The system of claim **1**, wherein the third set of exectable instructions include instructions to access and deploy arbitrary objects into said design framework using said corresponding arbitrary names.

9. The system of claim **1**, further comprising executable instructions for swapping an arbitrary object of one type with an arbitrary object of another type.

10. The system of claim **1**, further comprising executable instructions for caching objects.

11. The system of claim **10**, wherein the executable instructions for caching objects further comprises exectable instructions for specifying some elements of an arbitrary object to be dynamic elements and specifying some elements of said arbitrary object to be static elements.

12. The system of claim **1**, further comprising executable instructions for generating arbitrary objects in a programming language that is compatible and supported by said host system.

13. A system for generating a web site on a host system in an arbitrary object framework that separates a content of said web site, a form of said web site, and a functionality of said web site, said system including a computer comprising a processor and a memory operably coupled to said processor, said memory being configured for storing a computer program executable by said processor, said computer program comprising:

a first set of executable instructions for creating arbitrary objects with corresponding arbitrary names of content objects used in generating said content of said web site, form objects used in defining said form of said web site, and function objects used in executing said functionality of said web site, each arbitrary object being separate from each other arbitrary object;

a second set of executable instructions for managing said arbitrary objects in an arbitrary object library; and

a third set of executable instructions for deploying said arbitrary objects from said arbitrary object library to a container page to create said web site.

14. The system of claim **13**, wherein each of said various object types include a type selected from the group consisting of: text file pointers; binary file pointers; compiled executables; shell commands; remote procedure calls; global variables; cached executables; cached database queries; local variables; and local objects and global parent objects, wherein said local objects are capable of overriding said global parent objects, and wherein said local objects are capable of inheriting data from said global parent objects.

15. The system of claim **13**, wherein the third set of executable instructions are for deploying arbitrary objects locally.

16. The system of claim **13**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for revision tracking.

17. The system of claim **13**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using rollback.

18. The system of claim **13**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using signoff.

19. The system of claim **13**, wherein the third set of executable instructions include instructions to access and deploy arbitrary objects into said design framework using said corresponding arbitrary names.

20. The system of claim **19**, wherein the third set of executable instructions is capable of accessing and deploying the arbitrary objects into said container page using said corresponding arbitrary names.

21. A system for generating a computer application on a host system in an arbitrary object framework that separates a content of said computer application, a form of said computer application, and a functionality of said computer application, said system including a computer comprising a processor and a memory operably coupled to said processor, said memory being configured for storing a computer program executable by said processor, said computer program comprising:

a first set of executable instructions for creating arbitrary objects with corresponding arbitrary names of content objects used in generating said content of said computer application, form objects used in defining said form of said computer application, and function objects used in executing said functionality of said computer application, each arbitrary object being callable by name only, each arbitrary object being independently modifiable without corresponding modifications being made to any other arbitrary object, and each arbitrary object further being interchangable with other arbitrary objects;

a second set of executable instructions for managing said arbitrary objects in an arbitrary object library; and

a third set of executable instructions for deploying said arbitrary objects from said arbitrary object library into a design framework to create said computer application.

22. The system of claim **21**, wherein said computer application is a web site.

23. The system of claim **21**, wherein each of said various object types include a type selected from the group consisting of: text file pointers; binary file pointers; compiled executables; shell commands; remote procedure calls; global variables; cached executables; cached database queries; local variables; and local objects and global parent objects, wherein said local objects are capable of overriding said global parent objects, and wherein said local objects are capable of inheriting data from said global parent objects.

24. The system of claim **21**, wherein the third set of executable instructions are for deploying arbitrary objects locally.

25. The system of claim **21**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for revision tracking.

26. The system of claim **21**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using rollback.

27. The system of claim **21**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using signoff.

28. The system of claim **21**, wherein the third set of executable instructions include instructions to access and deploy arbitrary objects into said design framework using said corresponding arbitrary names.

29. The system of claim **21**, further comprising executable instructions for swapping an arbitrary object of one type with an arbitrary object of another type.

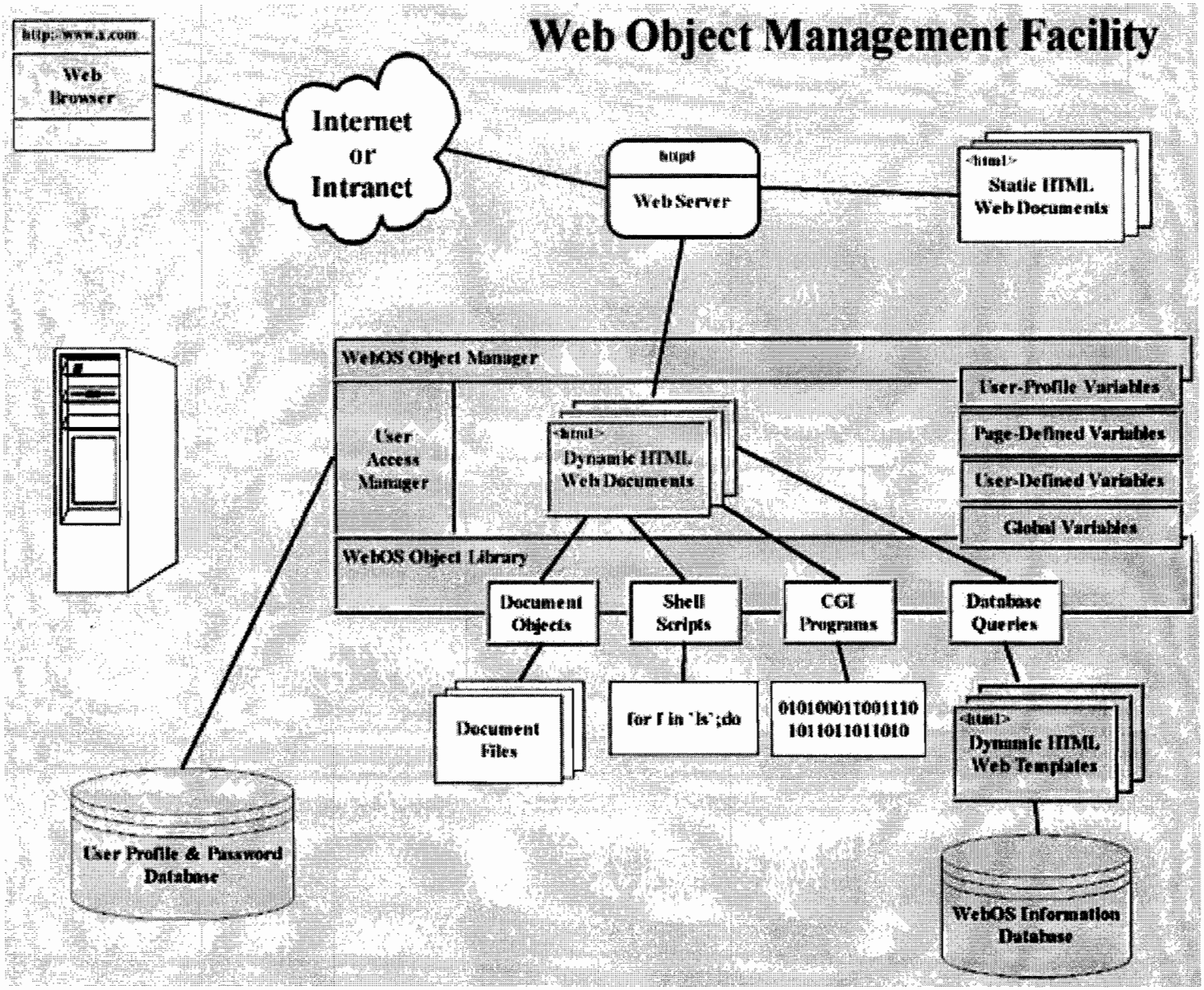
30. The system of claim **21**, further comprising executable instructions for caching objects.

31. The system of claim **30**, wherein the executable instructions for caching objects further comprises exectable instructions for specifying some elements of an arbitrary object to be dynamic elements and specifying some elements of said arbitrary object to be static elements.

32. The system of claim **21**, further comprising executable instructions for generating arbitrary objects in a programming language that is compatible and supported by said host system.

* * * * *

EXHIBIT C


Web Object Management Facility
http://www.x.com
Web Browser
Internet or Intranet
http
Web Server
<html>
Static HTML Web Documents
WebOS Object Manager
User Access Manager
<html>
Dynamic HTML Web Documents
User-Profile Variables
Page-Defined Variables
User-Defined Variables
Global Variables
WebOS Object Library
Document Objects
Shell Scripts
CGI Programs
Database Queries
Document Files
for f in `ls`;do
0101000110011101011011011010
<html>
Dynamic HTML Web Templates
User Profile & Password Database
WebOS Information Database

[Products] [Services] [Support] [Portfolio] [Demos]
[Inquire] [Success Stories] [Adhesive Media]

Powered by

Send mail to webmaster@adhesive.com with questions or comments about this web site.

Last modified: October 15, 1996

US007716629B2

(12) **United States Patent**
McAuley

(10) **Patent No.: US 7,716,629 B2**
(45) **Date of Patent: *May 11, 2010**

(54) **SYSTEM AND METHOD FOR GENERATING WEB SITES IN AN ARBITRARY OBJECT FRAMEWORK**

(75) Inventor: **Aubrey McAuley**, Austin, TX (US)

(73) Assignee: **Vertical Computer Systems, Inc.**, Fort Worth, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 613 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/999,911**

(22) Filed: **Nov. 29, 2004**

(65) **Prior Publication Data**

US 2005/0154486 A1 Jul. 14, 2005

Related U.S. Application Data

(63) Continuation of application No. 09/410,334, filed on Oct. 1, 1999, now Pat. No. 6,826,744.

(51) **Int. Cl.**
G06F 9/45 (2006.01)

(52) **U.S. Cl.** **717/100**

(58) **Field of Classification Search** 717/106, 717/100, 149

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,544,302	A *	8/1996	Nguyen	715/837
5,555,365	A *	9/1996	Selby et al.	715/765
5,894,554	A *	4/1999	Lowery et al.	709/203
5,895,476	A	4/1999	Orr et al.	
5,903,894	A *	5/1999	Reneris	707/100
5,930,512	A *	7/1999	Boden et al.	717/102
5,956,736	A *	9/1999	Hanson et al.	715/513
6,026,433	A *	2/2000	D'Arlach et al.	709/217
6,028,998	A *	2/2000	Gloudeman et al.	717/108
6,052,670	A *	4/2000	Johnson	705/27
6,199,082	B1 *	3/2001	Ferrel et al.	715/522
6,219,680	B1 *	4/2001	Bernardo et al.	715/501.1
6,226,648	B1 *	5/2001	Appleman et al.	707/102
6,247,032	B1 *	6/2001	Bernardo et al.	715/530

(Continued)

FOREIGN PATENT DOCUMENTS

CA	2110970	6/1995

OTHER PUBLICATIONS

Adhesive Media, Inc., Dynamic Web Management Tools, WebOS/NewsFlash/SiteFlash, Assorted Documents, pp. MSVERT002 through MSVERT126.

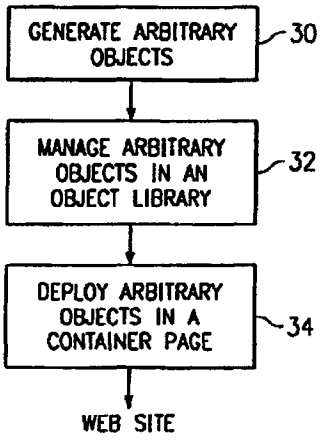
(Continued)

Primary Examiner—John Chavis
(74) *Attorney, Agent, or Firm*—Scheef & Stone, L.L.P.; Jack D. Stone, Jr.

(57) **ABSTRACT**

A method and system for generating a computer application is disclosed. The computer application is generated on a host system in an arbitrary object framework that separates a content of said computer application, a form of said computer application and a functionality of said computer application. Arbitrary objects are created with corresponding arbitrary names of various object types for generating said content of said computer application, said form of said computer application, and said functionality of said computer application. The arbitrary objects are managed in an object library. The arbitrary objects are deployed from said object library into a design framework to create said computer application.

32 Claims, 2 Drawing Sheets



U.S. PATENT DOCUMENTS

6,253,282	B1 *	6/2001	Gish	711/113
6,308,188	B1 *	10/2001	Bernardo et al.	715/530
6,553,563	B2	4/2003	Ambrose et al.	
6,574,635	B2	6/2003	Stauber et al.	
7,284,193	B1	10/2007	Lindhorst	
2002/0147805	A1 *	10/2002	Leshem et al.	709/223
2002/0161734	A1	10/2002	Stauber et al.	

OTHER PUBLICATIONS

Apple Computer, Inc., Inc., "The WebObjects Dynamic Elements Reference," http://developer.apple.com/ Documentation/ WebObjects/Reference/DynamicElements/DynamicElements.pdf, Jan. 10, 2006, WebObjects and the Enterprise Objects Framework, Chapt. 16.

Banick, et al.., "Using Microsoft FrontPage 98," Que Corporation (1998), Microsoft FrontPage, Chapt. 3.

Burke, "Web Databases with Cold Fusion 3," The McGraw-Hill Companies, Inc. (1998), Chapts. 1, 6, 9, and 15.

Buyens, "Running Microsoft FrontPage 98," Microsoft Press, Jim Buyens (1997), Microsoft FrontPage 98, Chapt. 4.

Darnell, et al., Using Macromedia Dreamweaver 1.2, Que (1998), Macromedia Dreamweaver, Chapts. 8 and 12.

Forta, "The ColdFusion 4.0 Web Application Construction Kit," Que (1998), ColdFusion 4.0, Chapts. 10 through 14.

Gray, "Web Publishing with Adobe PageMill 2," Ventana Communications Group, Inc., (1997), Adobe PageMill1, Chapts. 1 and 3.

Howell, "The Complete Idiot's Guide to Microsoft Visual InterDev," Que Corporation (1997), Microsoft Visual InterDev, Chapt. 4.

Jones, et al.,, "Cascading Style Sheets: A Primer," MIS:Press, Big Tent Media Labs, LLC (1998), Dynamic HTML and Cascading Style Sheets, Chapt. 4.

Martin, et al., "Object-Oriented Methods: A Foundation," P T R Prentice Hall, (1995), Object-Oriented Programming Art, Chapts. 1 through 3.

Meyer, "Object-Oriented Software Construction," 2nd Ed., Prentice Hall PTR, (1997), Object-Oriented Programming Art, Chapt. 2.

Pollizi, "Separating Form from Function: The StarView Experience," Astronomical Data Analysis Software and Systems III, ASP Conference Series, vol. 61, pp. 88-91 (1994).

Prague, et al., "Access 97 Bible," IDG Books Worldwide, Inc. (1997), Microsoft Access 97, Chapts. 1 through 9, (Part 1).

Prague, et al., "Access 97 Bible," IDG Books Worldwide, Inc. (1997), Microsoft Access 97, Chapts. 10 through18, (Part 2).

Prague, et al., "Access 97 Bible," IDG Books Worldwide, Inc. (1997), Microsoft Access 97, Chapts. 19 through 26, (Part 3).

Prague, et al., "Access 97 Bible," IDG Books Worldwide, Inc. (1997), Microsoft Access 97, Chapts. 27 through 33, and Appendices (Part 4).

Webster, "NetObjects Fusion Handbook," Hayden Books (1996), Webster, NetObjects Fusion, Chapts. 9 through 13.

* cited by examiner

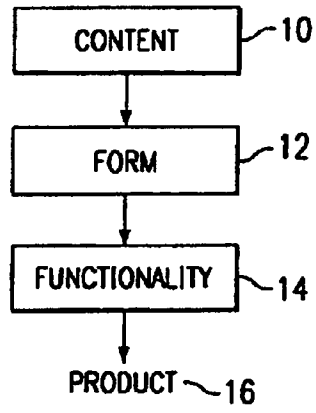


FIG. 1
(PRIOR ART)

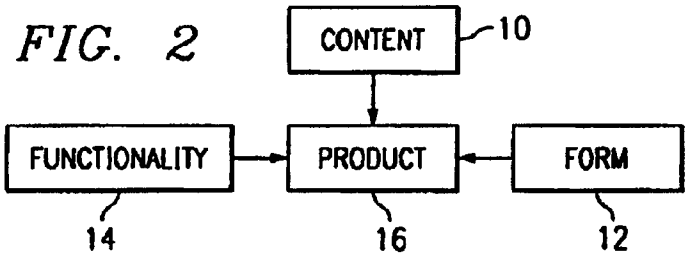


FIG. 2

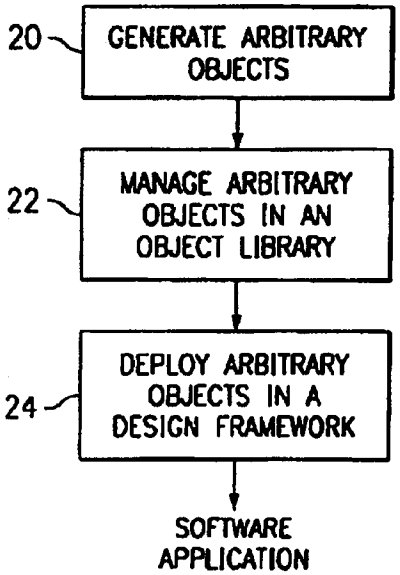


FIG. 3

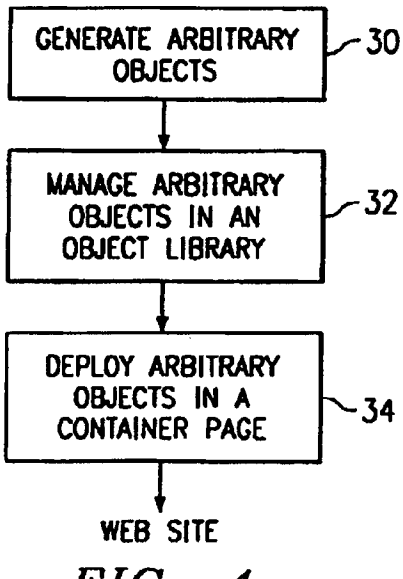


FIG. 4

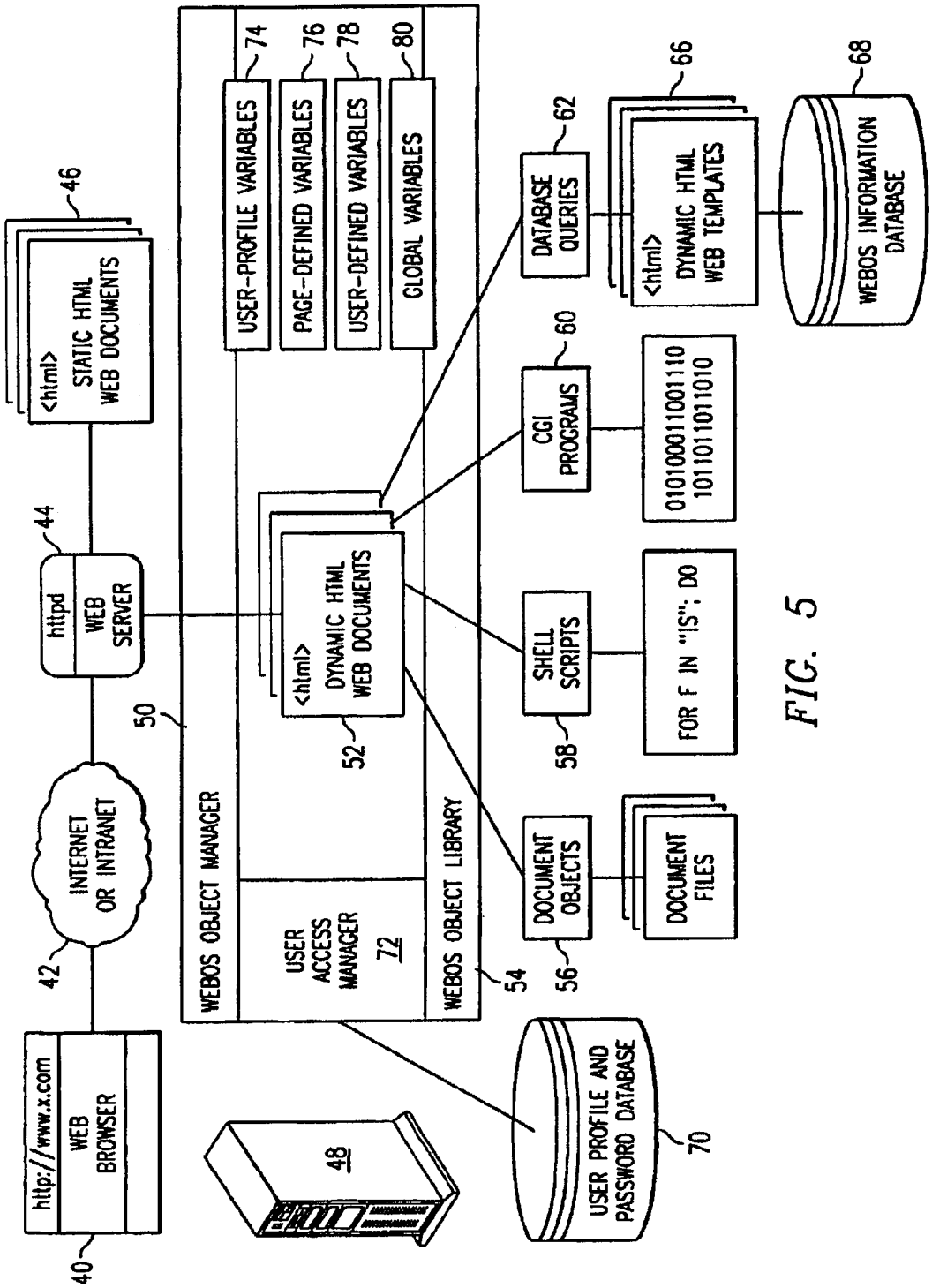

http://www.x.com
WEB BROWSER
40
INTERNET OR INTRANET
42
httpd WEB SERVER
44
<html> STATIC HTML WEB DOCUMENTS
46
48
WEBOS OBJECT MANAGER
50
<html> DYNAMIC HTML WEB DOCUMENTS
52
USER ACCESS MANAGER 72
WEBOS OBJECT LIBRARY
54
USER-PROFILE VARIABLES
74
PAGE-DEFINED VARIABLES
76
USER-DEFINED VARIABLES
78
GLOBAL VARIABLES
80
DOCUMENT OBJECTS
56
DOCUMENT FILES
SHELL SCRIPTS
58
FOR F IN "IS"; DO
CGI PROGRAMS
60
0101000111001110 1011011011010
DATABASE QUERIES
62
<html> DYNAMIC HTML WEB TEMPLATES
66
WEBOS INFORMATION DATABASE
68
USER PROFILE AND PASSWORD DATABASE
70

FIG. 5

SYSTEM AND METHOD FOR GENERATING WEB SITES IN AN ARBITRARY OBJECT FRAMEWORK

RELATED APPLICATIONS

This application is a continuation of U.S. application Ser. No. 09/410,334, filed Oct. 1, 1999, now U.S. Pat. No. 6,826,744.

TECHNICAL FIELD OF THE INVENTION

This invention relates generally to systems and methods for generating software applications in an arbitrary object framework, and more specifically to systems and methods for generating web sites in an arbitrary object framework.

BACKGROUND OF THE INVENTION

Three processes used to create complex software applications such as web sites are form, function, and content. Form includes graphic designs, user interfaces, and graphical representations created by a designer or a group of designers. Function includes logical functionality, which can be software code created by a programmer or group of programmers. Form includes informative content. Informative content can include written, recorded, or illustrated documentation, such as photographs, illustrations, product marketing material, and news articles. Content can be created by writers, photographers, artists, reporters, or editors.

Currently, typical workflows dictate a serial approach to integrating the form, function, and content to create complex software applications such as a web site. The serial approach is illustrated in FIG. **1**. In FIG. **1**, content **10** for a complex software application can be chosen or created. Form **12** for the presentation of content **10** can then be created. Functionality **14** can then be generated using code to create the complex software application (product **16**) with the desired information (content **10**) and style (form **12**). Using the method illustrated in FIG. **1**, every final component of the complex software application must be manipulated by a programmer before it is ready to be used. The exact workflow may vary from industry to industry or business to business, but the basic restrictions are generally the same.

A traditional approach such as that illustrated in FIG. **1**, may create unwanted bottlenecks in the production process. Each upstream revision, such as a change of content **10** or design **12**, forces a repetition of the entire process. As an example, consider a web site for a large newspaper. The web site may have a function that can include a file into the web site. The marketing department may decide to change the appearance of the header on the web site depending on the browser of a user. In this case, a programmer may need to invoke an external script or embed some specific logic within the web site. Unfortunately, if there is a large web site with thousands of pages of information stored on a server, the programmer may have to change every one of the thousands of pages. Therefore, a small change by the marketing department can cause a large burden on the programming department.

Prior art solutions have succeeded in partially separating some of these functions. Notably, content management databases and digital repositories provide a means of separating content from form and function. Likewise, sophisticated software development teams frequently employ internal code structuring techniques that can help to minimize dependencies between interface designs and the functions they access.

However, content management tools typically fail to address form/function issues. Therefore, there can still be production slow-downs due to changes in form that require a subsequent change in functionality.

SUMMARY OF THE INVENTION

Therefore a need exists for a method of generating complex software applications that reduces or eliminates production delays and the workload for programmers due to changes in content and/or form. This method should separate form, content and function so that each area can be independently changed.

The present invention provides a system and method for generating software applications that substantially eliminates or reduces disadvantages and problems associated with previously developed systems and methods used for generation of software applications. More specifically, the present invention provides a method for generating software applications in an arbitrary object framework. The method of the present invention separates content, form, and function of the computer application so that each may be accessed or modified independently. The method of this invention includes creating arbitrary objects, managing the arbitrary objects throughout their life cycle, and deploying the arbitrary objects in a design framework for use in complex computer applications.

The present invention provides an important technical advantage in that content, form, and function are separated from each other in the generation of the software application. Therefore, changes in design or content do not require the intervention of a programmer. This advantage decreases the time needed to change various aspects of the software application. Consequently, cost is reduced and versatility is increased.

The present invention provides another technical advantage in that users are not required to use a proprietary language to encode. These arbitrary objects may include encapsulated legacy data, legacy systems and custom programming logic from essentially any source in which they may reside. Any language supported by the host system, or any language that can be interfaced to by the host system, can be used to generate an object within the application.

The present invention provides yet another technical advantage in that it can provide a single point of administrative authority that can reduce security risks. For instance, a large team of programmers can work on developing a large group of arbitrary objects within the object library. If one object has a security hole, an administrator can enter the object library and disable that arbitrary object.

Still another technical advantage of the present invention is that it enables syndication of the software application. As noted above, functionality is separate from form and content. Consequently, a user can easily introduce a new look for the application or syndicate the content and functionality of the application to another group without having to recode all of the objects needed to access content.

Another technical advantage of the present invention is that it allows for personalization and profiling. With personalization, the web presentation is tailored to the specific needs of the web user based on the user's past history. Profiling also enables tailoring a web site or presentation. Profiling is dependent on environmental variables such as browser type or IP address.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the present invention and the advantages thereof may be acquired by referring to

the following description, taken in conjunction with the accompanying drawings in which like reference numbers indicate like features and wherein:

FIG. **1** illustrates a prior art workflow diagram for generating a software product;

FIG. **2** is a hierarchical workflow diagram for one embodiment of the present invention;

FIG. **3** is a flow diagram for one embodiment of the present invention;

FIG. **4** is a flow diagram for another embodiment of the present invention; and

FIG. **5** is a diagram illustrating the components of one environment of the present invention used to generate web sites.

DETAILED DESCRIPTION OF THE INVENTION

Preferred embodiments of the present invention are illustrated in the FIGUREs, like numerals being used to refer to like and corresponding parts of various drawings.

The present invention provides a system and method for using a hierarchical, arbitrary object framework for generating software applications. The method separates content, form, and function of the software application so that each can be accessed or modified independently. The method of this invention includes creating arbitrary objects, managing the arbitrary objects in an object library, and deploying the arbitrary objects in a design framework for use in computer applications.

FIG. **2** is a hierarchical workflow diagram for the present invention. Product **16** includes three contributing groups: content **10**, form **12**, and functionality **14**. Content **10** can include written, recorded, or illustrated collateral such as documentation, photographic illustrations, product marketing material, and articles. Form **12** can include graphic designs such as user interfaces and graphical presentations. Function **14** can include the logical functionality of software code and scripts. The hierarchical framework separates content **10**, form **12**, and functionality **14** to generate product **16**. Product **16** may be a computer software application such as a web site. Since content **10**, design **12**, and functionality **14** are separate entities independent of each other, modification in one group does not require corresponding modifications in another group. Each group can contribute to product **16** directly.

FIG. **3** is a flow diagram of one embodiment of the present invention. At step **20**, arbitrary objects can be generated. Arbitrary objects may include any combination of application logic and data desired by a developer. Arbitrary objects can include text file pointers, binary file pointers, compiled executables, scripts, data base queries, shell commands, remote procedure calls, global variables, and local variables. The arbitrary object framework allows arbitrary objects to be referenced in a consistent manner regardless of the type. Also, the arbitrary object framework allows local arbitrary objects to either override global parent arbitrary objects or inherit capabilities and data from the global parent, regardless of the type of the local arbitrary object.

At step **22**, these arbitrary objects can be managed in an object library. The life cycle of these objects may be managed in a consistent manner using revision tracking, roll back, and sign off. At step **24**, objects can be deployed from the object library into a design framework to create the software application. Because the object pointers are not tied in any way to the functionality of the object, an object of one type can be easily replaced with another object of another type. This eliminates a common problem in content management systems of the inability to preview content within its appropriate location on the site or within the system. Normally, a special system made for the purpose of previewing a piece of content would have to be hard-coded to view the current approved live content for all other pieces except the piece in question. This multiplies the design problem, because changes in the design in the main site change all previous templates. In the method of the present invention, since all that exists within the framework is an arbitrary object, the arbitrary object can be swapped for another object that pulls the current piece content in question.

Using one embodiment of this invention, for example, the Features or Editorials page of a newspaper can be dynamically replaced. The present invention can execute all the normal objects that can be placed on the page to show the content as it would appear, and then take the one piece in question and replace it with a second object to be examined. Objects may be deployed globally across an entire system or locally within a specific area or sub-areas of a system.

FIG. **4** represents a flow diagram of another embodiment of the present invention. At step **30**, arbitrary objects can be generated. At step **32**, the arbitrary objects can be managed in an object library. Arbitrary objects can be deployed in a container page at step **34** to generate a web site.

Arbitrary objects may include any combination of application logic and data desired by a developer. Arbitrary objects can include text file pointers, binary file pointers, compiled executable scripts, database queries, shell commands, remote call procedures, global variables and local variables. Arbitrary objects may also include cached data queries and executables. The arbitrary object framework allows arbitrary objects to be referenced in a consistent manner regardless of the type of object. Also, the arbitrary object framework allows local arbitrary objects to either override global parent arbitrary objects or inherit capabilities and data from the global parent arbitrary object.

Arbitrary objects can execute any function that can be run or understood by the host computer system so that any underlying functionality of the operating system used by the host system can be defined as an object within the arbitrary framework. Legacy data, document objects, CGI programs, and database queries can all be encapsulated as objects within the arbitrary framework. The arbitrary object can be accessed by an arbitrary object name. Arbitrary objects are not tied to their functionality. One arbitrary object can be easily replaced with another arbitrary object of another type.

Arbitrary objects can be managed in an object library. The life cycle of the arbitrary objects may be managed in a consistent manner using revision tracking, roll-back, and sign-off. The object library can include separate specialized object libraries that can be administered separately by different developers in each area. For instance, for a web site used to generate a newspaper, there may be an advertising object library that is physically distinguished from other object libraries, such as an object library for sports or an object library for news. Therefore, queries for advertising can be created without impacting any other area of the web site.

Arbitrary objects can be deployed from the object library into a container page to generate the web site. The container page is a truly dynamic page. Unlike prior art methods, where a static copy of information is often pushed over a firewall to a live web site, the present invention incorporates object caching. An arbitrary object can be cached, rather than caching an entire page. When the arbitrary object is cached, certain elements of the arbitrary object can be specified as dynamic elements while others can be specified as static elements. Therefore, a web site can contain multiple dynamic

web pages wherein objects used to construct the form, function, and content of the web page can contain dynamic elements and static elements. This provides flexibility for what needs to be computed or processed at the time that someone, such as a web user, accesses the web page.

FIG. **5** shows the components of one environment of the present invention used to generate web sites. A user with web browser **40** can connect to web server **44** through internet or intranet **42**. Web server **44** can access static HTML web documents **46** as well as dynamic HTML documents **52**. Dynamic HTML web documents **52** can be created using WebOS Object Manager **50**. Dynamic HTML Web document **52** can include document objects **56**, shell scripts **58**, CGI programs **60**, and database queries **62**. Document objects **56**, shell scripts **58**, CGI programs **60**, and database queries **62** can be stored in WebOS object library **54**. Database queries **62** can result from extracting information from WebOS Information Database **68** and inputting the information into Dynamic HTML Web Template **66**.

User Profile and Password Database **70** can provide web sites or systems with a means to take advantage of customer profiles to look at customer preferences or history, and dynamically replace a website object with another object that contains content information matching the user profile or preferences. Thus, the web site or system can dynamically allocate the correct content for a customer. This is important in commerce applications. A customer's buying history can be examined for trend items and the customer presented products that match his or her profile. Present personalization systems are written purely in custom code and require an inordinately large amount of time to construct the custom applications necessary to interpret the preferences of an individual user.

The method of present invention can perform object caching. This means that an object can be cached instead of caching an entire page. Object caching permits specifying elements of an object to be dynamic and elements of the object to be static. A system user can thus have the flexibility of specifying what needs to be computed or processed at the time a user accesses the system versus trying to anticipate and calculate in advance and cache and post the object over to a server.

Many functions are stored within an object library on an arbitrary object framework such that those functions can be accessed by name arbitrarily. This is in contrast to a traditional model where the function must be explicitly invoked with all its parameters included. Objects may execute any function that can be run or understood by the host computer system so that any underlying functionality of the host's operating system can be defined as an object within the framework of the method of the present invention. The object library can contain legacy data, document objects, CGI programs, and/or database queries, that can all be encapsulated as objects within a framework and accessed from within a design. All that is needed is the name of the function in order to access the function.

Objects can be controlled to perform functions based on a profile of an individual and environmental variables, such as the type of browser, the country of the individual or the individual's IP address. A specific competitor may be blocked from seeing certain objects on a web page created using the method of the present invention.

A critical distinction between the present invention and previous object oriented development systems is the need to know how a function can be called and what to expect it to return, rather than just knowing the function's name. This means that typically the system administrator calls the name of an object and passes parameters to the object. Any and all variable information or environmental information can be available to every object. The environment space can be available to all objects executed and an object can arbitrarily take advantage of any of the environmental information, depending on the design of the object.

Different areas of a web site can be administered separately by different developers in each of these areas. An advertising object library can be physically distinguished from other object libraries, such as those for sports and news. An advertising programmer can create new queries for the advertising section of a site without having to worry about affecting other areas of the site.

The present invention allows different object types to be interchangeable. The object name is essentially just another variable in the environment. Also different variables can also be interchangeable. The object framework can be designed such that objects and variables can be kept in the same name space, every object can have access to all the environmental settings, and every object pointer can potentially be another name in the name space.

Object caching, rather than page caching can be implemented with the present invention. These objects can be stored in an object library. An object in the object library can be a file, a global variable, an executable script, a database query, a cached executable or a cached database query. This means that the results of a query can be stored in a static file using the object name as long as the static file has not expired. This is important if the query is a lengthy query.

A technical advantage of the present invention is that it allows for syndication. Syndication enables the content and function of a particular web site to be syndicated to another web site or web presentation. For instance, if a company would like to roll out a new look or syndicate its content and functionality to another business, this can be easily accomplished using the present invention. Since there is no application code resident in a web page itself, the same data can be repackaged in a number of different ways across multiple sites. There is no need to recode the design elements or design pages on the web site or recode any functions that are needed to access the content of the website. The present invention enables electronic store fronts to sell from a single source with a unique interface design. Also, newspaper chains can distribute international and national content from a single source and add local content themselves.

Another technical advantage of the present invention is that it allows for a single point of control when developing a web site. Therefore, if a large team of developers are working on a site, and multiple persons are contributing arbitrary objects to the overall arbitrary framework, then if one of the arbitrary objects has a security hole in it, the arbitrary object can be easily accessed in the object library and disabled. This security feature can immediately shut down that function across the entire web site and patch the security hole.

The present invention provides still another technical advantage in that it allows for personalization. Personalization enables companies that want to take advantage of a customer profile to look at the customer's preferences or histories and deploy information to the web site specific to the customer.

Another technical advantage of the present invention allows for profiling. Profiling enables control over the arbitrary objects presented in a web site based on a profile of the individual accessing the web site. Profiling entails determining different environmental variables such as the type of browser hitting the site, the country of the individual accessing the site, and/or the individual's IP address. This can

enable a company to present specific information to the individual based on the individual's environmental variables.

Although the present invention has been described in detail herein with reference to the illustrative embodiments, it should be understood that the description is by way of example only and is not to be construed in a limiting sense. It is to be further understood, therefore, that numerous changes in the details of the embodiments of this invention and additional embodiments of this invention will be apparent to, and may be made by, persons of ordinary skill in the art having reference to this description. It is contemplated that all such changes and additional embodiments are within the spirit and true scope of this invention as claimed below.

The invention claimed is:

1. A system for generating a computer application on a host system in an arbitrary object framework that separates a content of said computer application, a form of said computer application, and a functionality of said computer application, said system including a computer comprising a processor and a memory operably coupled to said processor, said memory being configured for storing a computer program executable by said processor, said computer program comprising:

a first set of executable instructions for creating arbitrary objects with corresponding arbitrary names of content objects used in generating said content of said computer application, form objects used in defining said form of said computer application, and function objects used in executing said functionality of said computer application each arbitrary object being separate from each other arbitrary object;

a second set of executable instructions for managing said arbitrary objects in an arbitrary object library; and

a third set of executable instructions for deploying said arbitrary objects from said arbitrary object library into a design framework to create said computer application.

2. The system of claim **1**, wherein said computer application is a web site.

3. The system of claim **1**, wherein each of said various object types include a type selected from the group consisting of: text file pointers; binary file pointers;

compiled executables; shell commands; remote procedure calls; global variables; cached executables; cached database queries; local variables; and local objects and global parent objects, wherein said local objects are capable of overriding said global parent objects, and wherein said local objects are capable of inheriting data from said global parent objects.

4. The system of claim **1**, wherein the third set of executable instructions are for deploying arbitrary objects locally.

5. The system of claim **1**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for revision tracking.

6. The system of claim **1**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using rollback.

7. The system of claim **1**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using signoff.

8. The system of claim **1**, wherein the third set of exectable instructions include instructions to access and deploy arbitrary objects into said design framework using said corresponding arbitrary names.

9. The system of claim **1**, further comprising executable instructions for swapping an arbitrary object of one type with an arbitrary object of another type.

10. The system of claim **1**, further comprising executable instructions for caching objects.

11. The system of claim **10**, wherein the executable instructions for caching objects further comprises exectable instructions for specifying some elements of an arbitrary object to be dynamic elements and specifying some elements of said arbitrary object to be static elements.

12. The system of claim **1**, further comprising executable instructions for generating arbitrary objects in a programming language that is compatible and supported by said host system.

13. A system for generating a web site on a host system in an arbitrary object framework that separates a content of said web site, a form of said web site, and a functionality of said web site, said system including a computer comprising a processor and a memory operably coupled to said processor, said memory being configured for storing a computer program executable by said processor, said computer program comprising:

a first set of executable instructions for creating arbitrary objects with corresponding arbitrary names of content objects used in generating said content of said web site, form objects used in defining said form of said web site, and function objects used in executing said functionality of said web site, each arbitrary object being separate from each other arbitrary object;

a second set of executable instructions for managing said arbitrary objects in an arbitrary object library; and

a third set of executable instructions for deploying said arbitrary objects from said arbitrary object library to a container page to create said web site.

14. The system of claim **13**, wherein each of said various object types include a type selected from the group consisting of: text file pointers; binary file pointers; compiled executables; shell commands; remote procedure calls; global variables; cached executables; cached database queries; local variables; and local objects and global parent objects, wherein said local objects are capable of overriding said global parent objects, and wherein said local objects are capable of inheriting data from said global parent objects.

15. The system of claim **13**, wherein the third set of executable instructions are for deploying arbitrary objects locally.

16. The system of claim **13**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for revision tracking.

17. The system of claim **13**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using rollback.

18. The system of claim **13**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using signoff.

19. The system of claim **13**, wherein the third set of executable instructions include instructions to access and deploy arbitrary objects into said design framework using said corresponding arbitrary names.

20. The system of claim **19**, wherein the third set of executable instructions is capable of accessing and deploying the arbitrary objects into said container page using said corresponding arbitrary names.

21. A system for generating a computer application on a host system in an arbitrary object framework that separates a content of said computer application, a form of said computer application, and a functionality of said computer application, said system including a computer comprising a processor and a memory operably coupled to said processor, said memory being configured for storing a computer program executable by said processor, said computer program comprising:

a first set of executable instructions for creating arbitrary objects with corresponding arbitrary names of content objects used in generating said content of said computer application, form objects used in defining said form of said computer application, and function objects used in executing said functionality of said computer application, each arbitrary object being callable by name only, each arbitrary object being independently modifiable without corresponding modifications being made to any other arbitrary object, and each arbitrary object further being interchangable with other arbitrary objects;

a second set of executable instructions for managing said arbitrary objects in an arbitrary object library; and

a third set of executable instructions for deploying said arbitrary objects from said arbitrary object library into a design framework to create said computer application.

22. The system of claim **21**, wherein said computer application is a web site.

23. The system of claim **21**, wherein each of said various object types include a type selected from the group consisting of: text file pointers; binary file pointers; compiled executables; shell commands; remote procedure calls; global variables; cached executables; cached database queries; local variables; and local objects and global parent objects, wherein said local objects are capable of overriding said global parent objects, and wherein said local objects are capable of inheriting data from said global parent objects.

24. The system of claim **21**, wherein the third set of executable instructions are for deploying arbitrary objects locally.

25. The system of claim **21**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for revision tracking.

26. The system of claim **21**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using rollback.

27. The system of claim **21**, wherein the second set of executable instructions for managing said arbitrary objects further comprises executable instructions for using signoff.

28. The system of claim **21**, wherein the third set of exectable instructions include instructions to access and deploy arbitrary objects into said design framework using said corresponding arbitrary names.

29. The system of claim **21**, further comprising executable instructions for swapping an arbitrary object of one type with an arbitrary object of another type.

30. The system of claim **21**, further comprising executable instructions for caching objects.

31. The system of claim **30**, wherein the executable instructions for caching objects further comprises exectable instructions for specifying some elements of an arbitrary object to be dynamic elements and specifying some elements of said arbitrary object to be static elements.

32. The system of claim **21**, further comprising executable instructions for generating arbitrary objects in a programming language that is compatible and supported by said host system.

* * * * *