A Framework for Ensuring the Quality of a Big Data Service

Conference Paper · June 2016

DOI: 10.1109/SCC.2016.18

CITATIONS
7

READS
252

3 authors, including:

Junhua Ding
East Carolina University
72 PUBLICATIONS **507** CITATIONS
SEE PROFILE

Xin Hu
East Carolina University
107 PUBLICATIONS **2,268** CITATIONS
SEE PROFILE

A Framework for Ensuring the Quality of a Big Data Service

Junhua Ding[1,2]
[1,2]Dept. of Computer Science
East Carolina University
Greenville, NC, USA
dingj@ecu.edu

Dongmei Zhang[2]
[2]School of Computer Science
China University of Geosciences
Wuhan, China
jjilee@163.com

Xin-Hua Hu[3]
[3]Dept. of Physics
East Carolina University
Greenville, NC, USA
hux@ecu.edu

***Abstract*—During past several years, we have built an online big data service called CMA that includes a group of scientific modeling and analysis tools, machine learning algorithms and a large scale image database for biological cell classification and phenotyping study. Due to the complexity and "non-testable" of scientific software and machine learning algorithms, adequately verifying and validating big data services is a grand challenge. In this paper, we introduce a framework for ensuring the quality of big data services. The framework includes an iterative metamorphic testing technique for testing "non-testable" scientific software, and an experiment based approach with stratified 10-fold cross validation for validating machine learning algorithms. The effectiveness of the framework for ensuring the quality of big data services is demonstrated through verifying and validating the software and algorithms in CMA.**

Keywords-metamorphic testing; scientific software; machine learning; big data; diffraction image

I. INTRODUCTION

Big data has become the most important research subject recently. It includes large scale and complex data sets that traditional tools and algorithms cannot handle [1]. Its increasing volume, velocity and variety build grand challenges for general researchers to understand and create big data services. In addition, most big data services are intellectual properties protected by patterns, copyrights and other means [1]. Big data services that are available to general researchers are limited to open source data repositories such as digital libraries, Apache big data and public genome data. Comparing to industry level big data services, open source one is limited in the volume, velocity, variety and value. Big data research community needs new open source big data services that include growing data sets as well as software tools and data analytics algorithms to process them. We have been studying cell assay and classification for many years and building a big data service called Cell Morphology Assay (CMA) for modeling and analyzing 3D cell morphology and mining morphology patterns extracted from diffraction images of biology cells. Study of 3D morphology can provide rich information about cells that is essential for cell analysis and classification. Diffraction images of single cells are acquired using a diffraction imaging flow cytometer to quantify and profile 3D morphology of cells [15]. CMA tools can rapidly analyze large amount of diffraction images and obtain texture parameters in real time. Through these parameters, one can use various machine learning algorithms such as Support Vector Machine (SVM) [35] to optimize and identify a set of parameters to perform cell assay according to 3D morphology without the need to reconstruct the structures from the diffraction images. This new approach can thus provide an innovative approach for rapid assay of single cells without the need to stain them with fluorescent reagents. Another goal of CMA project is to provide researchers a significant source of big data and tools to conduct big data research. CMA adopts big data techniques to implement data management, analysis, discovery, applications and sharing into the development of morphology based cell analysis tools. It includes a group of scientific software tools for modeling, analyzing and producing image data, machine learning algorithms for feature selection and cell classifications, and database for managing the big data. CMA is classified as a big data service due to its large volume, fast growing, variety of data, and big potential values.

The most important component in CMA is its scientific software for data modeling and analysis. Scientific software includes a large computational component for supporting scientific investigation and decision making [16]. Due to the complexity of scientific software, many of them are "non-testable". It is infeasible to test a "non testable" program using regular software testing techniques due to the oracle problem, which means the absence of test oracles for testing the program. Metamorphic testing [3][25] as a novel software testing technique is a promising approach for solving oracle problems. It creates tests according to metamorphic relationship (MR) and verifies the predictable relations among the outputs of the related tests. However, the application of metamorphic testing to large scale of scientific software is rare because the identification of MRs for adequately testing complex scientific software is infeasible [16]. This paper introduces an iterative approach for developing MRs with guiding by the results from analyzing test execution and evaluation data.

The central component of CMA is a set of machine learning algorithms for analyzing diffraction images based on their textual patterns. Grey Level Co-occurrence Matrix (GLCM) features [7] are used to quantitatively characterize the textual patterns of diffraction images. However, the feature set calculated from GLCM often contains highly correlated features and creates difficulties in computation, learning and classification [10]. An approach called

978-1-5090-2628-9/16 $31.00 © 2016 IEEE
DOI 10.1109/SCC.2016.18

Extensive Feature Correlation Study (EFCS) was used in this research to select an optimal set based on the feature's formulation and numerical results on diffraction images. We also conducted an empirical study to find an optimal GLCM displacement *d* and image gray level for cell classifying using SVM. The validation of the classification is conducted with the stratified 10-Fold Cross Validation (10FCV) [27].

Although big data has become the most important research topic recently, work on the quality assurance of big data services is rare. Quality assurance of big data service includes measurement and validation of data sets, quality assurance of support environments, verification and validation of data analytics tools and applications. The framework introduced in this paper is on verification and validation of data analytics tools (e.g., the scientific software in CMA) in big data services. It includes an iterative metamorphic testing for testing scientific software, an experiment based approach for the feature selection and the validation of machine learning algorithms. The framework offers a whole solution for ensuring the quality of data analytics tools in big data services.

The rest of this paper is organized as follows: Section 2 describes big data service CMA. Section 3 discusses testing scientific software in CMA using iterative metamorphic testing. Section 4 discusses the feature selection and validation of machine learning algorithms. Section 5 describes related work and Section 6 concludes the paper.

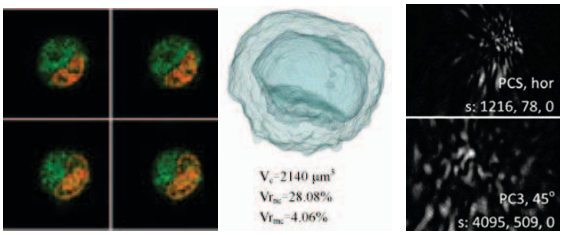
II. BIG DATA SERVICE CMA

A. CMA Architecture

The process of CMA consists of two parallel tasks: one is the cell classification using diffraction images of cells, which include imaging cells using a diffraction image flow cytometer, processing images for data selection, analyzing images for feature selection and calculating feature values, and cell classification with selected features using machine learning techniques. The other task is on modeling cell morphology to investigate the correlation between 3D structures of a cell and the textural pattern of the diffraction image of the cell. It includes imaging cells for confocal images, processing confocal image sections for constructing the 3D structure and producing the 3D morphological parameters with the 3D structure, calculating a diffraction image with the 3D cell morphological parameters, which are imported to a modeling program for simulating the light scattering of the cell. Through varying the 3D structures, one can correlate the variations with the image pattern changes in the corresponding calculated diffraction image. The two tasks in CMA are implemented with four major components: a database system for managing raw images and image processing data, the software for 3D structure reconstruction based on confocal image sections of cells, the software for calculating diffraction images based on cell 3D morphological parameters, and machine learning algorithms for image processing and cell classification.

B. 3D Structure Reconstruction Software

The software for 3D structure reconstruction is used for processing confocal image segments and calculating the 3D cell structure. The 3D cell structure is converted into 3D morphological parameters as an input to a light scattering modeling progrm called ADDA [30][34], which is an implementation of Discreate-Dipole Appoximation (DDA) method for calculating the diffraction image of a particle. The 3D structure is built based on confocal image sections that are acquired with a stained cell translated to different z-positions using a confocal microscope. Each image represents a section of the cell structure with very short focal depth (i.e. 0.5 *um*) along the z-axis. Individual nucleus, cytoplasm, and mitochondria stained with different fluorescent dyes are segmented from the image background outside the cell using multiple pattern recognition and image segmentation algorithms based on the pixel histogram and morphological analysis. Then the contours of segmented organelles between neighboring slices, after interpolation of additional slices along the z-axis to create cubic voxels, are connected for 3D reconstruction and voxel based calculations of morphology parameters such as size, shape and volume. Four confocal image sections of a cell is shown in Fig. 1 (a), snd (b) is a sectional view of the 3D strucure of the cell.



(a) (b) (c)

Fig. 1. (a) Confocal images, (b) 3D structure, and (c) diffraction images.

C. Software for Calculating Diffraction Images

ADDA is used to generate cross-polarized diffraction images using realistic 3D structures reconstructed from confocal images of cells [23][30]. As a general implementation of DDA, ADDA can be used for studying light scattering of many different particles from interstellar dusts to biological cells. The general input parameters of ADDA define the optical and geometry properties of a scatterer/particle including the shape, size, refractive index of each voxel, orientation of the scatterer to the incident light beam, definition of incident light beam, and many others. ADDA can be configurated for producing different outputs for different applications. In our study, we collect the Muller matrix to produce diffraction images using ray-tracing technique [24]. Fig. 1(c) shows two calculated diffraction images generated from ADDA simulations with different configurations.

D. Feature Selection and Cell Classification Algorithms

The software for feature selection and cell classification includes a tool for calculating GLCM of diffraction images, selecting optimal GLCM features for cell classification and then classifying cells using SVM. In order to fast calculate GLCM matrix and features, we developed a GPU based concurrent program and conducted an experimental study to select optimal GLCM features. The selected features are

used for cell classification based on diffraction images using SVM, and the effectiveness of the feature selection and cell classification is validated using 10FCV with 600 diffraction images. In 10FCV, the data is equally split into 10 groups where each group is held out in turn and the learning scheme is trained on the remaining nine-tenths; then its error rate is calculated on the holdout set (one-tenth used for testing). This learning procedure is repeated for a total of 10 times so that in the end, every instance has been used exactly once for testing. Finally, the 10 error estimates are averaged to yield an overall error estimation. The final goal of the study is to develop a supervised learning tool within the framework of SVM method for cell classification by the identification of optimal features to yield effective markers for fast cell classification based on diffraction images.

III. METAMORPHIC TESTING OF SCIENTIFIC SOFTWARE

Given an input to the 3D reconstruction program, it is difficult to check the consistency between a reconstructed structure and the morphology of the real cell. Given an arbitrary input to ADDA, it is hard to know the correctness of the output. They are both typical "non-testable" software due to absence of test oracles. Therefore, an iterative approach for developing MRs for serving the test oracles is proposed to test the software.

A. Iterative Metamorphic Testing

The iterative metamorphic testing consists of three major steps: development of initial MRs and tests, test evaluation, and refinement of MRs. (1) *Develop initial MRs and tests*: Based on domain knowledge of the Software Under Test (SUT) and general framework of metamorphic testing [22], one can develop a set of initial MRs. The initial test inputs are produced using general test generation approaches such as combinatorial testing, random testing and category-choice framework, and then each initial test input is converted into another set of test inputs according to an MR. The initial test inputs together with the converted one form a test of the MR. (2) *Evaluate MRs and tests*: We evaluate the testing with program coverage criteria, mutation testing, and mutated tests. Mutated tests are tests whose outputs violate an MR to ensure an MR can differentiate positive tests from the negative one. In mutation testing, everu mutant should be killed by an MR or weakly killed by a test. A mutant is weakly killed when the output of the mutated program is different to the original one. (3) *Refine MRs*. If the selected program coverage criteria cannot be adequately covered, or mutants cannot be killed or weakly killed by existing tests or by simply adding new tests, then new MRs should be developed or some existing MRs should be refined. If a mutated test passes the MR under test, then the MR should be refined. Analyzing existing software engineering data with advanced techniques such as machine learning algorithms is a potential approach for developing and refining MRs [17].

B. Testing the 3D Structure Reconstruction Software

The most difficult part in the software is on building the 3D structures of mitochondria. Each confocal image section may include many mitochondria that are so close to each other that two mitochondria in two neighboring sections could be incorrectly connected. The incorrect connection will result a wrong 3D structure. Metamorphic testing is used for testing the program.

1) Development of initial MRs and tests: Fig. 1 shows a sample input and output of the program, where (a) is partial of the input that consists of a complete set of confocal image sections of a cell, and (b) is a sectional view of the 3D reconstructed cell. Five MRs are first identified for checking the reconstruction of mitochondria [6]:

a) MR1: Inclusive/Exclusive. If an artificial mitochondrion is added/removed to/from a stack of original confocal image sections of a cell, the new added one should be recognized and included/excluded in/from the reconstructed structure, the total number of the mitochondria should be increased/decreased by one, and the calculated volume of the mitochondria in the cell should increase/decreased correspondingly.

b) MR2: Multiplicative. If the size of a mitochondrion in the original image sections is increased with a small percentage. The total number of the mitochondria should be kept as the same, and the volume of mitochondria is expected to increase.

c) MR3: Lengths. There is a gap between image sections. Therefore, a small mitochondrion may only appear in one section and a large one can appear in multiple sections. An artificial mitochondrion can be added to only one section or multiple sections. The mitochondrion shall be constructed based on the modified image sections. The new added mitochondrion is expected to appear in the 3D structure output along with the original one, and the volume of mitochondria shall increase as well.

d) MR4: Shapes. Mitochondria have different shapes and they determine the shapes of the 3D structure of the reconstructed mitochondria. Artificial mitochondria with different shapes are added to the confocal images to check whether the 3D structures of these mitochondria can be constructed as expected, and the 3D structure of the original one should not be changed.

e) MR5: Locations. The program processes mitochondria that are close to the nuclear differently to the one that is close to the cell boundary. Artificial mitochondria are added to different locations in the image, such as the location where is close to the nuclear or where is close to the cell boundary. The new added mitochondria should be recognized, and the 3D structure of the added mitochondria should be reconstructed as expected and the 3D structures of other original mitochondria should not be affected.

Tests are generated through transforming source tests (i.e., a stack of original image sections) according to MRs. Source tests and transformed one are paired together for testing their related MR, which are executed one by one and their outputs are compared to decide whether the test passes the MR. In order to increase the test coverage, it is necessary to create source tests using test generation techniques. Combinatorial testing technique is used for producing the initial source tests in this program. We

divided the MRs into two categories: {MR1, MR2} and {MR3, MR4, MR5}, and then create pair-wise tests for the two categories. For example, we create one test that is adding a round shape (i.e., MR4) artificial mitochondrion into one image section (i.e., MR1, MR3) closing to nuclear (i.e., MR5). The initial source tests have full pair-wise combinations, and then paired tests are created by converting the source tests with each MR.

2) Evaluation of MRs and tests: The function coverage, statement coverage, definition-use pair coverage, and mutation coverage were selected for evaluating the test quality. The code for checking the program coverage was manually instrumented into the source code. All tests satisfied the 5 MRs as designed, and the program coverage criteria were 100% covered by the tests.

Instead of conducting systematic mutation testing, only 18 mutants were manually instrumented to the program. The mutants that caused crash or exception of the execution were excluded. All mutants were killed or weakly killed. There was one mutant that is not killed through checking the MRs and the outputs, but it was detected by checking the test coverage information. The mutant was added to the function that is designed for smoothing the contour lines of segmented components in neighbor image sections. The smoothing function is applied to all connections. However, the mutant changed the program so that the function is only called when the difference between the two contour lines is over a threshold. The smoothed connection and non-smoothed one isn't easily detected even in a visualized 3D structure. The mutant does not change the number or volume of mitochondria. But we found only 40% of tests covered the function instead of 100% as expected.

Mutated tests have to be created with changing multiple test input parameters at the same time to create a test that would violate an MR. For example, for MR *inclusive*, one artificial mitochondrion is added, but one existing mitochondrion has to be removed at the same time. The test result of the new test should not satisfy MR inclusive even new mitochondrion is added.

3) Refinement of MRs: MRs *Inclusive*/*Exclusive* and *Multiplicative* can be further refined to know what's the exact change of the mitochondria's volume when a mitochondrion is added or removed. For example, if an artificial mitochondrion is added to the original confocal image sections, the mitochondrion volume can be calculated using Matlab based on its 3D model. Then the volume difference between the original images and the updated ones should be the one just calculated in the Matlab. If a result is different, something must be wrong. The refined MR is more effective to find subtle errors such as the one shown in Fig. 2, where A and B are supposed to be connected, but new added C causes C and B be connected instead. The number of mitochondria won't be affected, the volume might be also increased, but the volume difference will not be the one as expected.

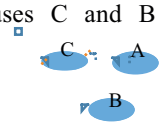


Fig. 2. one error

C. Testing ADDA

ADDA have been extensively tested with special cases [30]. For example, Mie theory was used for computing the light scattering of spheres to compare ADDA's results [23]. But Mie theory cannot calculate a scatterer in an irregular shape, and it is also difficult to find a different implementation of DDA or different method as a reference system to test ADDA. Therefore, iterative metamorphic testing is used to test ADDA.

1) Development of Initial MRs and Tests. It is difficult to find an MR directly on the inputs and the outputs of ADDA since an output may include thousands of closely related datum items. Therefore, the MRs are defined on the properties of the diffraction images that are generated from ADDA outputs. The textual pattern of the diffraction image is an example property for defining MRs. Each input parameter is a candidate for defining MRs. We define initial MRs based on the *shape*, *size*, *orientation*, and *refractive index* of a scatterer, and check the correlation between each parameter and the textual pattern of the diffraction image of a scatterer.

a) MR6: *when the size of a scatterer is changed, the textual pattern of the diffraction image is changed*. It is first defined on the relation between the size of a sphere scatterer and its textural pattern of the diffraction image generated from the ADDA simulation. We know that when the size of the sphere is changed, the textural pattern of the diffraction image is changed. Furthermore, we know there is a correlation between the size and the texture pattern for any scatterer, such as when the size of the sphere becomes larger, the texture bright lines become slimmer. For other shapes of scatterers, the change of its size also affects the textural pattern of its diffraction image.

ADDA can handle a large range of sizes of scatterers from nanometer to the size that computing resource can handle it. But for the simulation of the light scattering of blood cells, we set the range of scatterer sizes from 4μm to 16μm, which is the size range of blood cells. We tested size 4μm, 5μm, 7μm, 9μm 11μm, 12μm, and 16μm for regular shape scatterers such as sphere scatterers and ellipsoid scatterers. The irregular shape scatterers are constructed from real cells, and their sizes are within the range from 4μm to 16μm. All test results pass MR6.

b) MR7: when the shape of a scatterer is changed, the textual pattern of the diffraction image is changed. For example, the textual pattern of the diffraction image of a sphere scatterer is more regular than the one of an ellipsoid scatterer. For an irregular shape scatterer, the change of its shape also affects the textural pattern of its diffraction image. However, the exact correlation among them is unknown.

c) MR8: *when the orientation of an irregular scatterer is changed, the textual pattern of the diffraction image is changed. For a sphere shape scatterer, the textual pattern should be the same at all orientations.* For an irregular scatterer, first a reference orientation is set as <0, 0, 0> in the 3D coordinate system, and then the orientation is updated related to the reference. The number of valid orientations for any scatterer is infinite; therefore, we only

check some boundary orientations such as <0, 0, 0>, <0, 90, 90>, and <270, 90, 0>, and some in the middle.

d) MR9: *when the refractive index value of a scatterer is changed, the textual pattern of the diffraction image is changed*. The relation is held for both homogenous scatterers and heterogeneous scatterers. The homogeneous scatterer has a uniform refractive index for all voxels. The range of refractive index values of a blood cell is from 1.2 to 1.5, and each voxel of the cell 3D structure may have a different reflective index value. The modification of the index values of a cell is completed using a Matlab program to ensure the boundary index values are appropriately set. The homogenous scatterer such as a sphere scatterer is tested with the boundary index values from 1.01 to 1.20 and some values within the middle. A homogeneous scatterer normally has more strip lines in the diffraction image, and a heterogeneous has more speckles in the diffraction image.

e) MR10: when the morphology structure of a cell scatterer is changed, the textual pattern of the diffraction image is changed. However, the precise relation between the morphology of a cell scatterer and its diffraction image is a research topic we are studying with ADDA. Through changing the morphology of a 3D structure such as resizing the nuclear of the cell, change the refractive index values of cell organelle [6], one can produce different ADDA simulation results and check the correction between the textual patterns of the diffraction images and the morphology change. We had conducted preliminary experiment and the result showed that a relation existed between GLCM features and the 3D morphological structures of cells [27].

In order to create tests that can cover as many cases as possible, we use the combinatorial technique to create tests. For example, the four input parameters used for testing ADDA are the *scatterer size*, *shape*, *refractive index*, and *orientation*. The possible values of size are {3μm, 5μm, …, 16μm}, shapes are {sphere, ellipsoid, bi-sphere, prism, egg, cylinder, capsule, box, coated, cell1, cell2, …}, orientations are {<0, 0, 0>, <10, 90, 0>, <270, 0, 0> , …}, and refractive index values are {1.0, … 1.5}. Using the pair-wise technique, one can create many tests, and then select the valid tests as the source tests to create tests for each MR. Fig.3 shows a group of diffraction images calculated with different tests from MR6 to MR10. The results pass all MRs.

2) Evaluation of MRs and Tests. We created and tested several hundred tests for MR6 to MR10. All tests passed the MRs and they covered 100% statements of ADDA. We conducted partial mutation testing to check the effectiveness of the testing. The mutation testing was applied only to one module of ADDA program that is related to the most critical computing task. Instead of testing the program with full mutants created with mutation testing tools, only a few mutants were instrumented to the code manually. We checked the consistency between the outputs of the mutated program and the original one. In the case study, Absolute Value Insertion (ABS) and Relational

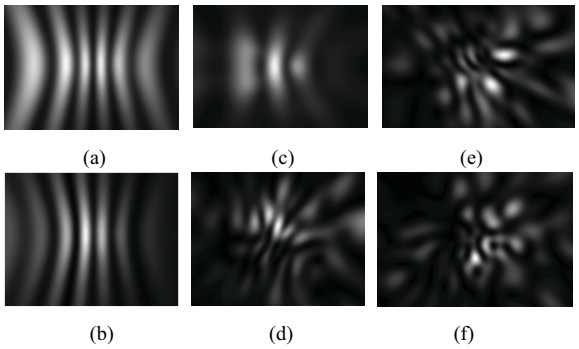


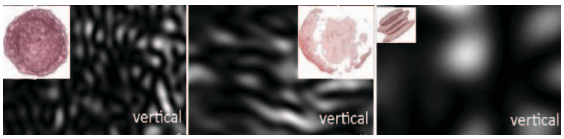
Fig. 3. The change of textual patterns of diffraction images calculated with different parameter values, (a) and (b) are spheres with different sizes, (c) is an ellipsoid, (d) and (e) are cells in different orientations, and (e) and (f) are different cells.

Operator Replacement (ROR) are the only two mutation operators were used for creating mutants [28]. Among total 20 mutants (10 ABS mutants, and 10 ROR mutants) we created for testing ADDA, 17 of them were killed by crashing of the program or exception handing. The other 3 mutants were killed by the MRs since the outputs didn't generate any textual pattern. Then we randomly created some mutants in the module, and we found mutants are easily killed by simple tests in the software. The possible reason could be the complexity of the software and many statements are linked to some precise scientific computations so that a slight change can be propagated and cause a catastrophic error in the calculation.

We found it is difficult to create mutated tests for testing MR6 to MR10. Given a scatterer, changing the shape, size, orientation or refractive index would change the textual pattern of the diffraction image. There is no way to change two parameters together so that they would mitigate the changes each other to create a scatterer with the same textual pattern. In other words, MR6 to MR10 are easily to be passed by almost any test. Therefore, the 5 initial MRs are not strong enough to test ADDA. Additional MRs are needed or the current MRs should be refined.

3) Refinement of MRs. Since scatterers in regular shapes such as sphere have been extensively tested [23], we are only interested in scatterers in irregular shapes such as cells. Here we discuss how to develop new MRs based on initial test results. We first measure the textual pattern of each diffraction image with GLCM features, and then use machine learning algorithms to find patterns for defining MRs. We developed two MRs. The first one is to check the consistency of textual patterns between the calculated diffraction images and the corresponding measured diffraction images. Then we develop the other MR based on cell classification, which classifies different types of cells based on ADDA calculated diffraction images.

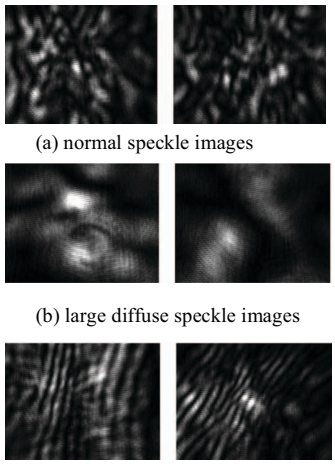
f) MR11: The textual pattern of a calculated diffraction image should be consistent to the textual pattern of the corresponding measured diffraction image. Based on domain knowledge and experimental study, we know that



(a) (b) (c)

Fig. 4. Calculated diffraction images: (a) a viable cell structure, (b) a large cell debris with most of cellular organelles, (c) aggregated small ellipsoids [31].

the diffraction image of a normal cell has small speckle patterns, and the diffraction image of a small agreegated paticle (about 1/5 of the size of a normal cell) have large diffuse speckle patterns, and a fragmened cell such as a cell shell has strip patterns in the diffraction image. The calculated diffraction images of the three types of the scatteres: normal cells, fragmented cells and agrregated particles are shown in Fig. 4, and the measured diffraction images of the same types of scatterers are shown in Fig. 5. Visually inspecting the image, one can find the textual patterns of calcultaed diffraction images and measured one are consistent. From this perspective, we know ADDA correctly simulates the light scattering of the three types of scatterers. Through testing different types of scatterers referring to the measured images, more genereal conclusion could be made.



(a) normal speckle images

(b) large diffuse speckle images

(c) strip images

Fig. 5, Three types of measured diffraction images.

g) MR12: Different types of cells can be classified by their diffraction images.

We first check whether different shapes of scatterers can be be classified by their ADDA calculdated diffraction imagegs. We first produced 100 diffraction images per shape of scatterers using ADDA. The 100 images of the same shape scatterers were generated with different sizes, refractive index values, orientations. Total 300 images were produced for three shapes: sphere, bi-shpere and ellipsoid. Each image was processed for GLCM features and labelled with the shape of the scatterer. SVM was trained with the feature vectors and classification results were valided with 10FCV. The results show SVM is effective for classifying the shape of the scatterer based on diffraction images. We now define MRs based on cell classification with calculated diffraction iamges.

In our previous work, 600 diffraction imges have been acquired using the diffraction iamging flow cytometer. The 600 images are belonged to 6 different types of cells. An SVM classifier based on selected GLCM features is trained on the images for each type of cells. 10FCV showed the SVM classifier can classify the cell types with more than 90% accuracy. Based on the result, we cehck whether the calculated diffraction images of the same type of cells can be trained and classified with the similar accuracy. We calculated 2 cells for each type of cells and simualted each cell with 25 different orientations, which result 50 diffraction images for each type of cells, and total 150 diffraction images were produced. Our preliminary reuslts show the ADDA diffraction images can be used for classifying cell types. If more experiments with many different types of cells will produce the same result, we would conclude ADDA is well implemented for simulate the light scattering of cells.

ADDA is the complex scientific software that is difficult to be adequately tested due to the challenge for developing highly effective MRs. Mining testing results is a promising solution for developing MRs. The empirical study has illustrated the iterative process for building MRs using machine learning approach and demonstrated its effectiveness for adequately testing ADDA. The key solution for adequately testing scientific software is to produce large amount of data such as program execution and program analysis data, and then mining out patterns for defining test oracles.

IV. Validation of Machine Learning Algorithms

The validation of machine learning algorithms in CMA includes two tasks: (1). To find optimal GLCM features for the cell classification. (2). To validate the SVM based cell classification. We conducted an empirical study to answer these questions. Automated cell classification based on GLCM features was developed before [7][29][8]. However, the GLCM feature set often contains highly correlated features and creates difficulties like increasing in computation time, lowering learning speed, and in some cases decreasing classification accuracy [27]. Thus, it is important to remove redundant features from the feature set. Feature selection is the process of selecting a subset of total features that are highly correlated with the subject or class and yet uncorrelated with each other [27]. SVM is used to classify cell types based on GLCM features. The goal of SVM is to build a model based on training data where each instance has a target value (or class label) with a set of attributes (or features) and predict the target values of the test data given only its attributes [13]. SVM performs binary classification in general; however, several SVM classifiers can be combined to do multiclass classification by comparing ‘one against the rest’ or ‘one against one’. In this research, LIBSVM [35], an open source library for SVM, was used to perform classification of diffraction images based on GLCM features.

1) Feature Selection: We used EFCS to select optimized GLCM features [27]. EFCS selects uncorrelated features by analyzing the trend of all features. It first lists the feature vectors that consists of all feature values for each diffraction image. Then each feature value is normalized to values between 0 and 1. Third, polynomial regression used to plot the data trend for every feature of all images. Finally, all features are plotted on a single graph to analyze the correlation between features.

In the experiment, four GLCMs were calculated using orientation at 0^o, 45^o, 90^o, 135^o for each image, respectively. Then the average of all 4 orientations was calculated for every single feature. We calculated the 17 feature values of each image for all 600 images using different distance *d* at 1, 2, 4, 8, 16 and 32, and gray levels at 8, 16, 32, 64, 128 and 256, which yielded 36 combinations of each image. GLCM features are categorized into three groups namely *Contrast*, *Uniformity*, and *Correlation* [11]. Every feature is categorized into its respective group. All features from each group are plotted on a single graph for all images with the same displacement and gray level. Finally, uncorrelated features are obtained from each group for all 108 (*i.e.* 3 groups x 36 combinations) graphs by visual inspection. The nature of correlation between the features remained similar in all combinations. 8 features of the 20 features from three groups were selected for the classification using SVM.

2) Validation of Algorithms: The cell type of each cell is known in advance. The feature vector including the label of the cell type for each image is sent to SVM for training the algorithm. We trained the SVM for each type of cells with the feature vector matrix of the 100 labelled diffraction images, and then used 10FCV to check the accuracy of the SVM for classifying the diffraction images based on the selected GLCM features. Specifically, in this study, 90 images per type of cells were used for training and remaining 10 images were used for testing. This process is repeated for 10 folds so that in the end every image is used exactly once for testing. Using the 8 selected features, the SVM classification accuracy achieved was 91.16%, which is slight better than what was achieved using all the features [27]. Based on the result, we conclude that SVM with the selected features is an effective way for classifying cell types based on diffraction images.

The experimental results conclude 8 GLCM features in addition to the grey level 64 and distance 2 are the best for classifying diffraction images using SVM. The SVM classification based on GLCM features requires large amount of GLCM calculations such as we processed total 21,600 diffraction images for 4320,000 feature values for the feature selection experiment. It is strongly arguable that an optimal set of eight features are sufficient to quantify these differences based on the accuracy of the classification results, where the classification accuracy using the selected features is better than using all GLCM features.

V. Related Work

Quality assurance of big data service includes many tasks, which can be classified as follows: quality assurance of data sets, quality assurance of the data management and running systems, quality assurance of the data analytics tools, and quality assurance of the applications of big data. Our focus in this paper is on the quality assurance of the data analytics tools. Adequately testing scientific software is a grand challenge. One of the greatest challenges that occurs due to the characteristics of scientific software is the oracle problem [16]. However, it is difficult to find the traditional test oracles for testing many complex scientific software systems. Metamorphic testing is a potential technique to address the oracle problem though developing oracles with MRs [25]. Metamorphic testing was first proposed by Chen et al. [1] for testing "non-testable" systems. It has been applied to several domains such as bioinformatics, machine learning, and online services [20][19][21][32]. Metamorphic testing has been also used for testing scientific software. Mayer and Guderlei developed a group of MRs for testing image process programs [22]. Chen, Fend and Tse [5] have applied metamorphic testing for testing partial differential equations. Knewala, Bieman and Ben-Hur recently reported a result on the development of MRs for scientific software using machine learning approach integrated with data flow and control flow information [17]. The 3D reconstruction software and ADDA are much larger and more complex than other scientific software reported in metamorphic testing research.

Test generation is another important task in software testing, and it is even more difficult for scientific software due to the complex data types and large amount of input parameters [16]. The focus of existing test generation is on producing a set of individual tests, not a set of tests for testing a relation among their outputs. In this paper, we not only discussed how to produce adequate tests for testing big data services, but also discussed how to create mutated tests for checking the quality of MRs. Creating mutated tests with complex data types is difficult. We proposed an approach for iteratively developing tests guided by machine learning test data.

The evaluation of metamorphic testing is focused on the selection of MRs since the effectiveness of metamorphic testing is highly depended on the selection of MRs. Many researchers support to conduct metamorphic testing using as more as possible MRs [33]. However, many redundant MRs could be developed, and they couldn't improve the testing effectiveness. Therefore, rigorously evaluating MRs with test adequacy criteria is important for selecting good MRs. Chen, Huang, Tse and Zhou discussed an approach for selecting MRs that could be good at fault detection through studying several cases [3]. Their case studies indicate that domain knowledge alone is inadequate for finding good MRs. Good MRs should be selected considering the implementation of the software under test [3]. However, Mayer and Guderlei had a different conclusion. They claimed that good MRs should be those defined based on the semantics of the software testing [22]. Asrafi, Liu and Kuo pointed out good MRs are those that can make the multiple executions of the program as different as possible [1]. The results further confirm the importance of the evaluation of MRs with program based coverage criteria that is integrated in the iterative metamorphic testing.

VI. Summary and Future Work

In this paper, we introduced a framework for ensuring the quality of big data service CMA. An iterative metamorphic testing was used for testing scientific software, feature selection was conducted for improving the effectiveness and performance of classification, and cross validation was performed for evaluating the machine learning algorithms. The framework addressed the most important quality assurance issues in big data services. In

the future, our focus is on the validation of big data directly using machine learning techniques.

ACKNOWLEDGMENT

The authors would thank Sai Thati, Eric King at East Carolina University and Jun Zhang at Tianjin University for assistances of the experiments. This research is supported in part by grants CNS-1262933 and CNS-1560037 from the National Science Foundation.

REFERENCES

[1] M. Asrafi, H. Liu, and F-C. Kuo, "On testing effectiveness of metamorphic relations: A Case study,' in *Fifth Intl. Conference on Secure Software Integration and Reliability Improvement (SSIRI*), pp. 147-156, 2011.

[2] M. Chen, S. Mao, Y. Liu, "Big data: A survey". *Mobile Networks and Applications, 19*(2), pp. 171-209, 2014.

[3] T. Y. Chen, S. C. Cheung, and S. Yiu, "Metamorphic testing: a new approach for generating next test cases", *Tech. Rep. HKUST-CS98-01*, Dept. of Computer Science, Hong Kong Univ. of Science and Technology, 1998.

[4] T. Y. Chen, D. H. Huang, T. H. Tse, and Z. Q. Zhou, "Case studies on the selection of useful relations in metamorphic testing", In Proc. of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering, pp. 569-583, 2004.

[5] T.Y. Chen; J. Feng; T.H. Tse, "Metamorphic testing of programs on partial differential equations: a case study," in Proc. of 26th Annual Intl. Computer Software and Applications Conference (COMPSAC), pp.327-333, 2002

[6] J. Ding, T. Wu, J. Q. Lu, X. Hu, "Self-Checked Metamorphic Testing of an Image Processing Program", 4th *IEEE Intl. Conference on Security Software Integration and Reliability Improvement*, Singapore, June 9 – 11, 2010.

[7] K. Dong et al., "Label-free classification of cultured cells through diffraction imaging," *Biomed. Opt. Express 2(6),* pp.1717–1726, 2011.

[8] Y. Feng et al., "Polarization Imaging and Classification of Jurkat T and Ramos B Cells Using a Flow Cytometer," *Cytometry Part A, 85*, pp. 817-826, June 2014.

[9] V. Gottemukkula, S. Saripalle, R. Derakshani, and S. Pavan Tankasala, "A Texture-Based Method for Identificaiton of Retinal Vasculature," 2011 *Intl. Conf. Technologies for Homeland Security (HST)*, pp. 434- 439, Nov 2011.

[10] M. A Hall, "Correlation-based Feature Selection for Machine Learning," *doctoral dissertation*, The University of Waikato, Hamilton, Newzealand, 1999.

[11] R. Haralick, "On a Texture-Context Feature Extraction Algorithm for Remotely Sensed Imagery," Proceedings of the *IEEE Computer Society Conference on Decision and Control*, Gainesville, FL, pp. 650-657, Dec. 1971.

[12] R. M. Haralick, K. Shanmugan, and I. H. Dinstein, "Textural features for image classification", *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp.610 -621, 1973.

[13] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, "A Practical Guide to Support Vector Classification," *Technical Report*, National Taiwan University, 2003.

[14] P. W. Huang, Cheng-Hsiung Lee, and Phen-Lan Lin, "Support Vector Classification for Pathological Prostate Images Based on Texture Features of Multi-Categories," 2009 *IEEE Intl. Conf. on Systems, Man, and Cybernetics*, pp. 912-916, Oct. 2009.

[15] K. M. Jacobs, J. Q. Lu, and X. H. Hu, "Development of a diffraction imaging flow cytometer," *Opt. Lett.34 (19),* pp. 2985–2987, 2009.

[16] U. Kanewala, J. M. Bieman, "Testing scientific software: A systematic literature review", *Information and Software Technology*, Vol. 56, Issue 10, Oct. 2014, pp. 1219-1232, 2014.

[17] U. Kanewala, J. M. Bieman, A. Ben-Hur, "Predicting Metamorphic Relations for Testing Scientific Software: A Machine Learning Approach Using Graph Kernels", *Journal of Software Testing, Verification and Reliability*, Nov. 16, 2015.

[18] R. Krishnan and S. Radhakrishnan, "Focal and diffused liver disease classification from ultrasound images based on isocontour segmentation," *IET Image Process*., Vol. 9(4), pp. 261–270, 2015.

[19] V. Le, M. Afshari, and Z. Su. "Compiler validation via equivalence modulo inputs". In Proceedings of the 35th *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '14)*. ACM, New York, NY, USA, pp.216-226. 2014.

[20] H. Liu, F.-C. Kuo, D. Towey, T.Y. Chen, "How Effectively Does Metamorphic Testing Alleviate the Oracle Problem?" *IEEE Trans. on Software Engineering*, vol.40, no.1, pp.4,22, Jan. 2014.

[21] M. Lindvall, D. Ganesan, R. Árdal, and R. E. Wiegand. "Metamorphic model-based testing applied on NASA DAT: an experience report". *In Proc. of the 37th Intl. Conference on Software Engineering*, Vol. 2. pp. 129-138. 2015.

[22] J. Mayer, and R. Guderlei, "An empirical study on the selection of good metamorphic relations", *In proc of 30th COMPSAC*, pp. 475-484, 2006.

[23] M. Moran, "Correlating the morphological and light scattering properties of biological cells", *PhD dissertation*, Dept. of Physics, East Carolina University, 2013.

[24] R. Pan, Y. Feng, Y. Sa, J. Lu, K. Jacobs, and X. Hu, "Analysis of diffraction imaging in non-conjugate configurations," *Opt. Express* 22, 31568-31574. 2014.

[25] S. Segura; G. Fraser; A. Sanchez; A. Ruiz-Cortes, "A Survey on Metamorphic Testing," in *IEEE Trans. on Software Engineering* , vol.PP(no. 99), doi: 10.1109/TSE.2016.2532875. 2016.

[26] M.B. Subramanya, V. kumar, S. Mukherjee, and M. Saini, "Classification of normal and medical renal disease using B-mode ultrasound images," 2015 2nd *Intl. Conf. on Computing for Sustainable Global Development (INDIACom)*, pp.1914-1918, 2015.

[27] S. K. Thati, J. Ding, D. Zhang, and X. Hu, "Feature Selection and Analysis of Diffraction Images", the 4th *IEEE Intl. Workshop on Information Assurance*, Vancouver, Canada, August 3-5, 2015.

[28] W. E. Wong and A.P. Mathur, "Reducing the cost of mutation testing: an experimental study," *J. Systems and Software*, vol. 31, no.3, pp.185-196, Dec. 1995.

[29] Xu Yang et al., "A quantitative method for measurement of HL-60 cell apoptosis based on diffraction imaging flow cytometry technique," *Biomed Opt Express, 5(7):* pp. 2172–2183, Jul. 2014.

[30] M.A. Yurkin and A.G. Hoekstra, "User manual for the discrete dipole approximation code ADDA 1.3b4", http://a-dda.googlecode.com/svn/trunk/doc/manual.pdf (2014). Last accessed on March 12, 2016.

[31] J. Zhang, Y. Feng, M.S. Moran, J.Q. Lu, L.V. Yang, Y. Sa, N. Zhang, L. Dong, X. H. Hu, "Analysis of cellular objects through diffraction images acquired by flow cytometry", *Optics Express*, 21, 24819-24828, 2013.

[32] Z.Q. Zhou, S. Xiang, T.Y. Chen, "Metamorphic Testing for Software Quality Assessment: A Study of Search Engines", *IEEE Trans. on Software Engineering*, Issue 99, 2015.

[33] Z.Q. Zhou, W. K. Chan, T.H. Tse, and P. Hu, "Experimental study to compare the use of metamorphictesting and assertion checking," *Journal of Software*, vol. 20, no. 10, pp. 2637-2654, 2009.

[34] ADDA project, https://code.google.com/p/a-dda/, Last accessed on March 12, 2016.

[35] LIBSVM, https://www.csie.ntu.edu.tw/~cjlin/libsvm/, last accessed on March 12, 2016.