Progress and Quality Modeling of Requirements Analysis Based on Chaos

Junwei Ge, Yiqiu Fang
{Faculty of Software, Faculty of Computer Science and Technology}
Chongqing University of Posts and Telecommunications
Chongqing 400065, China
{gejw, fangyq}@cqupt.edu.cn

Abstract

It is important and difficult for us to know the progress and quality of requirements analysis. We introduce chaos and software requirements complexity to the description of requirements decomposing, and get a method which can help us to evaluate the progress and quality. The model shows that requirements decomposing procedure has its own regular pattern which we can describe in a equation and track in a trajectory. The requirements analysis process of a software system can be taken as normal if its trajectory coincide with the model. We may be able to predict the time we need to finish all requirements decomposition in advance based on the model. We apply the method in the requirements analysis of home phone service management system, and the initial results show that the method is useful in the evaluation of requirements decomposition.

1. Introduction

Requirements specification is the beginning of the major phases of software design process. The initial description of the desired product may be vague, unreasonable, contradictory, or simply impossible to achieve. One of the main tasks of requirements analysis is to elaborate the user requirements to discover more about the implications of satisfying them. We need to determine exactly what it is that the client and system need and to find out from the client what constraints exist before system architecture design [1]. Generally speaking, the process of requirements engineering starts from initial user requirements and advances towards system requirements that are refined by a structural decomposition until they can be implemented [1], [2], [3], [4], [5]. Here are some outstanding problems in requirements decomposition. One is absence of clear guidelines for requirements decomposition [6]. The second is when the decomposition process can be thought enough. We should identify how we'll know when the system meets each requirement [7]. The third is about the reliability of decomposition process. We should know how to evaluate the correctness of the process when requirements are under decomposition. These problems puzzle us in requirements analysis.

Based on the theory of nonlinear dynamic system, we developed a chaotic decomposition modeling which can guide us decomposing requirements, and help us to estimate the decomposition phase and evaluate its quality.

2. Backgrounds

A typical process of software requirements decomposition is represented in Figure 1. The initial requirements for a complex software system should be decomposed into sub-requirements because the initial requirements usually are very abstract at the beginning and require further concretization. The decomposition principle is applied to break a complex problem into a hierarchy of clusters, sub clusters, sub-sub clusters and so on [1], [3], [8], [9], [10], [11].

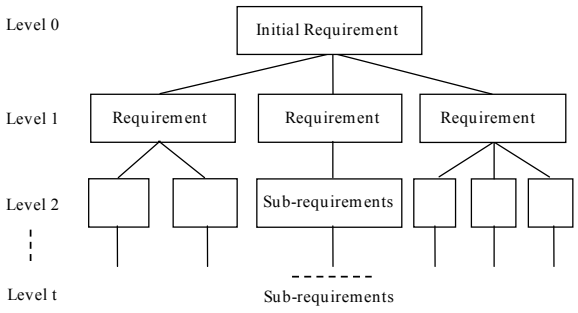

Figure 1. **Typical requirements decomposing process of a software system**

978-0-7695-3432-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ASEA.2008.22

Where, same objectives in the same level will be taken as one objective. Objectives in the upper level will be inherited if they will not be decomposed further. The goal of a system development is to build a system that will satisfy the agreed requirements. This task is very hard if there is a large and complex set of requirements for the system.

Software requirements decomposition can be taken as a dynamic system for it can evolve over time. It is a discrete process, so we shall discuss it as a discrete system in the following sections.

A discrete dynamical system is specified by difference equations, and it can be written as

$$x_{t+1} = f(x_t, t).$$

We studied the process of software requirements decomposition from dynamical system and system complexity [12]. The results showed that the process can be taken as a nonlinear dynamical system. It can be written as

$$x_{t+1} = \alpha g(x_t, N) + x_t. \quad (1)$$

Where, α is a constant (≥ 0) which we call the Requirements Decomposition Rate Parameter (RDRP) and represents the increased number of requirements derived directly from a single initial requirement in layer 0, t = 0, 1, 2, …represents the decomposition layer t, N is the maximum number of requirements that the system can have, and $x_t = n_t/N$ represents requirements saturation where the number of decomposed requirements in layer t is n_t. Function $g(x_t, N)$ is decided by requirements complexity, which is taken as in direct ratio of the number of interactions among requirements in the system [12].

For the simplest situation where requirements complexity is in direct ratio of n_t - 1, the process can be described as,

$$x_{t+1} = [\alpha/(1-1/N)](1-x_t)x_t + x_t. \quad (2)$$

Equation (2) is a chaotic system. It has bifurcations and a three-point attractor. It is sensitive to initial requirements x_0 (or n_0) and has unpredictable final states in chaos although it is deterministic.

Behaviors of (2) are determined mainly by the parameter RDRP, which can be divided into stable and unstable regions. Suppose N = 1000. If RDRP is in its stable region, requirements can be fully decomposed in a stable manner, as shown in Figure 2. If, on the other hand, RDRP is in its unstable region, the decomposition process will be unstable or in chaos and the decomposed results will be erratic, as shown in Figure 3. In this situation it may be impossible for us to identify all sub-requirements because a small change of initial requirements will cause a large change in the decomposition.

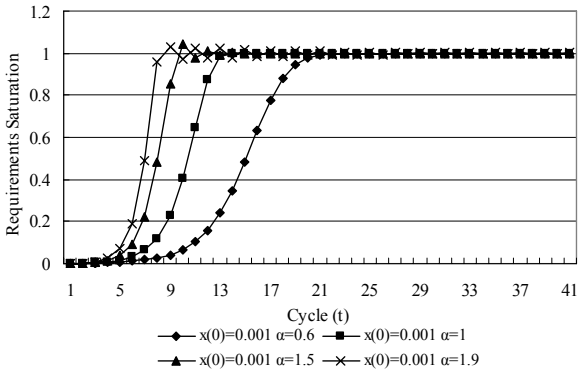

Figure 2. **Behaviors of decomposition in a stable manner**

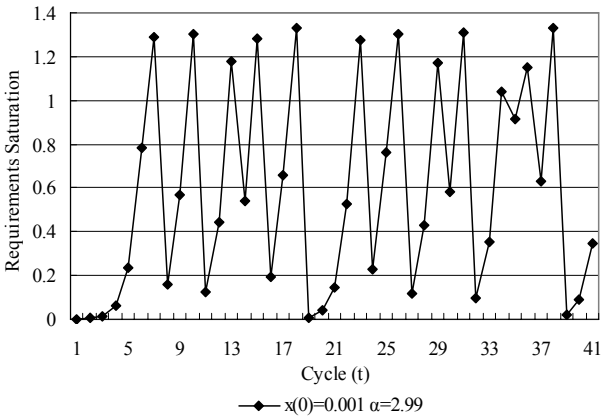

Figure 3. **Behavior of decomposition in an unstable manner**

Therefore, requirements decomposition should be performed in stable region. Based on the theory above, we'll derive following decomposition modeling which can help us to gain all requirements.

3. Modeling

From Equation (2), we can see that software requirements decomposition equation have three parameters i.e. RDRP α, initial value x_0, and maximum number N. Let N = 100, 1000 and 5000 under the situation of α = 1 and x_0 = 0.01, than we can get three decomposition trajectories, where α is in its stable region. The result shows that three trajectories coincide with each other. The trajectories with different N are almost same so long as α and x_0 don't change. Therefore, decomposition will not affected by N.

Let N = 1000, and x_0 = 0.001, 0.01, 0.1, 0.3, and 0.7 separately, we can get decomposition time series separately. The results show that the decomposition

trajectories are related with x_0 as well as α, but they affect decomposition differently.

Parameter α will affect the slope of trajectories mainly. Parameter x_0 will affect the iteration time when trajectory slope arrives its maximum. The time will be shorter as x_0 increases. Larger the initial value, more quickly we finish requirements decomposition.

We can divide a trajectory into three segments, which we call as initial segment, middle segment, and last segment separately, as shown in Figure 4.

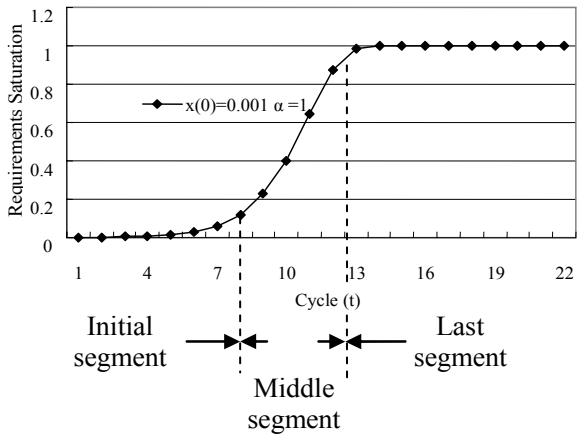

Figure 4. **Segments of a trajectory where *N*=1000, x_0=0.01 and α=1**

Initial segment is located at the beginning parts of decomposition trajectories. In these parts, only a few of requirements have been decomposed. One of its characters is that the number of requirements is small. Another is the increasing rate of requirements or the slope of trajectories is also small, so that the number of requirements increases very slowly. The length of initial segment is related with α and x_0. It will decrease when α or x_0 increases.

Middle segment is located at the middle parts of trajectories. In these parts, more and more requirements have been decomposed. Its character is that the number of requirements begins to increase quickly, that is, the slope of trajectories is large. The maximum slope appears in this segment. The length of this segment is also related with α and x_0. It will change as well as initial segment.

Last segment is located at the last parts of trajectories. Its characters are that the number of requirements is approaching the maximum and the slope of trajectories is decreasing to zero as decomposition is going. The trajectories may be vibrate near the maximum when α is near 2, but it will converge at the maximum.

We can use Figure 4 to estimate the decomposition phase and evaluate its quality. The method, which we call the chaotic decomposition modeling, is described as follows.

(1) Define the initial value of requirements. It may be initial number n_0 or initial requirements saturation x_0 if we can approximately define the final number of requirements N by concept exploration.

(2) Define RDRP α. RDRP should be in its stable area i.e. less than 2. RDRP is the initial requirements decomposition rate. The rate will be less slowly as requirements number n_t increases [12].

(3) Draw decomposition trajectories as we decompose requirements. The decomposition will be going until it arrives at its last segment.

(4) Evaluate decomposition. It can be taken as normal if the trajectory changes like Figure 4. Otherwise, the decomposition may be abnormal.

(5) Accept or reject results. We can stop decomposing requirements when the decomposition arrives at its last segment normally. Now we can argue that we have achieved almost all of requirements. We can also dismiss decomposition and redo it later when we find that the decomposition is abnormal.

In addition, we may predict the iterating time we need to finish the decomposition with the help of decomposition trajectories templates under different values of α and x_0 if we have known the approximate value of final number N.

4. Initial application

We apply the chaotic decomposition modeling in the requirements analysis of a kind of home phone service management systems which we developed successfully in china before. Let its initial requirements number equal 1 and RDRP equal 1 also.

Figure 5 gives the trajectory of the number of requirements decomposing results. The number increases from 1 to 138 through 12 levels of decomposition. It includes all of three segments. Initial segment is between decomposing time 1 and 5. Middle segment is between time 5 and 11. Last segment begins from time 11. We can achieve all of requirements at time 12. The results show that the decomposition is normal.

5. Discussion

Table 1 gives the maximum changing rate of decomposing trajectories from equation 2. It exists in middle segment. It shows that the rate is mainly controlled by RDRP α when $x_0 \leq 0.3$. Larger α is, larger the rate is. We may use larger α in order to have higher decomposing efficiency. RDRP should be

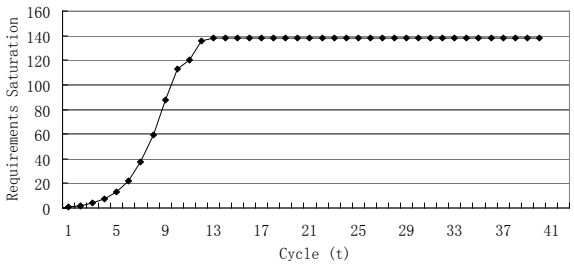

Figure 5. **Requirements decomposing behavior of home phone service management system**

suitable i.e. less than 2. Otherwise, decomposition may be unstable, even chaotic. Therefore, we can predict that the final decomposing results may be unreliable if the changing rate of middle segment is greater than 0.5. If the changing rate is less than 0.1, we can predict that the decomposition may also fail because the decomposing efficiency is too low to finish all decomposition on time for us.

Table 1. **Maximum changing rate of decomposing trajectories**

x_0	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 1.5$	$\alpha = 1.9$
0.001	0.124	0.241	0.375	0.475
0.01	0.125	0.25	0.335	0.469
0.1	0.125	0.245	0.375	0.411
0.3	0.125	0.25	0.355	0.4
0.7	0.105	0.21	0.315	0.399
0.9	0.045	0.09	0.135	0.171

6. Conclusion

From the discussion above, we can see that requirements decomposing procedure has its own regular pattern which we can describe in chaos and track in the procedure. When it is in normal situation, its trajectory can be divided into three parts i.e. initial segment, middle segment and last segment. Theoretical analysis and initial application have approved this kind of chaotic modeling. If we achieve all three segments or last two segments in requirements decomposing trajectory of a software system, we can argue that the procedure is normal and we have achieved all or near all of the requirements. We should take suitable requirements decomposition rate in order to insure the procedure normal, avoid chaos and keep a suitable decomposing efficiency. When decomposition is in middle segment, we can predict whether the procedure will be successful according to the changing rates of its trajectory. Both of too high and too small changing rates may cause the process failure.

Acknowledge

This work was supported in part by a grant from the China Association for International Exchange of Personnel, Chongqing Science & Technology Commission, and Chongqing Municipal Education Commission. J. Ge wishes to thank Shayne Flint for his great help on the research.

References

[1] Schach S.R., *Classical and Object-Oriented Software Engineering with UML and C++*, McGraw-Hill Companies, Inc., Singapore, 1999.

[2] Budgen D., *Software Design*, Person Education Limited, Essex, 2003.

[3] Heumesser N., *Framework for Requirements*, Document of EUREKA ∑! 2023 – ITEA 00103 project EMPRESS, EMPRESS consortium, http://www.empressitea.org/deliver-ables/D3.1_v1.0 _Public_Version.pdf, Apr. 2004.

[4] J. Noppen, M. Aksit, V. Nicola and B. Tekinerdogan, "Market-driven approach based on Markov decision theory for optimal use of resources in software development", *IEE Proc.-Softw.*, IET, London, Vol. 151, Issue 2, Apr. 2004, pp. 85-94.

[5] F. P. Brooks Jr, "No silver bullet: essence and accidents of software engineering", *IEEE Computer*, IEEE Computer Society, Washington, DC, Vol. 20, Issue 4, Apr. 1987, pp. 10-19.

[6] R. Chitchyan et al, *Survey of Aspect-oriented Analysis and Design Approaches*, Document of AOSD-Europe-ULANC-9, May 1995.

[7] I. Alexander, "10 small steps to better requirements", *IEEE Software*, IEEE Computer Society, Washington, DC, Vol. 23, Issue 2, Mar. 2006, pp. 19-21.

[8] Forman E. and M. A. Selly, *Decision by objectives*, George Washington University, Washington, DC, http://mdm.gwu.edu/Forman/DBO.pdf

[9] P. Andritsos and V. Tzerpos, "Information-theoretic software clustering," *IEEE Transactions On Software Engineering*, IEEE Computer Society, Washington, DC, Vol. 31, No. 2, Feb. 2005, pp. 150-165.

[10] B. S. Mitchell and S. Mancoridis, "On the Automatic Modularization of Software Systems Using the Bunch Tool," *IEEE Transactions On Software Engineering*, IEEE Computer Society, Washington, DC, Vol. 32, No. 3, Mar. 2006, pp. 193-208.

[11] H. B. K. Tan, Y. Yang, and L. Bian, "Systematic Transformation of Functional Analysis Model into OO Design and Implementation," *IEEE Transactions On Software Engineering*, IEEE Computer Society, Washington, DC, Vol. 32, No. 2, February 2006, pp. 111-135.

[12] J. Ge, Y. Fang, and S. Flint, "Using Nonlinear Dynamic System to Better Understand and Control Requirements Decomposition Process," *The 2nd International Conference on Computer Science & Education*, Proceedings of ICCSE 2007, Wuhan, China, July 2007, pp. 998-1003.