

CLAIMS

1. An apparatus for testing applications, said apparatus including:

a data-processing machine including a memory and a processor operably coupled to said memory, said data-processing machine having been configured to implement a data-driven testing framework that includes a data engineering module, a computational environment manager, and a result analysis module,

wherein said data engineering module is configured to create engineered test data based at least in part on an application to be tested, including

 using data-lineage to track derivation of at least one derived variable through one or more transformations among at least two datasets associated with the application to be tested, where each of the at least two datasets includes a plurality of records, each record including one or more fields,

 identifying at least one input variable from which said derived variable is derived, and

 including in said engineered test data a first value for said input variable that corresponds to a second value for said derived variable, wherein said first value is obtained from a field of at least one of the at least two data sets,

wherein said computational environment manager is configured to control a computational environment in which said application is to operate on said engineered test data, wherein said input variable and said derived variable are accessed within said computational environment, and

wherein said result analysis module is configured to compare an output resulting from said engineered test data being operated upon by said application with an expected output.
2. The apparatus of claim 1,

wherein said data engineering module is configured to extract a subset of production data,

wherein said subset is selected to achieve a specified code coverage, and wherein said engineered test data includes said subset of said production data.

3. The apparatus of claim 1, wherein said data engineering module includes a data still for generating distilled data from production data.
4. The apparatus of claim 3, wherein said data engineering module includes a data enhancer for receiving distilled data from said data still and enhancing said distilled data.
5. The apparatus of claim 1, wherein said data engineering module is configured to generate data based at least in part on said application to be tested, wherein said generated data is selected to achieve specified code coverage, and wherein said engineered test data includes said generated data.
6. The apparatus of claim 1, wherein said data engineering module further includes a positive-data manufacturer for generating positive data.
7. The apparatus of claim 1, wherein said data engineering module is configured to generate data based at least in part on said application to be tested, and wherein said data is absent from production data.
8. The apparatus of claim 1, wherein said data engineering module further includes a negative-data manufacturer for generating negative data.
9. The apparatus of claim 1, wherein said data engineering module includes an integrity checker for determining referential integrity between different tables of said engineered test data, wherein said engineered test data includes a plurality of tables, each table including a plurality of records, and each record including one or more fields.
10. The apparatus of claim 1, wherein said data engineering module is further configured to detect errors in referential integrity.
11. The apparatus of claim 1, wherein said data engineering module includes a re-

referencer for correcting a loss of referential integrity in data prior to outputting said data as engineered test data.

12. The apparatus of claim 1, wherein said data engineering module is further configured to correct a loss of referential integrity in data.

13. The apparatus of claim 1, wherein said data engineering module includes an inspection unit for receiving said engineered test data and enabling a user to at least one of view said engineered test data and profile said engineered test data.

14. The apparatus of claim 1, wherein said data engineering module includes a data-inspection unit for receiving said engineered test data and enabling a user to view said engineered test data.

15. The apparatus of claim 1, wherein said data engineering module includes a profiler for receiving said engineered test data and enabling a user to profile said engineered test data.

16. The apparatus of claim 1, wherein said data engineering module is further configured to enable a user to profile said engineered test data.

17. The apparatus of claim 1, wherein said data engineering module is further configured to enable a user to view said engineered test data.

18. The apparatus of claim 1, wherein said data engineering module includes plural means for generating engineered test data, and wherein a particular means for generating engineered test data is generated based at least in part on information concerning said application to be tested.

19. The apparatus of claim 1, wherein said data engineering module includes a data enhancer, a data still, a negative-data manufacturer, and a positive-data manufacturer, each of which is configured to provide data that forms a basis for said engineered test data.

20. The apparatus of claim 1, wherein said data engineering module includes a logic extractor configured to identify logical functions within said application to be

tested that are to be tested and provides those logical functions to a data still.

21. The apparatus of claim 1, wherein said data engineering module is further configured to identify logical functions within said application to be tested that are to be tested and provides those logical functions to be used as a basis for obtaining a subset of production data.

22. The apparatus of claim 1, wherein said computational environment manager includes means for automatically setting up and taking down a computational environment in which said application is to be tested.

23. The apparatus of claim 1, wherein said computational environment manager includes an environmental-transition machine, wherein said environmental-transition machine is configured to identify a source of said engineered test data, and wherein said environmental-transition machine is further configured to identify a target in which to place data that results from processing of said engineered test data by said application to be tested.

24. The apparatus of claim 23, wherein said environmental-transition machine is further configured to copy engineered test data from a first repository to said source.

25. The apparatus of claim 24, wherein said environmental-transition machine is further configured to copy engineered test data from said target to a second repository.

26. The apparatus of claim 1, wherein said computational environment manager includes an environmental-backup machine, and a restoration machine, wherein said environmental-backup machine is configured for backing up a first environment prior to transforming said first environment into a second environment, wherein said restoration machine is configured for replacing said second environment with said first environment, and wherein said second environment is an environment in which testing of said application to be tested is to take place.

27. The apparatus of claim 1, wherein said computational environment manager includes an executioner, wherein said executioner is configured to cause execution

of said application to be tested.

28. The apparatus of claim 27, wherein said executioner is configured to automatically execute a script when causing execution of said application.

29. The apparatus of claim 1, wherein said computational environment manager includes an environmental-transition machine, an environmental-backup machine, a restoration machine, and an executioner, wherein said environmental-transition machine is configured to identify a source of said engineered test data, wherein said environmental-transition machine is further configured to identify a target in which to place data that results from processing of said engineered test data by said application to be tested, wherein said environmental-backup machine is configured for backing up a first environment prior to transforming said first environment into a second environment, wherein said restoration machine is configured for replacing said second environment with said first environment, wherein said second environment is an environment in which testing of said application to be tested is to take place, and wherein said executioner is configured to cause execution of said application to be tested.

30. The apparatus of claim 1, wherein said data engineering module is configured to extract a subset of existing data and to identify logical functions of the application to be tested, wherein said data engineering module is further configured to augment said subset of existing data, thereby generating augmented data, wherein said engineered test data includes said augmented data, and wherein said augmented data includes one or more fields added to one or more records of said subset of existing data based on the identified logical functions of the application to be tested.

31. The apparatus of claim 30, wherein said augmented data is selected to achieve specified code coverage.

32. The apparatus of claim 30, wherein said data engineering module is further configured to generate data to fill the added one or more fields based on one or more supplied rules.

33. The apparatus of claim 1, wherein said data engineering module is configured to create engineered test data by distillation of existing data, wherein said

engineered test data has a higher logic concentration than said existing data.

34. A method for processing data in a computing system, said method including:
- testing applications, wherein testing applications includes receiving information representative of an application to be tested over one of an input device and a port of a data processing system, and processing said received information, wherein processing said received information includes
- creating engineered test data based at least in part on said information, including
- using data-lineage to track derivation of at least one derived variable through one or more transformations among at least two datasets associated with the application to be tested, where each of the at least two datasets includes a plurality of records, each record including one or more fields,
 - identifying at least one input variable from which said derived variable is derived, and
- including in said engineered test data a first value for said input variable that corresponds to a second value for said derived variable, wherein said first value is obtained from a field of at least one of the at least two data sets,
- controlling a computational environment in which said application is to operate on said engineered test data, wherein said input variable and said derived variable are accessed within said computational environment,
- comparing an output resulting from said engineered test data being operated upon by said application with an expected output, and
- outputting a result indicative of said comparison.

35. The method of claim 34,
- wherein creating engineered test data includes extracting a subset of production data,
- wherein said subset is selected to achieve a specified code coverage, and

wherein said engineered test data includes said subset of said production data.

36. The method of claim 34, wherein creating engineered test data includes determining referential integrity between different tables of said engineered test data, wherein said engineered test data includes a plurality of tables, each table including a plurality of records, and each record including one or more fields.

37. The method of claim 34, wherein creating engineered test data includes identifying logical functions within said application to be tested that are to be tested and providing those logical functions to be used as a basis for obtaining a subset of production data.

38. The method of claim 34, wherein creating engineered test data includes extracting a subset of existing data, identifying logical functions of the application to be tested, and augmenting said subset of existing data, thereby generating augmented data, wherein said engineered test data includes said augmented data, and wherein said augmented data includes one or more fields added to one or more records of said subset of existing data based on the identified logical functions of the application to be tested.

39. The method of claim 38, wherein creating engineered test data further includes generating data to fill the added one or more fields based on one or more supplied rules.

40. The method of claim 38, wherein said augmented data is selected to achieve specified code coverage.

41. The method of claim 34, wherein creating engineered test data includes creating engineered test data by distillation of existing data, wherein said engineered test data has a higher logic concentration than said existing data.

42. A computing system for testing applications, said computing system including:
means for remembering information, and

means for processing information, wherein said processing includes:

receiving information over at least one of an input device and a port of a data processing system, wherein said information is representative of an application that is to be tested;

creating engineered test data based at least in part on said application that is to be tested, including

- using data-lineage to track derivation of at least one derived variable through one or more transformations among at least two datasets associated with the application to be tested, where each of the at least two datasets includes a plurality of records, each record including one or more fields,

- identifying at least one input variable from which said derived variable is derived, and

- including in said engineered test data a first value for said input variable that corresponds to a second value for said derived variable, wherein said first value is obtained from a field of at least one of the at least two data sets;

managing a computational environment in which said application is to operate on said engineered test data, wherein said input variable and said derived variable are accessed within said computational environment; and

comparing an output resulting from said engineered test data being operated upon by said application and an expected output.

43. A non-transitory computer-readable medium storing software for managing testing of applications, said software including instructions for causing a computing system to execute processing steps that include:

- creating engineered test data based at least in part on an application to be tested, including

- using data-lineage to track derivation of at least one derived variable through one or more transformations among at least two datasets associated with the application to be tested, where each of the at least

two datasets includes a plurality of records, each record including one or more fields,

identifying at least one input variable from which said derived variable is derived, and

including in said engineered test data a first value for said input variable that corresponds to a second value for said derived variable, wherein said first value is obtained from a field of at least one of the at least two data sets;

controlling a computational environment in which said application is to operate on said engineered test data, wherein said input variable and said derived variable are accessed within said computational environment;

comparing an output resulting from said engineered test data being operated upon by said application with an expected output; and

outputting an analysis of said comparison.

44. The non-transitory computer-readable medium of claim 43,

wherein creating engineered test data includes extracting a subset of production data,

wherein said subset is selected to achieve a specified code coverage, and

wherein said engineered test data includes said subset of said production data.

45. The non-transitory computer-readable medium of claim 43, wherein creating engineered test data includes determining referential integrity between different tables of said engineered test data, wherein said engineered test data includes a plurality of tables, each table including a plurality of records, and each record including one or more fields.

46. The non-transitory computer-readable medium of claim 43, wherein creating engineered test data includes identifying logical functions within said application to be tested that are to be tested and providing those logical functions to be used as a basis for obtaining a subset of production data.

47. The non-transitory computer-readable medium of claim 43, wherein creating engineered test data includes extracting a subset of existing data, identifying logical functions of the application to be tested, and augmenting said subset of existing data, thereby generating augmented data, wherein said engineered test data includes said augmented data, and wherein said augmented data includes one or more fields added to one or more records of said subset of existing data based on the identified logical functions of the application to be tested.

48. The non-transitory computer-readable medium of claim 47, wherein creating engineered test data further includes generating data to fill the added one or more fields based on one or more supplied rules.

49. The non-transitory computer-readable medium of claim 47, wherein said augmented data is selected to achieve specified code coverage.

50. The non-transitory computer-readable medium of claim 43, wherein creating engineered test data includes creating engineered test data by distillation of existing data, wherein said engineered test data has a higher logic concentration than said existing data.