Software Trustworthiness Complexity Analysis and Trustworthiness Measurement

Presenter: Wei Wei
Email: weiw@buaa.edu.cn

Beihang University, 100191, Beijing, China

2009.10.18
1 Introduction
1 Introduction

2 Dynamical Model of Software Trustworthiness

3 Examples: How Complex Software Trustworthiness Evolution Being
Introduction

Dynamical Model of Software Trustworthiness

Examples: How Complex Software Trustworthiness Evolution Being

Characteristics of Software Trustworthiness Complexity

Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

Conclusion
# Table of contents

1. Introduction

2. Dynamical Model of Software Trustworthiness

3. Examples: How Complex Software Trustworthiness Evolution Being

4. Characteristics of Software Trustworthiness Complexity

5. Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

6. Conclusion
Introduction

Software systems do not work in an expected way very often, and faults and failures arise frequently, directly or indirectly causing the losses for users. Therefore, software systems are not always trustable.

Software trustworthiness is for the purposes of overall characterizing and measuring the behaviors of software systems, and comprehensively considering the characteristic attributes of reliability, safety and security, fault-tolerance and so on.
The scale and complexity of software systems and their components are increasing.

- In the software for 2005 Audi A8 the length of the code is over 90 MB;
- In the software for AirBus A380 with 400 MB code length, about 100 functions provided.

Meanwhile, the running environments of software systems, including network environments and physical environments, are more open and are dynamically changing constantly.
1 Introduction

- The software systems are under the effect of multiple factors of internal risks, e.g., errors and bugs, and external risks, e.g., viruses and malicious codes.

- The software systems survive and evolve in dynamical complex environments, and, in later sections we will point out that their trustworthiness exhibits procedural, structural and nonlinear behaviors, that can be very complex.
The existing measurement and assessment mechanisms for software trustworthiness can not ensure the safe and reliable running of software systems.

In this sense, quantitative indices, measurement and assessment mechanisms for software trustworthiness are the key research topics.

In what follows, we first propose a dynamical model for software trustworthiness, based on the model we show how to analyze software trustworthiness complexity, then give software trustworthiness measurement method by the complexity analysis.
2 Dynamical Model of Software Trustworthiness

2.1 Software behavior trustworthiness

► Generally, the software system trustworthiness includes identity trustworthiness, function trustworthiness and behavior trustworthiness.

► Behavior trustworthiness, the central problem in software trustworthiness study, involves with the behavior evolutions of software systems. The behavior trustworthiness relates to networks, OS’s, middlewares, human-computer interactions and so on. We will focus on behavior trustworthiness study.
2.1 Software behavior trustworthiness

In the authors’ point of view, the behavior trustworthiness of software systems can be considered as that:

- behaviors and results of software systems can be predicted,
- states of software systems can be monitored,
- results of software systems can be assessed,
- exceptions of software systems can be controlled.
2.2 Factors affecting behavior trustworthiness and their evolution

Because software systems dynamically evolve in their life cycle in open and complex environments, the software trustworthiness is evolving under the effect of both internal and external factors. Factors affecting the software trustworthiness mainly include:

- internal crisis, i.e., the errors, bugs, faults, aging and failures, etc.
- external crisis, i.e., virus, malicious codes, etc.
- software survival evolutions in complex and changing environments.
2 Dynamical Model of Software Trustworthiness

2.2 Factors affecting behavior trustworthiness and their evolution

According to the above three factors affecting the software trustworthiness, the following three evolution forms should be implied at least:

- **Natural evolution**: e.g., software systems login and logout in running processes, functions aging and failures, performance degrading and so on.

- **Evolution with human interactions**: e.g., software system testing, maintenance, version update, patches, software knowledge base update and so on.

- **Survivability**: e.g., under network attacks, failures and damage of some components, systems still able to continue finishing central activities of an application task, and to resume the damaged functions.
2.3 Dynamical model for software trustworthiness

Combining with the laws of natural evolution, evolution with human interactions and survival evolution of software systems, software trustworthiness evolutions can be modeled as

\[ F : M \times N \rightarrow N. \]

\( N \) : the space of software trustworthiness attributes, \( N \subset \mathbb{R}^n \),
\( n \) : the number of the attributes under consideration,
\( M \) : the parameter space characterizing external factors, \( M \subset \mathbb{R}^m \), \( m \) : the number of parameters.

A given set of external parameter values determines a specific external environment.
2 Dynamical Model of Software Trustworthiness

2.3 Dynamical model for software trustworthiness

For example, in the discrete case, the model can be written as

\[
\begin{align*}
    x_{1}^{i+1} &= f_1(x_1^i, \cdots, x_n^i, \alpha_1, \cdots, \alpha_m) \\
    \cdots & \cdots \\
    x_{n}^{i+1} &= f_n(x_1^i, \cdots, x_n^i, \alpha_1, \cdots, \alpha_m)
\end{align*}
\]

Here, \((x_1, \cdots, x_n) \in N, (\alpha_1, \cdots, \alpha_m) \in M, F = (f_1, \cdots, f_n)\).
2 Dynamical Model of Software Trustworthiness

2.3 Dynamical model for software trustworthiness

Considering the characteristics of internal and external factors, the evolutionary law of the software trustworthiness can be formulated as \( F = f \circ g \).

**Remark:** \( f \) reflects the internal (natural) evolutionary law and \( g \) reflects the external (human interaction and survival) evolutionary law.

Thus, software trustworthiness evolution can be represented by the iterative behaviors of dynamical system \( F \left( F^1, F^2, \ldots, F^i, \ldots \right) \).
3 Examples: How Complex Software Trustworthiness Evolution Being

➢ The following two simple examples suggest that the evolution of the software trustworthiness could be very complex and involve procedural complexity, structural complexity and nonlinear complexity.

➢ The examples also suggest that the evolution of the software trustworthiness undergo a process from trustworthiness (determinate behavior) to untrustworthiness (indeterminate behavior) with a transformation stage of fluctuation (with some regularity).
Example 1:

Consider that the variation of memory consumption is affected by attacks of viruses in the course of the software system running.

- $x_i$: the memory consumption at time $i$.
- $\alpha$: the relative amount of memory consumption (i.e., $x_{i+1}/x_i$) affected by the virus attack.
- $\beta$: the reduction in the amount of memory consumption by competition is proportional to $x_i^2$, assuming that viruses compete for memory each other.
3 Examples: How Complex Software Trustworthiness Evolution Being

The amount of memory consumption can evolve as follows:

\[ x_{i+1} = x_i (\alpha - \beta x_i). \]

Let \( y_i = \frac{\beta x_i}{\alpha}, \) then

\[ y_{i+1} = \alpha y_i (1 - y_i). \]

That is, the evolutionary law of the system memory consumption is \( F(x, \alpha) = \alpha x (1 - x), \) which is the famous Logistic map.

In this example, only the attribute of memory consumption \( x \) is considered, which can measure the capacity against the attacks of viruses of the software system.
3 Examples: How Complex Software Trustworthiness Evolution Being

Figure: The bifurcation graph \((0 < \alpha < 4)\) for the above system. The \(Y\)-axis is the memory consumption.
3 Examples: How Complex Software Trustworthiness Evolution Being

- By the relation between $x_i$ and $y_i$, the bifurcation type of the evolution of $x_i$ is the same as that of $y_i$.

- For a given parameter $\alpha$ in $(0, 3)$, $x_i$ takes a deterministic value, which suggests that the evolutionary trend of the memory consumption can be predictable and then the software can be trustable.
3 Examples: How Complex Software Trustworthiness Evolution Being

When parameter $\alpha$ varies in $(3, 1 + \sqrt{6})$, $x_i$ exhibits period-2 bifurcation, period-4 bifurcation, $\cdots$, period-$2^k$ bifurcation and so on. Obviously, software trustworthiness enters a stage of fluctuation with some regularity.

When $\alpha$ passes $1 + \sqrt{6}$, the chaos phenomenon appears. Thus, the amount of memory consumption, $x_i$, will be totally unpredictable and the corresponding software is completely untrustable.
Example 2:

Consider that a software system is affected by both an internal factor (the number of concurrent users) and an external one (the memory consumption).
3 Examples: How Complex Software Trustworthiness
Evolution Being

► \( x_i \): the memory consumption at time \( i \), and assume that the relative amount of memory consumption is the same at different time and this ratio denoted as parameter \( \alpha \).

► It is assumed that viruses compete for memory each other, and the reduction of amount of memory consumption by competition is proportional to \( x_i^2 \), and the ratio is denoted as \( b \).

► \( y_i \): the number of concurrent users at time \( i \), and the number of concurrent users will lead to memory consumption.

► \( c \): the parameter for adjustment between the number of concurrent users and the amount of memory consumption.

► Assuming that the ratio of the number of concurrent users at time \( i + 1 \) and memory consumption at time \( i \) is fixed as \( d \).
3 Examples: How Complex Software Trustworthiness Evolution Being

This system can be mathematically described as follows:

\[
\begin{align*}
    x_{i+1} &= x_i(a - bx_i) + cy_i \\
    y_{i+1} &= dx_i.
\end{align*}
\]

This system can be reduced to a famous model in astronomy, i.e. the Hénon map

\[
\begin{align*}
    u_{i+1} &= 1 - \alpha u_i^2 + v_i \\
    v_{i+1} &= \beta u_i.
\end{align*}
\]
3 Examples: How Complex Software Trustworthiness Evolution Being

Figure: The chaos phenomenon at parameters $\alpha = 1.4$, $\beta = 0.3$ for the system.

That is, the software trustworthiness evolves in a disordered way and even is ergodic in some intervals, which makes the software trustworthiness completely uncontrollable and unpredictable. It indicates the software untrustable that the software trustworthiness evolution exhibits extreme unstableness.
3 Examples: How Complex Software Trustworthiness Evolution Being

The above software trustworthiness evolution models give the interpretations of software trustworthiness in dynamics:

- predicting the behaviors and results of software systems corresponds to a class of dynamical behaviors of dynamical systems,
- monitoring the behavioral states corresponds to the orbit analysis of dynamical systems,
- assessing the behavioral results corresponds to the behavioral analysis of the limit sets of dynamical systems,
- controlling the behavioral exceptions corresponds to the study of structural stability and bifurcation of dynamical systems.
4 Characteristics of Software Trustworthiness Complexity

4.1 Procedural, structural and nonlinear complexity characteristics

Software systems dynamically evolve in their life cycles in open and complex environments. As the above examples shown, dynamical evolutions possess the following complexity characteristics:

- **Procedural complexity**: software trustworthiness is developing and evolving constantly, and their behaviors and results are uncertain, or even unpredictable.

- **Structural complexity**: many factors affecting the software trustworthiness, dynamically increase or decrease, and the relationships dynamically change.

- **Nonlinear complexity**: factors may interact to one another in nonlinear form.
4.2 How to approach procedural, structural and nonlinear complexity

- Structural complexity and nonlinear complexity of software trustworthiness are reduced to the complexity of modeling of software trustworthiness,

- Procedural complexity is reduced to the complexity of iteration processes and the limit behaviors of the dynamical systems corresponding to software trustworthiness models.

They determine the trustworthiness of software systems comprehensively.
According to the above characteristics of software trustworthiness complexity, the invariant-measure theory, from the dynamical statistical analysis, will be used to characterize the statistical laws governing the software trustworthiness complexity in this presentation.
5.1 Brief for invariant-measure theory

The basic idea of invariant-measure theory is that,

Do statistics of the average return frequency of each attribute (index) in software systems by tracing and recording the values of the trustworthiness attributes during software system running,

Obtain the probability distributions of trustworthiness attributes taking certain values in the limit states of software trustworthiness evolutions.

Remark: The invariant-measure theory can reflect the long term behavior complexity characteristics of software trustworthiness.
In Example 1, consider that the variation of memory consumption is affected by attacks of viruses in the course of the software system running and assume that the system is not attacked by viruses at the initial time. The evolutionary law of the system memory consumption is $F(x, \alpha) = \alpha x (1 - x)$.

The $v_{th}$ iteration of the system is denoted by

$$\xi_v(\alpha) = F^v(0, \alpha)$$

where $\alpha$ is an external parameter characterizing the attacking capacity of viruses.
5 Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

5.2 Revisiting Example 1

For any given memory consumption amount $x \in [0, 1]$, do statistics for the times of $\xi_v(\alpha)(v = 0, 1, 2, \cdots, i)$ taking $x$.

Let $\delta_{\xi_v(\alpha)}(x)$ be counting functions, which are 1 for $\xi_v(\alpha) = x$, otherwise, are 0,

then, $\sum_{v=1}^{i} \delta_{\xi_v(\alpha)}(x)$ is the number of times of the value of memory consumption amount taking $x$ after $i$ times measurements.
5 Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

5.2 Revisiting Example 1

Let \( \mu_i(x; \alpha) = \frac{1}{i} \sum_{v=1}^{i} \delta \xi_v(\alpha)(x) \)

then \( \mu_i(x; \alpha) \) will weakly converge to \( \mu(x; \alpha) \) when \( n \to \infty \).
5 Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

5.2 Revisiting Example 1

Here $\mu_i(x; \alpha)$ is the frequency that the memory consumption amount takes the value $x$ after measuring $i$ times.

For any given $x$, the limit $\mu(x; \alpha)$ is the statistical distribution of probability that the memory consumption amount takes the value $x$, and reflects the objective law of software trustworthiness complexity.
Figure: The frequency $\mu_i(x; \alpha)$ versus $x$ with $\alpha = 4$ and $i = 1 \times 10^5$. The red curve is the simulation result and black curve is the theoretical result of $\mu(x; \alpha)$. 
5 Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

5.2 Revisiting Example 1

By the results shown in the above figure,

- memory consumption amount can certainly take almost all the values in interval $[0, 1]$ during the software trustworthiness evolution,

- the probability that the memory consumption amount takes values in any given small intervals exists, ensuring the software trustworthiness can be measurable.
5 Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

5.2 Revisiting Example 1

- This shows that the memory behaviors of software systems exhibit strong randomness under the effect of virus,
- the memory behaviors are totally unpredictable after long term evolutions.

Consequently, the behaviors of the software system are untrusted when parameter $\alpha$ characterizing viruses takes value 4.
5 Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

5.2 Revisiting Example 1

Consider software systems with multiple trustworthiness attributes. Similarly above, we can define the weak limit $\mu_F$ of

$$\frac{1}{i^m} \left[ \sum_{v_1=1}^{i} \sum_{v_2=1}^{i} \cdots \sum_{v_m=1}^{i} \delta_F(v_1,v_2,\ldots,v_m)(\alpha)(x) \right]$$

to be the statistical measurement of the software trustworthiness complexity for the software systems.
5 Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

5.2 Revisiting Example 1

Thus, procedural, structural and nonlinear complexity can be approached as above.

**Remark:** If $\mu_{\mathcal{F}}$ is invariant and is absolutely continuous under Lebesgue measure ($L^1$), then the system is ergodic, and the corresponding dynamical behaviors are chaotic, which shows that the behaviors of the corresponding software system are untrusted.
5 Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

5.3 A criterion for untrustworthiness

Thus, based on the invariant-measure theory of dynamical systems, the authors

- give a dynamical criterion for untrustworthiness of software systems, i.e., a necessary condition for software trustworthiness.

- describe the dynamical criterion for software untrustworthiness by the behavior characteristics of the software trustworthiness limit evolutions.
5 Analyzing Software Trustworthiness Complexity by Dynamical Statistical Analysis Methods

5.3 A criterion for untrustworthiness

Once the mean tendency of the indices of trustworthiness attributes weakly converges to an absolutely continuous invariant measure, then the behaviors of the corresponding software systems are untrusted eventually.
6 Conclusion

The software trustworthiness is some statistical characteristics of the behaviors of software systems in dynamical and open environments, and interpret the dynamical characteristics of software trustworthiness and their evolutionary complexity as follows:

- Predicting the behaviors and results of software systems corresponds to a class of dynamical behaviors of dynamical systems,
- Monitoring the behavioral states corresponds to the orbit analysis of dynamical systems,
- Assessing the behavioral results corresponds to the behavioral analysis of the limit sets of dynamical systems,
- Controlling the behavioral exceptions corresponds to the study of structural stability and bifurcation of dynamical systems.
Based on the dynamical model of software trustworthiness, we

► interpret the characteristics of behaviors of software systems and the basic scientific problems of software trustworthiness complexity,

► analyze the characteristics of complexity of software trustworthiness,

► indicate that the software trustworthiness measurement can be studied by means of the complexity of software trustworthiness.
Using the dynamical statistical analysis methods, we propose an invariant-measure based assessment method of software trustworthiness by statistical indices, and provide a dynamical criterion for the untrustworthiness of software systems.

By examples, the feasibility of the proposed dynamical statistical analysis method in software trustworthiness measurement is demonstrated using numerical simulations and theoretical analysis.
Thank you!

Email: weiw@buaa.edu.cn