SCIENCE CHINA
Information Sciences

• INSIGHT •
Special Focus on Advanced Technology of Software
May 2016, Vol. 59 050101:1–050101:3
doi: 10.1007/s11432-016-5546-4

Growing construction and adaptive evolution of complex software systems

Huaimin WANG* & Bo DING

National Key Laboratory of Parallel and Distributed Processing, College of Computer, National University of Defense Technology, Changsha 410073*, China*

Received December 29, 2015; accepted January 29, 2016; published online April 8, 2016

Citation Wang H M, Ding B. Growing construction and adaptive evolution of complex software systems. Sci China Inf Sci, 2016, 59(5): 050101, doi: 10.1007/s11432-016-5546-4

Distributed software systems are becoming more and more complex today. It is easy to find a huge amount of computing nodes in a nationwide or global information system. For example, WeChat (WeiXin), a well-known mobile application in China, has reached a record of 650 million monthly active users in the third quarter of 2015. At the same time, researchers are starting to talk about software systems which have billions of lines of codes [1] or can last one hundred years.

The complexity of distributed software systems can be measured not just by the scale. Those software systems also exhibit a set of unique characteristics in terms of both the internal organization and the external interaction. A new kind of software system, the complex software system, emerged in practice. In short, a complex software system is a system which consists of a large amount of autonomic software systems via coupling and interconnection. It has the following three characteristics: (1) It is a "system of systems" [2], which means that it consists of a set of independent (or semi-independent) systems, and the interaction among them is dynamic and complex. Its behavior is not a simple linear combination of the behavior of those subsystems. (2) It is a "cyber-physical system" [3], which means that it is closely bound to the physical world. For example, a minor software malfunction in many mission-critical systems (such as a smart grid system) may directly result in disastrous consequence in the real world. (3) It is a "socio-technical system", which means that the boundary between the people and the software is becoming increasingly blurred. People and other social factors are being a part of the software itself.

In the 1960s, the software engineering discipline was spurred by the so-called software crisis [4]. Today, another crisis emerged along with the appearance of complex software systems. Since those systems have the features of "systems of systems", "cyber-physical systems" and "socio-technical systems", many assumptions in traditional software engineering are no longer valid. As a "cyber-physical system" and a "socio-technical system", it is impossible to predict the user requirements and the running environment of a complex software system precisely in advance. Thus, the "top-down decomposition" strategy (such as the well-known waterfall model [5] in traditional software engineering) cannot be applied. Besides, a traditional software product can be regarded as a linear combination of its sub components and thus it can be developed by the "bottom-up assembly" strategy. However, the relationships among the subsystems of a complex software system are more

* Corresponding author (email: whm_w@163.com)
The authors declare that they have no conflict of interest.

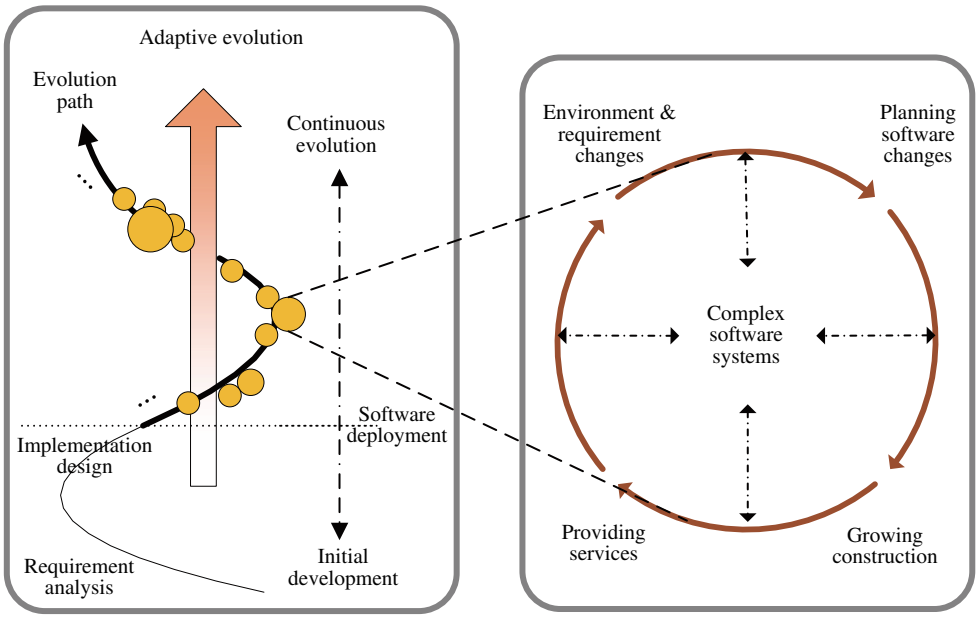

Figure 1 (Color online) Relationships between growing construction and adaptive evolution.

dynamic and unpredictable.

Complex software systems present a great challenge to traditional software engineering methodology, i.e., how to construct such kind of systems and ensure its reliable operation at runtime. To address this challenge, we should learn the experience from the formation of complex systems in the real world [6]. Taking a big city (such as Beijing) as an example, it cannot be built by the "top-down decomposition, bottom-up assembly" strategy. We cannot design the whole city in advance and then construct the buildings one by one. Instead, the city "grows" as a result of the loosely coordinated and regulated actions of many individuals such as companies or organizations. The most important thing in this process is to introduce certain mechanisms to enable the growth, such as appropriate city plans or well-defined laws/rules. And to ensure the sustainable development of the city, those mechanisms have to "evolve" continuously to adapt the city to the social and economic changes in the world.

Similarly, we propose two important principles with regard to complex software systems. The first one is "Growing Construction". This principle can be described briefly as follows: a complex software system is formed in the process of dynamic connection of many autonomous systems. It lays emphasis on two points: (1) A complex software system cannot be built totally from scratch by traditional engineering approach. Instead, it grows gradually. (2) The "growing" is realized by dynamic connection of existing or newly-developed autonomous systems. The second one is "Adaptive Evolution", i.e. a complex software system is continuously evolved by adapting itself to the changes in its environment and the user requirements. It lays emphasis on two points as well: (1) Evolution is a common phenomenon in complex software systems, and thus we should embrace software changes instead of fighting them. (2) The major factors driving the evolution process are the changes in the environment and the user requirements. The relationship between those two principles can be illustrated by Figure 1. From the perspective of realization, the growing construction principle mainly concerns constructing an evolvable software system, and the adaptive evolution principle mainly concerns how to drive the evolution process.

In the past decades, we have witnessed many successful practices in software engineering discipline, such as middleware [7], Internetware [8] and adaptive software [9]. Those practices pushed us toward the goal of realizing complex software systems. At the same time, some initial discussions directly towards the complex software system have emerged [10, 11]. However, the related theory and techniques are far from mature, and there are still a lot of scientific challenges which should be addressed. As regarding to the growing construction principle, we should find a model (or a set of models) to abstract the growth process of complex software systems as well as the practicable development method to rectify this model. And as regarding to the adaptive evolution principle, there are also many open problems, such as what kind of common patterns can be observed in the

evolution process, how can we gather the environmental and the internal states of a complex software system [12], how can the online evolution actions be enforced, and how can we conduct the co-evolution between the software system and the physical world it resides in.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 91118008, 61202117).

References

1 Northrop L, Feiler P, Gabriel R, et al. Ultra-Large-Scale Systems: the Software Challenge of the Future. Carnegie Mellon University Technical Report. 2006
2 Maier M W. Architecting principles for systems-of-systems. Syst Eng, 1998, 1: 267–284
3 Lee E A. Cyber-physical systems-are computing foundations adequate. In: Proceedings of NSF Workshop on Cyber-Physical Systems: Research Motivation, Techniques and Roadmap, Austin, 2006. 1–9
4 Naur P, Randell B. Software Engineering: Report of a Conference Sponsored by the NATO Science Committee. NATO Technical Report. 1969
5 Boehm B. A view of 20th and 21st century software engineering. In: Proceedings of International Conference on Software Engineering, Shanghai, 2006. 12–29
6 Holland J H. Hidden Order: How Adaptation Builds Complexity. New York: Perseus Books, 1995
7 Wang H M, Wang Y F, Tang Y B. StarBus+: distributed object middleware practice for Internet computing. J Comput Sci Tech, 2005, 20: 542–551
8 Mei H, Huang G, Zhao H Y, et al. A software architecture centric engineering approach for Internetware. Sci China Ser F-Inf Sci, 2006, 49: 702–730
9 Wang H M, Ding B, Shi D X, et al. Auxo: an architecture-centric framework supporting the online tuning of software adaptivity. Sci China Inf Sci, 2015, 58: 092103
10 Sommerville I, Cliff D, Calinescu R, et al. Large-scale complex IT systems. Commun ACM, 2012, 55: 71–77
11 Nielsen C B, Larsen P G, Fitzgerald J, et al. Systems of systems engineering: basic concepts, model-based techniques, and research directions. ACM Comput Surv, 2015, 48: 18
12 Mi H B, Wang H M, Zhou Y, et al. Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems. IEEE Trans Parall Distr Syst, 2013, 24: 1245–1255