A Novel Software Evolution Model Based on Software Networks

Weifeng Pan, Bing Li, Yutao Ma, and Jing Liu

State Key Laboratory of Software Engineering,
Wuhan University, 430072 Wuhan, P.R. China
panweifeng1982@gmail.com

Abstract. Many published papers analyzed the forming mechanisms and evolution laws of OO software systems from software reuse, software pattern, etc. There, however, have been fewer models so far merely built on the software components such as methods, classes, etc. and their interactions. In this paper, a novel Software Evolution Model based on Software Networks (called SEM-SN) is proposed. It uses software network at class level to represent software systems, and uses software network's dynamical generating process to simulate activities in real software development process such as new classes' dynamical creations and their dynamical interactions with already existing classes. It also introduces the concept of node/edge ageing to describe the decaying of classes with time. Empirical results on eight open-source Object-Oriented (OO) software systems demonstrate that SCM-SN roughly describes the evolution process of software systems and the emergence of their complex network characteristics.

Keywords: software networks, evolution model, software complexity.

1 Introduction

With the development of software development technologies, the size of software systems is becoming larger and larger, together with the structure of software systems. It makes the development, testing, maintenance and management of software systems a hard work. It is often the case that software development is out of control. Researchers always can not view the structures and evolution laws of software systems in whole perspectives, which make themselves in the darkness about the essences of software systems. Hence, it is a challenge for people in software engineering (SE) to recognize, measure, management and control the complexity of software systems [1].

In resent years, researchers in the field of statistical physics and complex system used complex networks (software network) to represent software systems by taking software components as nodes and their relationships as edges. It provides us a new way to analyze software complexity. In 2002, Valverde et. al. are the first to carry out researches on software networks [2]. They obtained the class diagrams from source codes by reverse engineering tools. Then, they used undirected diagrams to represent software systems and revealed some complex network characteristics in software networks. Later on, lots of literatures got similar conclusions on the class diagrams of many OO software systems [3] [4] [5]. But the pioneering researchers are mainly from complexity science and statistical physics, what they focus on is to find out complex network statistical characteristics in software systems, usually ignoring studying the forming mechanisms and evolution laws of software systems.

A novel software evolution model based on software networks (SEM-SN), with reference to the recent advancements in the area of weighted complex networks studies [6] [7] [8] [9] [10], is proposed in this paper. Its main focus is to study the evolution process of software systems from forming mechanisms and evolution laws and give some insights to the reasons for why so many software systems share complex network characteristics. The experiments on eight open-source OO software systems show that SEM-SN can roughly depict the emergence of complex networks properties of real software systems, and it provides us a new angle of view to analyze the software complexity.

The rest of this paper is organized as follows. In section 2, a brief introduction to existing software evolution models will be given. Section 3 details SEM-SN model. In section 4 the proposed SEM-SN will be applied to simulate the forming process of software networks obtained from eight open-source OO software systems. Section 5 concludes the paper. Implication of the discoveries is given and future work is proposed.

2 Related Work

There have existed some typical software networks evolution models. Sergi Valverde and Ricard V.Sole found the duplication rule of node growth in cell networks [11] [12], and based on which they presented a model of network growth by duplication and divergence [13]. Christopher R.Myers, based on refactoring process which captures some of the salient features of the observed systems [14], presented a model for software system evolution. He Keqing, Peng Rong et. al. proposed an OO software network evolution growth method and its algorithms according to the analysis of the growth features of software patterns [15].

Though this is not the first work on evolution models of software systems, we will cover a different angle: SEM-SN uses software network at class level to represent software systems, and uses software network's dynamical generating process to simulate software development process. It quantifies nodes (class) and edges (interactions between classes) in software network by strength values, dynamically update the strength values on nodes and edges, fully take into consideration the influences of newly added nodes to the existing software network, and introduces the concept of node/edge ageing to describe the decaying of classes with time.

3 SEM-SN Model

Software systems as artificial complex systems are the collective pearls of wisdom. The components in software systems possess purposiveness and autonomy to a certain degree. And the interactions between autonomy and flexibility to environment ultimately drive the development and evolution of software systems. The emergence of complex network characteristics in software networks is from such a software evolution. In this section, we will detail the principles of SEM-SN, and the model algorithm.

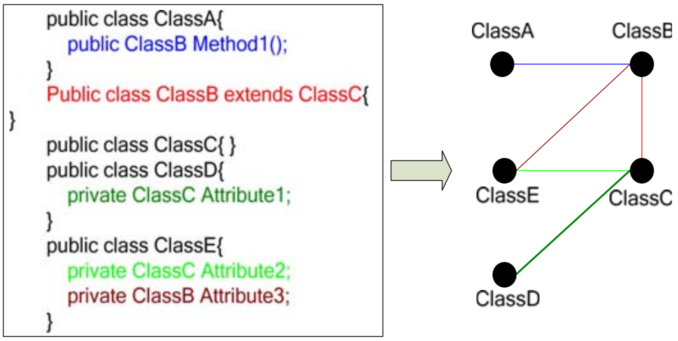
3.1 Principles of SEM-SN

Classes are key software components in OO software systems, and they are gathered up by interactions such as dependency, inheritance, association, etc to compose a software system to fulfill certain functions. In this paper, software systems are represented by software network at class level by taking classes as nodes, and their relationships as edges. The definition of software network at class level is as follows:

Definition 1. *Software Network at Class Level (SNCL). Every class maps to a single node in SNCL, and the relationships between classes are represented by edges. Here we only take into consideration three kinds of class-class relationships such as dependency, inheritance and association. And the edge directionality is discarded, i.e., we only consider undirected version of SNCL. And here we consider only the presence of relationships and neglect the multiplicity such as A depends three times on B. See fig.1 for an example. Therefore SNCL can be described as:*

$$Network_{SNCL} = (Nodes, Edges) \tag{1}$$

Where $Network_{SNCL}$ is an undirected network denoting SNCL; Nodes are classes; Edges denotes the relationships between classes.



```
      public class ClassA{
         public ClassB Method1();
      }
      Public class ClassB extends ClassC{
}
      public class ClassC{ }
      public class ClassD{
         private ClassC Attribute1;
      }
      public class ClassE{
         private ClassC Attribute2;
         private ClassB Attribute3;
      }
```

Fig. 1. A simple example of SNCL

In the following, the principles of SEM-SN will be detailed on four aspects: edges in SNCL, nodes in SNCL, the growth of SNCL and node/edge aging.

Edges in SNCL. As is shown in fig.1, edges in SNCL do not share the same meaning. Some denote dependency relationship (such as the type of return value of Method1 in ClassA is type ClassB). Some denote inheritance relationship (such as Class ClassD is a subclass of Class ClassC). Some denote association relationship (such as Class

ClassD has a type ClassC member attribute Attribute1, and ClassE has type ClassC and ClassB member attributes Attribute2 and Attribute3 separately). But there is lack of a quantified criterion to differ different relationships in literatures. So here we do not take into account detailed relationship semantics, i.e. all relationships have been treated all equal and have the same importance. But we can find intuitively that edges existing between two important nodes which have more edges attached have a relatively more important role. If we delete such edges, the functions and performances of the system will be greatly affected. In SEM-SN, we will use the strength values on the edges to make this distinction. In SEM-SN, the strength value of each edge is determined by the degree of the nodes on its two sides, which will be detailed in Definition 3, 3.2.

Nodes in SNCL. Classes in software systems usually have different importance on the whole system. Abstract classes are usually easy to be inherited [1], classes which are design patterns themselves will be widely used in the system, classes to fulfill basic functions are easy to be referenced, etc. Such kinds of classes usually have more edges attached to them. So the importance of the classes usually can be revealed by the number of edges attached to it. As we talked above, edges attached to nodes have different importance, so in software systems, we can not simply use the number of edges to describe the importance of the nodes. To distinguish the importance of each class in software systems, SEM-SN puts a strength value (defined in 3.2 Definition 4) to a node to quantify the strength of the node according to the strength value of edges attached to it.

Software networks growth. In software development process, new classes will be added into the software system from time to time. And the new classes always will interact with a certain number of existing classes in software systems by dependency, inheritance, association, etc. This interaction will definitely put some influences on the nodes and edges, and finally influence the whole system. Hence, in SEM-SN, in order to depict such kind of dynamical influences, the strength value of nodes and edges should be updated real time with the newly added nodes until the size of SNCL reach a certain size. And the probability that a node will be linked to newly added node is depend on their strength value. It will be detailed in 3.2 Definition 5.

Node/edge aging. In software development the newly added classes always denote some new functions to the system, and they will be frequently used in a certain period of time, but with the time steps, they will become increasingly rarely used. In SEM-SN this phenomenon is called aging. So in SNCL, the strength value of nodes and edges should also decrease with time steps. In SEM-SN node aging is realized indirectly by the edge aging. It will be detailed in definition 7, 3.2.

3.2 Basic Definitions

In this subsection, some basic concepts used in SEM-SN will be given first.

Definition 2. Degree [1]. *Define the degree of a node as the number of edges attached to it.* k_i *is usually used to denote the degree of node i.*

Definition 3. Edge Strength value. *When a new node is added to SNCL, it will interact with other already existing nodes by adding certain number of edges to them. In SEM-SN the strength value of the newly added edge will be given according to*

$$s\_edge_{i \leftrightarrow NewNode} = \frac{k_i}{\sum_{i' \in Set1} k_{i'}} \tag{2}$$

Where $s\_edge_{i \leftrightarrow NewNode} = s\_edge_{NewNode \leftrightarrow i}$ *denotes the strength value of edge linking node i and node j;* k_i *denotes the degree of node i;* $Set1$ *denotes the node set which consists of nodes already existing in software networks;* i' *is a node chosen from* $Set1$.

Definition 4. Node Strength value. *In SNCL, the strength value of a node is closely associated with the edges between the nodes. SEM-SN ignores the complexity of the node (class) itself such as how many methods/attributes/lines it has, and computes the strength value of the node only through the edges attached to it according to*

$$s\_node_i = \sum_{j \in Set2} s\_edge_{i \leftrightarrow j} \tag{3}$$

Where s_node_i *denotes the strength value of the node i;* $s\_edge_{i \leftrightarrow j}$ *denotes the strength value of edge linking node i and j;* $Set2$ *denotes the node set which consists of nodes that linked to node i by edge.*

Definition 5. Creating Operator. *Generate a new node NewNode with strength value 0, and link it to n already existing node* $ExistNode_i (i = 1,2,...,n)$ *with* $m' (m' < n)$ *edges. The new node NewNode will be linked to node i according to probability:*

$$P_{NewNode \to i} = \frac{s\_node_i}{\sum_{j \in Set3} s\_node_j} \tag{4}$$

Where $P_{NewNode \to i}$ *denotes the probability that NewNode will be linked to node i;* s_node_i *denotes the strength value of node i;* $Set3$ *is a node set denoting the already existing nodes in SNCL. After adding a new edge to SNCL, the strength value of the nodes and edges should be given according to (2) and (3).*

SEM-SN uses roulette wheel selection [16] to select the nodes to be linked to *NewNode*. The selection procedure can be described as follows:

```
int RouletteWheelSelection
{//begin
   double sum=0.0;
   double r=0.0;
   r=random(0,1);
   for(int i=0;i<n+1;i++)
```

```
      {
            sum=sum+p[i];
            //p[i] is the probability to select node i
            If(sum<r && r<=sum+p[i+1])
            {
                  j=i+1;
                  return j;//j is the selected node
                  Break;
            }
      }
   }//end
```

The edge number *m'* is randomly selected from 1 to *(m+1). m* is an integer.

Definition 6. Dynamic updating Operator. *When node j is linked to new node NewNode, the strength value of the node should be updated according to*:

$$s\_node_i = s\_node_i + s\_edge_{i \to NewNode} \tag{5}$$

Where $s\_edge_{i \to NewNode}$ *is the strength value of the edge linking node i to NewNode.*

At the meanwhile, the degree of nodes, edge strength value and probability $P_{NewNode \to i}$ all should be updated correspondingly.

Definition 7. Aging Operator. *Here, we introduce the concept of node/edge aging, i.e., each node and edge has its age, and he will gradually aging with time steps. The aging process of edges is shown as:*

$$s\_edge_{i \leftrightarrow j} = s\_edge_{i \leftrightarrow j}(1-D)^{timeStep} \tag{6}$$

Where $D \in (0,1)$ *is the aging factor; timeStep denotes the time step of algorithm running (it is equal to the number of added nodes); the other denotations have the same meaning as talked above.*

And the node aging is implicitly included in (6) by the updating process of strength value according to (5).

Definition 8. Clustering Coefficient [1]. *The clustering coefficient of a network describe how well connected the nodes are. Fist, define the coefficient of the node C_i. Suppose that a node i has K_i neighbours; then at most $k_i(k_i$-1)/2 edges can exist between them. Let C_i denotes the fraction of these allowable edges that actually exist. That is:*

$$C_i = \frac{2 \times the\ actual\ edges\ that\ exist}{k_i(k_i - 1)} \tag{7}$$

The clustering coefficient for the whole system as the average of the clustering coefficient for each node:

$$C = \frac{\sum_{i=1}^{N} C_i}{N} \tag{8}$$

Where N is the number of nodes.

Definition 9. Average path length [1]. *Define the average path length l:*

$$l = \frac{1}{\frac{1}{2}N(N-1)} \sum_{i<j} d_{ij} \tag{9}$$

Where d_{ij} is the shortest path length between node i and node j; N is the number of nodes. The average path length can be used to assess the overall property of the network.

3.3 SEM-SN Model Algorithm

Based on the analysis above, in this subsection the model algorithm of SEM-SN will be detailed.

The SEM-SN model algorithm can be described as:

STEP 1. Initialize the parameters of SEM-SN model algorithm, including the number of node in software *N*, the size of initial SNCL N_0, the number of edges to be added for each newly added node *m*, and the aging factor *D*.

STEP 2. Generate a completely coupled SNCL with N_0 nodes; calculate the initial strength value of edges and nodes according to (2) and (3). But when calculate the initial strength value of edges, the *NewNode* in (2) should be replaced by any already existing nodes. So the initial strength value of edges is $2/N_0$, the initial strength value of nodes is $2(N_0-1)/N_0$. Then calculate $P_{NewNode \to i}$.

STEP 3. Execute creating operator according to Definition 5, then execute dynamic updating operator according to Definition 6 to update the strength value of nodes /edges, degree of nodes and $P_{NewNode \to i}$.

STEP 4. Execute the aging operator, and then execute dynamic updating operator to update related data.

STEP 5. Judge the number of existing nodes N_{exist}, if $N_{exist} \geq N$, then go to STEP6; else go to STEP3.

STEP 6. Calculate the number of nodes and edges, average path length and clustering coefficient.

STEP 7. Finish executing the algorithm and output the relevant statistical data.

4 SEM-SN Model

In this section SEM-SN will be applied to simulate the evolution process of SNCLs obtained from eight open-source OO software systems including Freetype, gchempaint, 4tp, Prospectus and so forth. The statistics for these software systems are all chosen from [5]. The source code of these systems can be downloaded from [19]. And some useful tools [17] [18] can be used to extract the data listed in [5].

4.1 Parameter Setting

Table 1 shows the parameter settings for each experiment. The denotations in Table 1 denote the same meaning as talked above. Where N is set to be the number of classes in the real software systems; N_0 should be adjusted according to the size of software systems, its value between 3 and 12 in this paper; the value of m also should be adjusted according to the size of software systems, its value is between 2 and 10 in this paper; D is a real number between 0 and 1.

Table 1. Parameter settings for each experiment

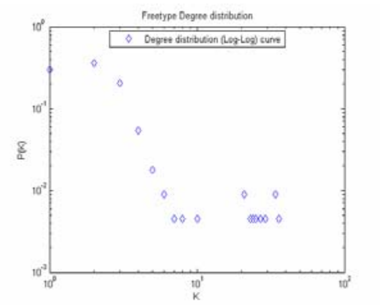
Dataset	N	N_0	m	D
Freetype	224	11	2	0.3
gchempaint	27	4	3	0.3
4yp	54	4	3	0.3
Prospectus	99	4	3	0.3
Openvrml	159	5	4	0.3
Dm	162	8	2	0.1
Yahoopops	373	8	3	0.45
Gpdf	162	4	3	0.3

4.2 Numerical Experiments

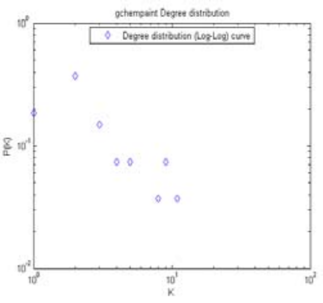
Apply SEM-SN to simulate the evolution process of eight open-source OO software systems and the results shown in Table 2. N_m is the number of nodes obtained by

Table 2. Data Comparision

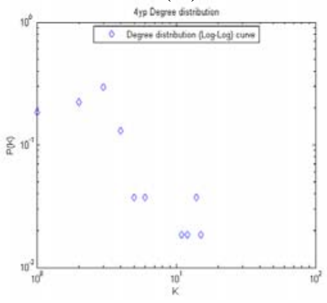
Dataset	N	N_m	L	L_m	d	d_{rand}	d_m	C	C_{rand}	C_m
Freetype	224	224	363	361	4.29	4.71	3.56	0.193	0.014	0.11
gchempaint	27	27	41	42	2.85	3.26	2.83	0.204	0.102	0.15
4yp	54	54	90	91	3.28	3.44	3.01	0.069	0.059	0.07
Prospectus	99	99	168	174	3.80	3.77	3.50	0.14	0.034	0.13
Openvrml	159	159	335	343	3.53	3.53	3.14	0.08	0.026	0.10
Dm	162	162	254	251	4.32	4.45	3.43	0.304	0.022	0.12
Yahoopops	373	373	711	712	5.57	4.47	3.35	0.336	0.01	0.12
Gpdf	162	162	300	293	4.02	3.93	3.33	0.303	0.022	0.10



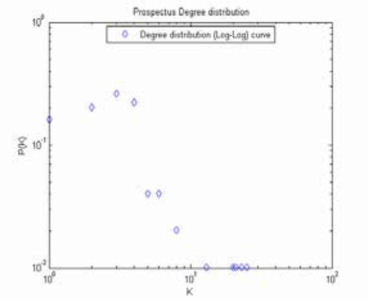
(A)



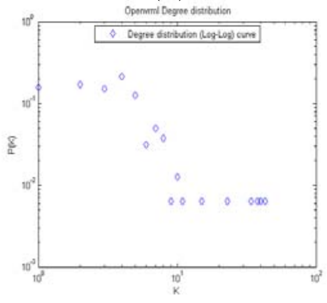
(B)



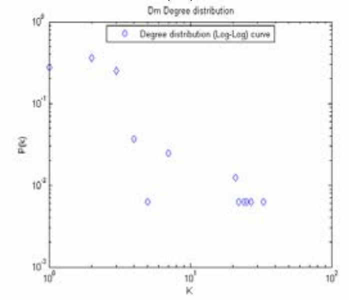
(C)



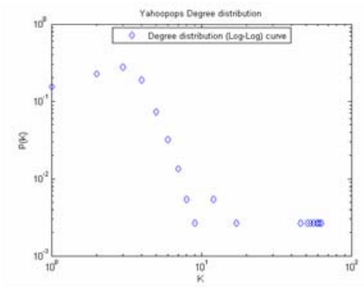
(D)



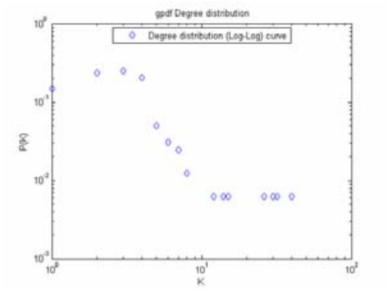
(E)



(F)



(G)



(H)

Fig. 2. Log-log plot of the degree distribution for each experiment

SEM-SN; L is the number of edges in real software systems, and L_m is that obtained by SEM-SN; d is the average path length in real software systems, d_{rand} is that in a random network, and d_m is that obtained by SEM-SN; C is the cluster coefficient in real software, C_{rand} is that in random network; C_m is that obtained by SEM-SN. Fig.2 shows the degree distribution curve which denoted by diamond in a log-log coordinate.

4.3 Analysis

We have measured N_m, L_m, d_m and C_m in all SNCLs generated by SEM-SN described above. Comparison with real software networks and random networks shows that the SNCLs generated by SEM-SN are instance of small-worlds (see Table 2). For every software networks, we have observed that $C_m >> C_{rand}$ and $d_m \approx d_{rand}$. Further research into the degree distribution of the eight OO software networks reveals that they both demonstrate the power law distributions (Fig.2), as the same results have been reported before [3] [4] [5]. What's more, N_m, L_m, d_m and C_m are very close to that in the real software networks, which demonstrates that SEM-SN can roughly describe the evolution process of real software networks. SEM-SN only needs to adjust three parameters and can get simulated software networks with similar properties as real software networks. We may roughly infer that the dynamical interaction between classes may contribute to the emergence of real software systems sharing some complex network characteristics.

5 Conclusions

In this paper we represent the software systems by software networks at class level, and study the forming mechanisms and evolution laws behind from nodes and their dynamical interactions, and finally propose SEM-SN model. In experiments, SEM-SN has been applied to simulate the evolution process of software networks at class level obtained from eight open-source OO software systems. The results show that the networks generated by SEM-SN roughly have the same characteristics as real software networks. Thus SEM-SN gives some insights to the emergence of complex network characteristics in real software systems.

Although SEM-SN shows some feasibilities in simulating the evolution process of some software networks obtained from real software systems, the broad validity of our model demands further demonstration. Thus, the future work include: (1) investing the relationships between the parameters in experiments through correlation analysis; (2) judging whether there exists a threshold in parameters for the simulated software networks to share complex networks properties; and (3) taking into consideration more other factors in real software development to more precisely simulate software development and reveal the evolution mechanisms of software systems.

Acknowledgements

This work is supported by National Basic Research Program (973) of China under grant No. 2007CB310801, National High Technology Research and Development

Program (863) of China under grant No. 2006AA04Z156, National Natural Science Foundation of China under grant No. 60873083 and 60803025, and Research Fund for the Doctoral Program of Higher Education of China under grant No. 20070486065.

References

1. Keqing, H., Yutao, M., Jing, L., Bing, L., et al.: Software networks. Science Press (2008) (in Chinese)
2. Valverde, S., Cancho, R., Solé, R.: Scale Free Networks from Optimal Design. Europhysics Letters 60, 512–517 (2002)
3. Myers, C.R.: Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. Physical Review E 68(4), 46116 (2003)
4. Potanin, A., et al.: Scale-free geometry in OO programs. Communications of the ACM 48(5), 99–103 (2005)
5. Valverde, S., Sole, R.V.: Hierarchical Small Worlds in Software Architecture. Arxiv preprint cond-mat/0307278 (2003)
6. Barrat, A., Barthelemy, M., Vespignani, A.: Weighted Evolution Networks: Coupling Topology and Weight Dynamics. Physical Review Letters 92(22), 228701 (2004)
7. Li, C., Chen, G.: A comprehensive Weighted Evolution Network model. Physica A 33, 288–294 (2004)
8. Li, M., Fan, Y., Wang, D., et al.: arXiv:cond-mat/0601495v1
9. Bing, L., Hao, W., Zhengyang, L., et al.: Software Complexity Metrics Based on Complex Networks. Acta Electronica Sinica 12(34), 2371–2375 (2006)
10. Solé, R.V., Ferrer, R., Montoya, J.M., Valverde, S.: Tinkering and Emergence in Complex Networks. Complexity 8(1), 20–33 (2002)
11. Kim, J., et al.: Infinite-order percolation and giant fluctuations in a protein interaction network. Physical Review E 66(5), 55101 (2002)
12. Sole, R.V., et al.: A Model of Large-Scale Proteome Evolution. Advances in Complex Systems 5(1), 43–54 (2002)
13. Valverde, S., Solé, R.V.: Network motifs in computational graphs: A case study in software architecture. Physical Review E 72(2), 26107 (2005)
14. Myers, C.R.: Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. Physical Review E 68(4), 46116 (2003)
15. He, K., Peng, R., et al.: Design Methodology of Networked Software Evolution Growth Based on Software Patterns. Journal of Systems Science and Complexity 19(2), 157–181 (2006)
16. Zheng-jun, P., Li-Shan, K., Yu-Ping, C.: Evolutionary Computation. Tsinghua University Press (1998)
17. Doxygen5.1, `http://www.stack.nl/~dimitri/doxygen/`
18. Dependency Finder, `http://depfind.sourceforge.net/`
19. Open-source software systems, `http://sourceforge.net/`