Thoughts on Application of Evolutionary Software Development for Complex, Large-scale, Integrated, Open Systems

Joe Manganelli[1,2] , J.B.F. Mulder[3]

[1] Fluor Enterprises, Inc., 100 Fluor Daniel Drive, Greenville, SC, 29607, USA
[2] Kent State University, 1125 Risman Drive, Kent, OH, 44242, USA
[3] University of Antwerp, Belgium
{joe.manganelli@fluor.com; jmangane@kent.edu; hans.mulder@uantwerpen.be}

Abstract. This paper presents two thought experiments that consider the potential benefits, concerns, and questions of evolvable software developed using Normalized Systems Theory when applied to the emerging complex, interactive project types known as cyber-physical systems (CPS), socio-technical systems (STS), ultra-large scale systems (ULS), and Complex, Large-scale, Integrated, Open Systems (CLIOS).

Keywords: evolutionary software, normalized systems theory, cyber-physical systems, socio-technical systems, complex systems.

1 Introduction

Business-critical applications fail to keep up with current technology and deteriorate in functionality and performance over time [1]. In addition, the upkeep and maintenance of these applications is costly and risky. This is not new. As far back as the 1970s, Prof. Dr. Manny Lehman proposed his Law of Increasing Complexity, stating, “As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it.” [2] Normalized Systems Theory enhances the software development process to not only accommodate change, but also to promote change [3]. However, emerging project types are orders of magnitude more complex and integrated than today’s large, complex project types. The scale, complexity, and integrated-ness of emerging project types present new design and management challenges and must be cultivated.

1.1 Literature Review: Evolutionary Software Development

Normalized Systems Theory has been developed to build systems that are immune to the Increasing Complexity law—in other words, systems for which the impact of changes is proportional to the additional functionality, not to the size of the existing system. As a result, some related properties like scalability, reliability, and testability can be achieved. Indeed, applications conforming to this theory can become very large and complex, without restraining the adaptability over time. [4] This research is based on applying well-known engineering knowledge from other domains, such as the concepts of *stability* from systems theory and *entropy* from thermodynamics [5] and has already received several best paper and industry awards [2]. Results show that it is now feasible to realize both an increase in productivity by several factors in the development of transaction systems, as well as attain an unprecedented level of control during maintenance. At the same time, performance testing has exhibited excellent results, showing that there is no fundamental trade-off between performance and modularity, as is often assumed. Fine-grained modular structures combined with highly systematic development processes offer both [6]

1.2 Literature Review: Complex, Large-scale, Integrated, Open Systems

The early 2000's yielded a proliferation of emerging project types for hardware and software systems that entail unprecedented scale, complexity, integrated-ness, and real-time interactivity. As a result, the U.S. Army engaged the Software Engineering Institute (SEI) to conceptualize interconnected systems of people, software, machines, and data so complex that they are, "…likely to have billions of lines of code…"and to identify methods and tools for designing and operating such systems. SEI produced a technical report, titled, *Ultra-Large-Scale Systems: The Software Challenge of the Future* [7] that found that existing engineering methods and tools are not suited to develop and operate large-scale systems of systems projects. [8] Ultra-large-scale systems will exist on a scale and of a complexity that is impossible to fully comprehend, model, or control. Ultra-large scale systems will be composed of other emerging, complex systems of system types, including: cyber-physical systems,[9], [10], [14] socio-technical systems,[11] complex, large-scale, integrated, open systems (CLIOS),[12], [15] and multi-scale systems (MSS).[13] The primary differentiating characteristics of these emerging systems of systems types are:

- "A component of a larger complex/interactive systems of systems while being composed of systems of systems;
- Real-time hardware/software interactions amongst and between internal and external systems to function successfully; and
- Real-time human-machine-software interactions are essential to meeting user goals and expectations."[16].

SEI noted that methods and tools to design and specify ultra-large scale systems may never fully exist, but that methods and tools from the building development and urban planning communities are instructive of how to design, build, and maintain systems that are too large to fully abstract. These methods are similar in conceptual approach to the use of Normalized Systems Theory for guiding software development in that building development and urban planning methods reduce the degrees of freedom of the complex systems through rule-based strategies, so that simple representations and analyses suffice to model and predict the performance of complex systems.

2 Current Context: Normalizing the complexity of current software design is the major challenge for the next decade

The Standish Group has been formally researching the causes of software project success and failure since 1994 [1], [17]. The efforts at the University of Antwerp in developing Normalized Systems Theory [2] and The Standish Group's CHAOS research on project performance provides some very interesting mutual findings. The dovetail of these two efforts provides a concrete approach to modernize and create large-scale business applications and systems. In the 1996 The Standish Group introduced the iterative process. The iterative process is at the heart of all of the agile process methods, including Scrum. The Normalized Systems approach provides for applications and systems to be able to evolve over time in an iterative fashion. [18] The combination of Scrum and Normalized Systems creates a pipeline of nanoprojects which allows for dynamic, stable applications and systems. Preliminary results show that it can be feasible to realize both an increase in productivity by several factors in the development of transaction systems, as well as attain an unprecedented level of control during maintenance. At the same time, performance testing has exhibited excellent results, showing that there is no fundamental trade-off between performance and modularity, as is often assumed. Fine-grained modular structures combined with highly systematic development processes offer both. [19]

3 Challenge on the Horizon: How does one develop an evolvable software paradigm to handle a higher order of openness, integration, complexity, and interactivity?

Within the next decade, the nature of the software development process must address continuous, ever-evolving integration with an open and unknown network of other software, databases, sensors, actuators, users, use cases, and deployment conditions. As a point of reference, the current state of the practice for a large-scale an example is given of point of sale system of systems which is deployable on traditional point of sale terminals, as well as smartphones and tablets used by customers, sales personnel, and logistics, so that every phone and tablet is a point of sale device.

But the next evolution of such retail systems entails orders of magnitude more integration with real-time user data analysis, and this creates the condition whereby the fundamental nature of developing the software system of systems changes. In the near future, these systems of systems will enhance their predictive analytics by accessing and analyzing real-time user biometric data, location data, environmental preferences settings (whether in car, building, or outdoors), time of day, weather, season, local cultural events, schedule, health, email, social media, what dominates the news cycle for each individual user, typical preferences and trends within the top ten social/economic communities within which the user participates, as well as all of these data points for the twenty other people closest to the user of interest. Data will be pulled from an unknown number of systems and databases but that may be estimated to number in the low tens of thousands.

A point of sale system of systems, with so much potential to manipulate and exploit the user, may be governed by hundreds of sets of regulations and that at any given moment the system of systems is undergoing dozens of separate audits from private, investor, and regulatory agents, both human and software-based. Furthermore, while the software of interest is evolving, all of these other systems of information with which it interacts will also be constantly evolving.

4 Thoughts on Possible Benefits, Concerns, Questions

The benefits of developing software for such complex, interactive open systems of systems of software, sensors, actuators, and people are overall effectiveness, efficiency, and resiliency of information systems used to support smart cities, commerce, security, governance, and human health, well-being, and productivity. Evolvable software systems based upon Normalized Systems theory can potentially bound the complexity of software systems of systems and optimize modularization, thereby limiting unintended consequences that result from problematic emergent systems properties.
This becomes a question of how such a system of systems is abstracted within a representational framework via sets of constructs and about how those constructs and that framework is operationalized.

5 Suggestions for Research Trajectories

To engage a software development challenge such as described above, the software development process will have to compose integrated systems of evolvable software systems. That is, an ecosystem that exhibits something like evolutionary biological processes as a unit --- a biosphere --- an ecological niche [20] will have to be *cultivated*. Research trajectories may include developing software agents that self-organize into social systems and that interact with other self-evolving software species. In addition, these software agents must be placed into the same representational framework as biological agents so that their interactions may be represented and analyzed. Therefore, attempts should be made to extend Linneas' system of biological taxonomy to include non-biological, naturally evolving agents. The future of software development entails cultivating self-evolving, cognizing software agents that manifest varying degrees of personhood and function in symbiosis with humans within a share social/biological/technological ecological niche [21]. Philosophical and legal discourse on personhood and the constructs, theories, methods, and tools of bio-engineering, evolutionary biology, and socio-technical systems development may be instructive.

Acknowledgments. The authors thank Tom MacKnight and Scott Carlson of Fluor for feedback

References

[1] Chaos, tech. report, Standish Group Int'l, 1994.

[2] Lehman, Meir M. (September 1980), "The Law of Increasing Complexity," Proceedings of the IEEE 68 (9), p. 1068.

[3] Mannaert, Herwig; Verelst, Jan (2009). "Normalized Systems: Re-creating Information Technology Based on Laws for Software Evolvability." Koppa. ISBN 978-90- 77160-00-8.

[4] Oorts, G et al., Easily Evolving Software Using Normalized System Theory A Case Study, ICSEA 2014 : The Ninth International Conference on Software Engineering Advances

[5] Herwig Mannaert and Jan Verelst. Normalized Systems - Re-creating Information Technology Based on Laws for Software Evolvability, Koppa, 2009. ISBN: 978-90-77160-008.

[6] Op 't Land, M et al., Exploring Normalized Systems Potential for Dutch MoD's Agility, PRET 2011, LNBIP 89, pp. 110–121,Springer-Verlag Berlin Heidelberg 2011

[7] Pollak, B (Ed.), Feiler, P., Gabriel, R., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Northrop, L., Schmidt, D., Sullivan, K., Wallnau, K. (2006). Ultra-Large-Scale Systems: The Software Challenge of the Future. Software Engineering Institute. Retrieved from: http://resources.sei.cmu.edu/asset_files/Book/2006_014_001_30542.pdf

[8] Manganelli, J. (2013). Designing complex, interactive, architectural systems with CIAS-DM: A model-based, human-centered, design & analysis methodology. Pp. 64-89. (dissertation (Order No. 3609779, Clemson University)). ProQuest Dissertations and Theses, 690. Retrieved from http://search.proquest.com/docview/1499237325?accountid=6167. (1499237325) Retrieved from: http://tigerprints.clemson.edu/cgi/viewcontent.cgi?article=2250&context=all_dissertations.

[9] Lee, E. (2008). "Cyber-physical systems: Design challenges," International Symposium on Object/Service-Oriented- Real-Time Distributed Computing (ISORC). May 6, 2008, Orlando, FL, USA. Retrieved from: http://chess.eecs.berkeley.edu/pubs/427/Lee_CyberPhysical_ISORC.pdf

[10] Xie, F. (2006). "Component-based cyber-physical systems," in NSF Workshop on Cyber-Physical Systems, Austin, TX, 2006. Retrieved from: http://varma.ece.cmu.edu/CPS/Position-Papers/Fei-Xie.pdf

[11] Fischer, G., Hermann, T. (2011). "Socio-technical systems: A meta-design perspective," International Journal for Socio-technology and Knowledge Development, vol. 3, no. 1, pp. 1-33, 2011.

[12] Dodder, R., Sussman, J., McConnell, J. (2004). The concept of the "CLIOS process": Integrating the study of physical and policy systems using Mexico City as an example, Cambridge, MA: MIT ESD, 2004. Retrieved from: http://web.mit.edu/mtransgroup/presentations/pdf/CLIOS%20Process%20Concept.pdf

[13] Kevrekidis, I., Gear, C., Hummer, G. (2004). "Equation-free: The computer-aided analysis of complex multiscale systems," AIChE Journal, vol. 50, no. 7, pp. 1346-1355.

[14] see 10.

[15] see 12.

[16] see 8.

[17] Johnson, J., Mulder, J.B. F. (2011). Nonstop Modernization: Modernize in Place. Retrieved from: http://blog.standishgroup.com/post/27

[18] Mulder, J.B.F., J. Johnson, The Marriage of CHAOS Research with Normalized Systems Theory. IMSCI 2016, Orlando, Florida, USA; 07/2016

[19] Mannaert, H., J. Verelst, P. De Bruyn. (2016). Normalized Systems Theory From Foundations for Evolvable Software Toward a General Theory for Evolvable Design. ISBN: 978 90 77160 091 D/2016/9617/1 NUR: 980, 994

[20] Odling-Smee, F., Laland, K., Feldman, M. (2003). Niche Construction: The Neglected Process in Evolution. Princeton, NJ: Princeton University Press.

[21] Manganelli, J. (2015). "Tending the Artifact Ecology: Cultivating Architectural Ecosystems." Retrieved from: http://datastructureformdesign.com/2015/10/04/tending-the-artifact-ecology-cultivating-architectural-ecosystems/