Modeling Clones Evolution in Open Source Systems
Through Chaos Theory

Doaa M. Shawky
Engineering Mathematics Dept.
Faculty of Engineering, Cairo University
Giza, Egypt
doaashawky@ieee.org

Ahmed F. Ali
Biomedical Engineering Dept.
Faculty of Engineering, Helwan University
Helwan, Egypt
ahmedfarag@ieee.org

Abstract—A code clone is a code fragment that is identical or similar to another according to a certain similarity definition. Usually, it is a result of certain programmer’s practices. Unjustified cloned codes can cause an increase in maintenance effort. In addition, they are sometimes a sign of poor design. This paper presents an approach for modeling clones evolution in open source systems. It adapts chaos theory for predicting clones in new versions of a software system. The number of clones in each version is identified and analyzed as a time series data. The existence of chaos is tested through the calculation of Lyapunov exponent and correlation dimension. Experimental results show that clones evolution in open source systems is a chaotic process. Thus, prediction in new versions can be done with high prediction accuracy using chaos theory.

Keywords—chaos theory; clones evolution; clones detection.

I. INTRODUCTION

Code clones are program segments which are similar to a certain degree of similarity. Usually inconsistency is introduced in the code, if one code segment is modified in correspondence to maintenance efforts, meanwhile its clones are not [1]. In addition, the existence of a large number of clones indicates design problems such as lack of encapsulation or abstraction [2]. For these reasons, much effort has been spent on the detection and removal of code clones from software systems [2].

In this paper, we propose an approach for modeling clones in successive versions of a system using chaos theory. Chaos is a property of some deterministic systems. A deterministic system is one in which future states depend on the current conditions. They can be modeled by dynamical systems. Historically the idea has been that all processes occurring in the universe are deterministic, and that if we knew enough of the rules governing the behavior of the universe and had measurements about its current state we could predict what would happen in the future [3]. These ideas have been applied with a great deal of success to many systems such as falling objects, tide prediction, and weather prediction, etc. However, long time prediction is not possible due to our incomplete knowledge of the system.

Given a set of subsequent versions of a software system, the number of clones in each version is detected. The number of clones in subsequent versions is used as a time series data. Then, a study of the time series dynamics is done through the use of chaos theory. Finally, new values of the time series is predicted and compared to the actual ones. The presented approach has been applied to two open source systems. Mean square error of the predicted values is computed to evaluate the prediction accuracy.

The rest of this paper is organized as follows. Section II of this paper introduces some definitions and concepts. Section III describes the analysis process in details, while Section IV presents the case studies and the obtained results. The related research in the area is summarized in Section V. Finally, Section VI draws conclusions, limitations, and outlines ideas for future work.

II. DEFINITION OF RELATED TERMS

A. Clones

Code clones are pieces of source code that are duplicated in several locations [4]. They are generated mainly by some common programmers’ practice such as copy and paste [5-6]. Clones are believed to be harmful, and if not removed, at least they should be detected. In [7, 8], the authors studied the effect of cloning on program correctness. Moreover, in [8], the harmfulness of clones is studied. The authors found out that cloned functions are subject to change more frequently than non-cloned ones. Usually, clones are classified into three types [9-10]:

- Type 1: is an exact copy without modifications (except for whitespace and comments).
- Type 2: is a syntactically identical copy; only variable, type, or function identifiers have been changed.
- Type 3: is a copy with further modifications; statements have been changed, added, or removed.

Sometimes, Type 1 clones are called exact copy clones. Meanwhile, Type 2 and 3 are referred to as near miss clones. However, there is no clear definition of the degree of
similarity that should be used in detecting near-miss clones [11-12].

B. Chaos Theory

Chaos refers to systems which are at an intermediate point between the completely predictable and the totally random ones. Examples of chaotic systems in nature include tornadoes, stock markets, turbulences, and weather [13]. Chaos theory is the one that deals with such systems. At the heart of chaos theory is the notion that complex systems can often be characterized by fairly simple mathematical equations [14]. Typically, an equation is determined which relates the value of a variable at certain time. This equation is then used iteratively to calculate the value at next time steps. The output of the former calculation is used as an input for the next period’s calculation.

Chaos theory is the one that deals with such systems. At the heart of chaos theory is the notion that complex systems can often be characterized by fairly simple mathematical equations [14]. Typically, an equation is determined which relates the value of a variable at certain time. This equation is then used iteratively to calculate the value at next time steps. The output of the former calculation is used as an input for the next period’s calculation.

Chaotic systems must have the following characteristics [14-16]:
- Non-linearity: nonlinear relationships between cause and effect are a necessary, but not sufficient, condition for chaotic systems.
- Feedback: Typically the feedback is represented in the process of iteratively characterizing or projecting a time series, where the results of previous calculations are the input for succeeding calculations.
- Sensitivity to initial conditions: Small initial errors and perturbations sometimes magnify to create major changes.
- Attractors: most complex systems exhibit what is called attractors which are the states or patterns the system eventually settles into. Some systems tend toward traditional fixed points, limit cycles, or strange attractors which are unrepeatable patterns.
- Long-term prediction is mostly impossible due to sensitivity to initial conditions.

The lack of long-term predictability in chaotic systems does not imply that short-term prediction is impossible. In counterpoint to purely random systems, chaotic systems can be predicted for a short interval into the future. Thus, a successful analysis of the time series of a chaotic system allows prediction or forecasting of the system’s behavior in the near future [13]. The first step in the analysis of a chaotic time series data was introduced in [17], in which state-space reconstruction of time series data was proposed for the first time. The mathematical justification of this approach was presented in [18]. In [18], the reconstructed state space is proved to be one-to-one equivalent to the original state space of the real-life system. The reconstruction of state space can be summarized as follows: Given s (1); a scalar function describing the system, sampled at time interval τ , and starting at some time t , the n sample can be represented as:

\[ s_n = s(t_0 + (n - 1) \tau), \quad n = 1, 2, \ldots \]  

A delay-coordinate reconstruction can be formed by plotting the time series versus one or more time-delayed version(s) of it. For a 2-dimensional reconstruction, we plot the delay vector \( y(n) = [s_n, s_{n+1}], n = L + 1, L + 2, \ldots \), where L is the lag or sampling delay, i.e., the difference between the adjacent components of the delay vector in number of samples. For a d-dimensional reconstruction, the delay vector, \( y(n) \) can be written as given by (2):

\[ y(n) = [s_n, s_{n-L}, \cdots, s_{n-(d-2)L}, s_{n-(d-1)L}] \]  

(2)

It was proved by Taken’s theory [18] that if d is large enough, the vector series \( y(n) \) reproduces many of the important dynamical characteristics of the original series. Thus, one does not need the original vector series in order to analyze many of the system properties of the data series. Specifically, if the dimension of the reconstructed space, d, is larger than twice m which is the number of active degrees of freedom, the equivalence of the spaces is guaranteed [18]. From a mathematical point of view, the selection of L has no effect on the embedding of a noise-free time series. However, in practical applications and for data contaminated with noise, a good choice of L has an important impact on the analysis [19]. If L is too small in comparison with the dynamic variation of the system, successive elements of the delay vectors are strongly correlated. If L is too large, successive elements are almost independent. In delay-coordinate reconstruction, the selection of time delay and dimension are the most important issues [19-21]. For the calculation of time lag, different approaches are proposed in the literature [20, 22]. Among them, the autocorrelation function and mutual information approach are the most general and common [23]. In our approach, we will use the mutual information method which can account for any nonlinear dynamical correlation in contrast to the use of the autocorrelation function [23].

The mutual information for sampling delay L can be defined as:

\[ I(L) = \sum P(s_i, s_{i+L}) \log \frac{P(s_i, s_{i+L})}{P(s_i)P(s_{i+L})} \]  

(3)

where \( P(s_i, s_{i+L}) \) is the probability that the signal has a value in the histogram representing the mutual information function. Informally, this function quantifies the information that we have about \( s_{i+1} \) given that we know \( s_i \). Usually, the sampling lag related to the first minimum of the mutual information function specifies the point where the information about \( s_{i+1} \) given knowledge of \( s_i \) or the redundancy has a local minimum.

Another important concept in chaos theory is the fractal dimension. Fractal dimensions quantify the self-similarity of a geometrical object [13]. One of the most used dimensions is the correlation dimension which is non integer for chaotic data indicating the presence of strange attractors. It can be calculated using correlation integral, which is an estimate of the probability that two points on the attractor lay less than a distance R from each other. Given the N values of the series and for fixed embedding dimension d and time lag L, we
calculate the percentage of points within a certain distance $R$ from one another, for increasing values of $R$, through the correlation integral ($C(R)$). As proposed in [3], $C(R)$ can be calculated as given by (4) for a fixed time lag $L$:

$$C(R) = \frac{1}{N(N-1)} \sum_{i=0}^{N-2} \sum_{j=1}^{N-1} \sum_{l=0}^{N-1} H(R - ||y(i) - y(j)||)$$  \hspace{1cm} (4)$$

Where $H(x)$ is the Heaviside step function with $H(x) = 1$ for $x > 0$, and $H(x) = 0$ for $x \leq 0$. A log/log plot of the output and an estimate of the slope of the linear region of this graph gives the correlation dimension $d_c$ due to the fact that, by increasing the value of $R$, $C(R)$ should increase as $R^{d_c}$, or, after taking the logarithm of both sides, $\log(C(R)) = d_c \log(R) + \text{constant}$. We repeat these calculations for increasing values of embedding dimensions $d$ while keeping the value of the time lag $L$ fixed. The value of $d_c$ should eventually converge, by increasing $d$, to the true value of the fractal dimension of the attractor. Usually, the proper embedding dimension must be an integer greater than or equal to twice $d_c$ plus one [3].

III. THE PROPOSED APPROACH

The proposed approach for predicting how clones evolve in subsequent versions of a software system consists of the following main steps:

1. Clones identification.
2. Generation of time series data.
3. Testing of chaos existence.
4. Clones prediction.
5. Evaluation of the predicted values.

The previous steps will be explained in more details through the following subsections.

A. Clones Identification

For clones identification, we use the CloncDR tool [24], which is a tool that detects and aids the removal of duplicate code. The used tool was proved to have the highest precision [25]. CloncDR is an Abstract Syntax Tree (AST) based detector. The source code is parsed and an AST is produced for it. After that, three main algorithms are applied to find clones. The first algorithm tries to detect sub-tree clones. The second one is concerned with the detection of variable-size sequences of sub-tree clones, and is used essentially to detect statement and declaration sequence clones. The third algorithm looks for more complex near-miss clones by attempting to generalize combinations of other clones. This is done by hashing on program’s substructures encoded as abstract syntax trees.

B. Time Series Generation

The sequence of numbers representing detected exact and near-miss clones in each version constitutes a basis for time series data. Since these values must be sampled at equal time distances, cubic spline is used for interpolating them. Then 1000 points are sampled from the resulting curve at time step= (time span period of the analyzed versions) /1000.

C. Testing for Chaos

A common method for the identification of chaos in state-space systems is to calculate the maximal Lyapunov exponent [14, 23]. The calculation of this exponent from time series data has been extensively considered in the literature [26]. Lyapunov exponents represent the average exponential rates of divergence or convergence of nearby orbits in phase space [23]. Any system containing at least one positive Lyapunov exponent is defined to be chaotic, with the magnitude of the exponent reflecting the time scale on which system dynamics become unpredictable [27]. Informally, the exponents measure the rate at which the system dynamics create or destroy information, thus the exponents are expressed in terms of bits/iteration. First of all, if the data come from experimental time series, the entire spectrum of Lyapunov exponents cannot be evaluated. In this case, actually, the so-called trajectory tracing method has been developed [27] in order to calculate at least the largest exponent. Positive values for this exponent would prove the divergence of nearby points in the reconstructed phase space and the existence of a strange attractor with sensitive dependence on initial conditions. To calculate Lyapunov exponent, a step by step evolution of a pair of points, a reference one and a candidate one is done. Each time the distance between these two points becomes too long, a replacement procedure of the candidate is applied in such a way that the orientation between the new pair of points is as close as possible to that of the original pair. The details of the used algorithm for calculating Lyapunov exponent can be found in [27].

Another method for detecting the presence of chaos is the calculation of correlation dimension as given in (4). The non integer value indicates the existence of chaos [17].

D. Clones Prediction

The chaotic time series prediction is to forecast the unknown chaotic signals based on observed values. Chaotic systems are non-predictable over a long period of time. However, there is a time duration over which accurate predictions can be made. The approximate period limit on accurate predictions of a chaotic system is inversely proportional to the largest Lyapunov exponent [28]. Thus, chaotic predictions are possible only for systems with Lyapunov exponents between zero and one. If the exponent is much less than one, long, accurate predictions are possible. For prediction, one starts with the unfolded attractor in d-dimensional space and time lag $L$. Given an initial vector $y(n_0)$, one selects the $k$ nearest trajectories on the attractor, and then the $k$ nearest points to $y(n_0)$, one on each trajectory. An average of these trajectories is used to find the next point on the predicted trajectory, $y(n_0 + dL)$. The predicted point is then set as the new starting vector and the process is repeated. The details of the prediction algorithm can be found in [29].
E. Evaluation of the predicted values

For the evaluation of the predicted time series values, we compare the predicted and actual values of the time series by calculating the mean square error (MSE).

IV. AN EXPERIMENTAL STUDY

A. Subject Systems

To evaluate our approach we used as case studies two open source systems. The first one is FileZilla [30] which is an open source, cross-platform, and FTP client. The second system is VLC [31], which is a cross-platform open-source multimedia framework, player and server. Table I presents some metrics about the used two systems.

<table>
<thead>
<tr>
<th>System</th>
<th>Language</th>
<th>KSLOC</th>
<th># versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileZilla</td>
<td>C++</td>
<td>22-105</td>
<td>50</td>
</tr>
<tr>
<td>VLC</td>
<td>C</td>
<td>140-145</td>
<td>100</td>
</tr>
</tbody>
</table>

B. Results

We ran CloneDR tool on each of the analyzed systems. For each subsequent version of the two analyzed systems, we recorded the total number of exact match clones, and near miss clones. For all the conducted experiments, similarity threshold was set to 95%. The recorded data values for each version represent a time series that was analyzed using chaos theory. Thus we have for each system, two sets of time series data of 50 and 100 points (for FileZilla and VLC respectively). The first one represents exact-match clones (DS1); meanwhile, the second time series represents the number of near-miss clones (DS2). The generated data sets were analyzed using the methods discussed in the previous section. For lack of space, complete results are provided for FileZilla only. However, a summary of the results obtained when VLC is used will also be presented.

Fig. 1 shows the interpolated times series for FileZilla. The time step is calculated as the difference between the release date of the last and the first version studied divided by the number of points. Thus it was set to 1.5 day (4.2x12x30/1000) approximately. Moreover, fig. 2 represents the mutual information function vs. the time delay for each of the two datasets for FileZilla.

![Interpolated time series for FileZilla](image1)

![Mutual information function (I) vs. time delay (L) for FileZilla](image2)

We then calculate the correlation dimension for each of the used datasets. Fig. 3 shows the relation between logC(R) and log(R) for different embedding dimensions d for one of the used datasets (DS1). Notice that only the curves for d=7, 9, and 10 are presented since the other 7 curves nearly coincide with these three curves. The slope of the linear part approximates d. In addition, Fig. 4 shows the correlation dimension (d) vs. the embedding dimension (d) for the used datasets. It is shown that as d increases, d saturates at a non integer value (d_c) which indicates chaos. Correlation dimensions for the two data sets of the used systems is presented in Table III.

![LogC(R) vs. Log(R)](image3)

Another indication of chaos is the positive largest Lyapunov exponent (LLE). Fig. 5-6 indicate that LLE’s for DS1 and DS2 for FileZilla are positive. Meanwhile, Table IV presents LLE’s for the four data sets. After detecting the presence of chaos using the above two indicators, we used chaos theory in predicting the evolution of clones in versions in the near future. We use as embedding dimension 2d=6 for the predictions [23]. The last 15 time series values were predicted using the readings at time step=986 and the 15 predicted values were compared with the actual values.
Fig.'s 7-8 present the actual vs. the predicted values for DS1 and DS2 for FileZilla. MSE of the predicted values was 0.007 for DS1; meanwhile it was 0.122 for DS2. It should be noticed that the prediction of new values of DS1 was more accurate than that for DS2. This is due to the smaller LLE for DS1 which makes it possible to make accurate prediction over longer period of time in comparison with DS2.

Table III: Correlation Dimensions for the Used Data Sets

<table>
<thead>
<tr>
<th>System</th>
<th>DS1</th>
<th>DS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileZilla</td>
<td>1.28</td>
<td>1.43</td>
</tr>
<tr>
<td>VLC</td>
<td>1.36</td>
<td>1.56</td>
</tr>
</tbody>
</table>

Table IV: Largest Lyapunov Exponents for the Used Data Sets

<table>
<thead>
<tr>
<th>System</th>
<th>DS1</th>
<th>DS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileZilla</td>
<td>0.1</td>
<td>0.43</td>
</tr>
<tr>
<td>VLC</td>
<td>0.28</td>
<td>0.45</td>
</tr>
</tbody>
</table>

V. RELATED RESEARCH

Many techniques were proposed in the literature for the identification of clones in source code. However, only few works tried to track or model clones in evolving systems, especially in open source systems [8, 32]. As mentioned in [33, approaches for modeling clones evolution can be divided into two categories. The first approach is detecting clones for a single version and tracking clones in following versions according to some information obtained from a source repository [33-34]. The disadvantage is that new clones which are introduced in later versions are not detected. The second is detecting clones in each version and mapping those of consecutive versions afterwards [35-36]. However, with increasing number of versions, the precision of the approach decreases [33]. A more detailed discussion of the existing approaches can be found in [32].

VI. CONCLUSIONS AND FUTURE WORK

In this paper, an approach for modeling clones evolution in subversions of open source systems is presented. The approach employs chaos theory for modeling and estimating new clones in new versions with good accuracy. However, many approximations have been done to be able to use chaos theory. Firstly, the precision of the used clone detection tool which may constitute a threat to validity, especially when the analysis adopts chaos theory which is very sensitive to initial conditions. Secondly, the interpolation of the time series data raises some additional questions about precision. Thirdly, the parameters involved in calculating quantities related to chaos (e.g., mutual information, Lyapunov exponent, etc.) and how the variation of these parameters affects the prediction accuracy...
and even the validity of the presented approach. We believe that these points must be considered as a future work.

In addition, we intend to not only track the average number of clones in subsequent versions but also extend our analysis methods to track the clones themselves in subversions.

REFERENCES


