From Formal Methods to Software Components: Back to the Future?

Kung-Kiu Lau(✉)

School of Computer Science, The University of Manchester,
Manchester M13 9PL, UK
kung-kiu.lau@manchester.ac.uk

Abstract. Looking back at the past, I believe Formal Methods and Component-based Software Engineering have missed opportunities to synergise. Looking forward to the future, I believe even more strongly that this synergy will be crucial for developing Software Engineering techniques that tackle scale and complexity. In this position paper I outline the fundamentals of my belief, in terms of existing work and future challenges.

1 The Future?

Any engineering discipline is based on: (i) a well-established underlying theory; (ii) standard parts or *components* for building systems; and (iii) tools for *constructing systems* from components, and for *verifying systems*. So it would seem logical to conclude that Component-based Software Engineering (CBSE) and Formal Methods (FM) are the essential ingredients for Software Engineering.

As software becomes ever more pervasive (witness the Internet of Things), the challenge facing Software Engineering nowadays is how to tackle ever increasing scale and complexity, while guaranteeing safety. Is CBSE + FM up to the challenge? Is CBSE + FM addressing the challenge? To answer in the affirmative, I believe we need to accomplish two things: (i) compositional construction; and (ii) compositional verification.

2 Compositional Construction

Compositional construction is what CBSE sets out to achieve. The general picture of CBSE is depicted in Fig. 1. The basic idea is that components should preexist, i.e. they should be built independently from specific systems and deposited in a repository. Repository components can be reused in many different systems constructed by composing the components.

Whilst a generic component (Fig. 2(a)) is a unit of composition with provided and required services, commonly used components fall into three main categories: (i) objects (Fig. 2(b)); (ii) architectural units (Fig. 2(c)); and (iii) encapsulated components (Fig. 2(d)). Composition mechanisms (Fig. 3) used by these categories are respectively: (i) direct message passing (method call); (ii) indirect message passing (port connection); (iii) coordination (exogenous composition).

O. Kouchnarenko and R. Khosravi (Eds.): FACS 2016, LNCS 10231, pp. 10–14, 2017.
DOI: 10.1007/978-3-319-57666-4_2

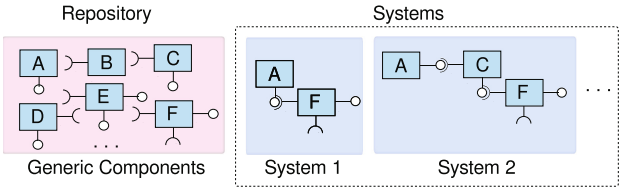

Fig. 1. CBSE: compositional construction.

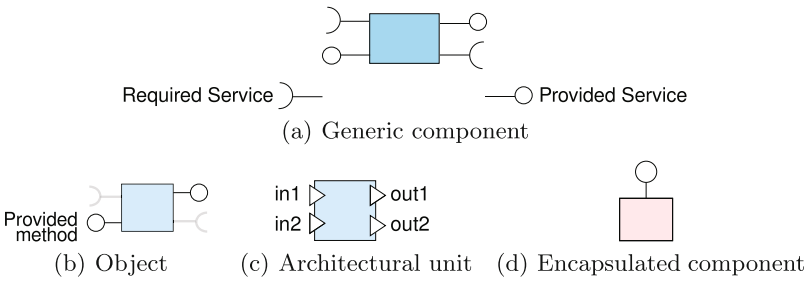

Fig. 2. Types of components.

Components	Provided services	Required services	Composition mechanism
Objects	Methods	——	Method call
Architectural units	Out-ports	In-ports	Port connection
Encapsulated components	Methods	*None*	Exogenous composition

Fig. 3. Composition mechanisms.

The desiderata for compositional construction are embodied in the *idealised component life cycle* [1], illustrated in Fig. 4. Apart from the use of pre-existing components from a repository, the key desiderata include composition in both the design phase and the deployment phase;[1] since maximum composition equates to maximum reuse.

2.1 Component Models

For compositional construction, components and their composition mechanisms [5] have to be defined properly. We advocate to do so in a *component model* [9,10]. Figure 5 shows a taxonomy of current component models with respect to the idealised component life cycle. Categories 1–4 do not support composition in both design and deployment phases. Category 5 does, but has only a lone member, namely X-MAN [2,7,8,12], that we have defined and implemented. X-

[1] Run-time composition, or dynamic reconfiguration, is also meaningful, though it may be harder to define, implement and verify.

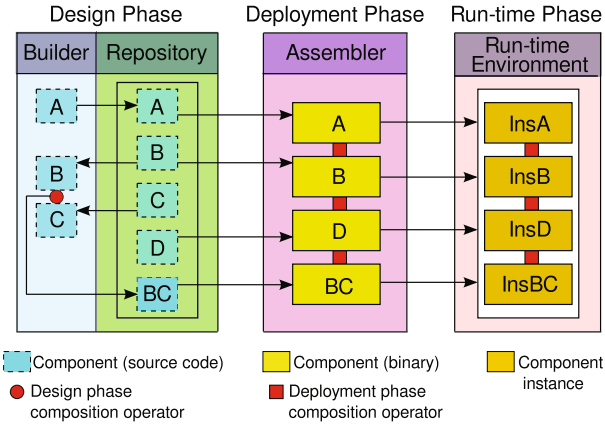

Fig. 4. Idealised component life cycle.

MAN achieves compositional construction, but it currently lacks tool support for compositional verification [3].

3 Compositional Verification

With compositional construction, we should be able to accomplish compositional verification, i.e. hierarchical, bottom-up, verification of component-based systems whereby the smallest (atomic) components at the lowest level are verified first, and their verification is reused, i.e. not repeated, in the verification of a composite at the next level up.

This is illustrated in Fig. 6, which shows the W model [6] for component-based development life cycles: component life cycle and system life cycle, and how they intersect. The W model supports compositional verification. Component verification is done in the component life cycle when components are developed for the repository, independent of specific systems. Compositional verification is done when a specific system has been assembled from already verified repository components. By reusing the verification of sub-components at successive levels of composition, instead of verifying the complete system as a monolith, compositional verification should be able to scale to large complex systems which are beyond the capability of current verification techniques and tools.

4 Back to the Future?

Looking back, I advocated synergy between FM and CBSE [4] at the early stages of the International CBSE Symposium. In my opinion, hitherto this synergy has not really materialised, or at least what little there is has not been effective. According to [11], there has been little FM activity at the CBSE symposium. I

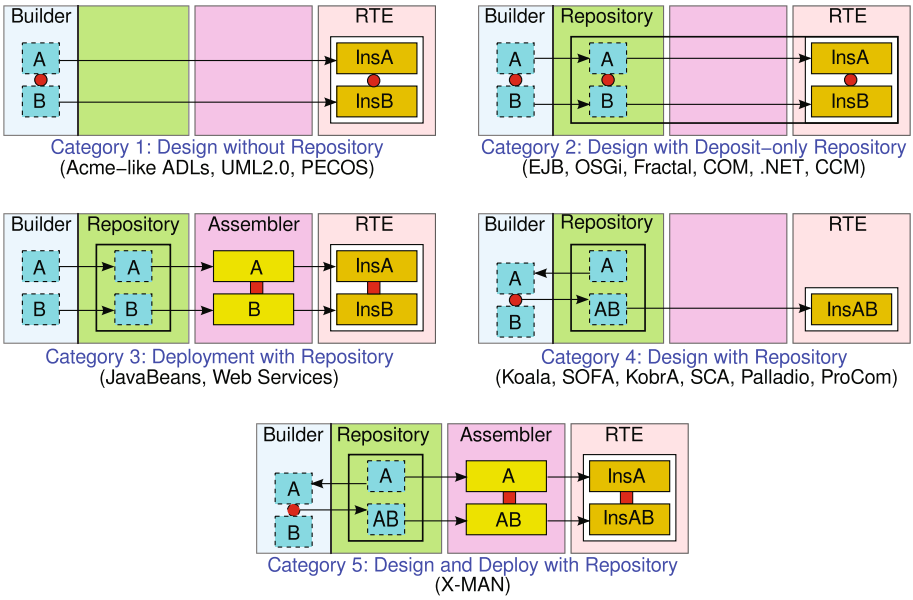

Fig. 5. Idealised component life cycle: taxonomy of component models.

surmise the converse is true of CBSE activity at FM conferences – maybe even FACS?

Looking forward to the future, I strongly believe that this synergy will be crucial for developing Software Engineering techniques that are not only truly engineering techniques as in traditional engineering disciplines, but can also tackle scale and complexity. In other words, this synergy can provide not only an engineering (compositional) approach to software construction from standard parts, but also compositional reasoning, which together can tackle ever increasing scale and complexity in software systems and their V&V.

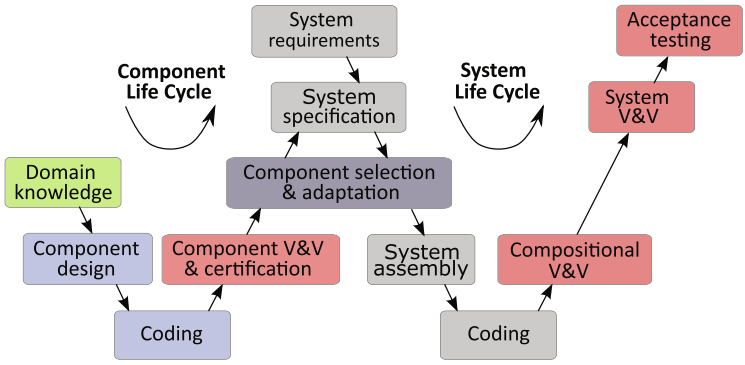

Fig. 6. The W model.

References

1. Broy, M., Deimel, A., Henn, J., Koskimies, K., Plasil, F., Pomberger, G., Pree, W., Stal, M., Szyperski, C.: What characterizes a software component? Softw. Concepts Tools **19**(1), 49–56 (1998)
2. di Cola, S., Tran, C., Lau, K.-K.: A graphical tool for model-driven development using components and services. In: Proceedings of 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA) 2015, pp. 181–182 (2015)
3. He, N., Kroening, D., Wahl, T., Lau, K.-K., Taweel, F., Tran, C., Rümmer, P., Sharma, S.: Component-based design and verification in X-MAN. In: Proceedings of Embedded Real Time Software and Systems (2012)
4. Lau, K.-K.: Component certification and system prediction: is there a role for formality? In: Crnkovic, I., Schmidt, H., Stafford, J., Wallnau, K. (eds.) Proceedings of the Fourth ICSE Workshop on Component-based Software Engineering, pp. 80–83. IEEE Computer Society Press (2001)
5. Lau, K.-K., Rana, T.: A taxonomy of software composition mechanisms. In: Proceedings of 36th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 102–110. IEEE (2010)
6. Lau, K.-K., Taweel, F., Tran, C.: The W model for component-based software development. In: Proceedings of 37th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 47–50. IEEE (2011)
7. Lau, K.-K., Tran, C.: X-MAN: an MDE tool for component-based system development. In: Proceedings of 38th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 158–165. IEEE (2012)
8. Lau, K.-K., Velasco Elizondo, P., Wang, Z.: Exogenous connectors for software components. In: Heineman, G.T., Crnkovic, I., Schmidt, H.W., Stafford, J.A., Szyperski, C., Wallnau, K. (eds.) CBSE 2005. LNCS, vol. 3489, pp. 90–106. Springer, Heidelberg (2005). doi:10.1007/11424529_7
9. Lau, K.-K., Wang, Z.: Software component models. IEEE Trans. Softw. Eng. **33**(10), 709–724 (2007)
10. Lau, K.-K., Wang, Z., di Cola, S., Tran, C., Christou, V.: Software component-models: past, present and future. In: Tutorial at COMPARCH 2014 Conference, 30 June 2014, Lille, France (2014)
11. Maras, J., Lednicki, L., Crnkovic, I.: 15 years of CBSE symposium - impact on the research community. In: Proceedings of the 15th International ACM SIGSOFT Symposium on Component-Based Software Engineering, pp. 61–70. ACM (2012)
12. Elizondo, P.V., Lau, K.-K.: A catalogue of component connectors to support development with reuse. J. Syst. Softw. **83**, 1165–1178 (2010)

http://www.springer.com/978-3-319-57665-7

Formal Aspects of Component Software
13th International Conference, FACS 2016, Besançon, France, October 19-21, 2016, Revised Selected Papers
Kouchnarenko, O.; Khosravi, R. (Eds.)
2017, XVIII, 281 p. 104 illus., Softcover
ISBN: 978-3-319-57665-7