

System Composition Strategies, A Position Paper for an ECBS Panel

My Position: Analogy questionable, intentions estimable.

John Leaney

Computer Systems Engineering Group Faculty of Engineering
University of Technology, Sydney Australia jrleaney @eng.uts.edu.au

Abstract

There can surely be no doubt that appropriate, automated design techniques are essential for the success of any complex computer based systems. No computer system can succeed unless the complexity is well managed. The techniques cited in the brief are fundamental for the management of complexity and the development of computer based systems. My position is that automated system composition is inevitable if computer based systems are to be well engineered.

However, I question the analogy of software as glue, and the goal of a single integrated system.

Software is more than glue, it is function providing. The work of other branches of engineering demonstrates the importance of automated design techniques to gain success in complex systems. The architecture of separate autonomous components, managed over a reliable infrastructure is proposed in contrast to a single integrated system.

1. Introduction

Australia is gripped by national and state elections all around the country; the USA is still recovering from an election. Europe is mercifully without them – not so Israel. Indonesia has its turmoil, China thinks. In that light I present a position paper.

Following is the position brief the panel received for discussion, and upon which I have come to the position described in this position paper with respect to the position expressed in the position brief.

“Software typically serves as the “glue” in CBSs, coupling the physical processes into a single integrated system. To ensure consistency across subsystems, and to facilitate the dynamic composition and re-composition of systems, requires highly automated system/software synthesis and integration technologies. These include simulation-based design, methods for systematic handling of the many goals and constraints impinging on the design process, performance evaluation techniques, trade-off measurements, evaluation of multi-level multi-component hierarchically specified models, and experimental frameworks.”

1.1 Composition vs. Integration

The title “System Composition Strategies” imbues hope, compared with the integration strategies often employed in practice. System composition suggests a focus upon the whole system from its inception, not “how will we put this all together to deliver something” at the end of a contract. In other words, it suggests the system is/will be engineered.

2. On software glue and single systems

The following is a bit pedantic, and I risk the ire of my colleagues, but I (obviously) feel it is important to this panel discussion.

The panel brief says ““Software typically serves as the “glue” in CBSs, coupling the physical processes into a single integrated system.” I think this statement needs discussion.

2.1 Software as glue

It is true, that software is glue – it makes us blue, when the new (software) is not true, and comes unstuck. In practice, too often software is glue. It holds subsystems together, and to pull them apart in order to recompose, replace, or rebuild the system, parts of the system are often broken. There are hundreds of glue tendrils sticking the pieces of the system together, and it becomes rather like pulling sausage out of a pizza. Unlike removing the cabanossi, reworking most computer based systems is not a pleasant experience.

2.2 Software as function

As well as glue, I would accentuate the role of software as providing, or delivering, function. In 1990 the European Commissioner on Technology [3] stated that by the year 1995, 80% of the *total* cost in world-wide manufacturing would be software. Some telecommunications companies put it at 95% in the year 2000. What is all that software providing? (Not much, some would say). The software is providing the function of the system, it is achieving the goals of the system.

A dramatic example is the steering mechanism of a motor car. Motor car makers are currently replacing the mechanical steering gear, including rack and pinion, compensation, stabilisation, etc, with two electric motors and a computer. All of the function which is the steering

mechanisms of an automobile will now have to be realised in software. I hope the software is not glue.

2.3 What is a system

Bunge [2] has given a simple and insightful way to identify what is (not) a system. If the output of the proposed system can be generated by a linear combination of the signals emanating from its components, then the collection of components and connections is not a system. The corollary proves a system.

2.4 A single integrated system

To say that the software, having glued the components together, creates a single, integrated system does not accord with either my observations, or the precious little theory that exists in systems theory. It may be true for relatively small systems, or the subsystems of a large system. Coherence in large systems is rarely achieved, and, from a human point of view rarely desirable. A model which federates, or loosely combines, a number of smaller systems, to effect a larger system seems a more tractable problem to solve.

2.5 Open systems

Elsewhere in these proceedings are papers which deal with definitions and details of open systems [5]. For the purposes of this argument, open systems are systems whose function performing, goal achieving components are connected together by an *open* infrastructure. The components are very loosely coupled to each other (eg., by data coupling); the infrastructure provides the communications. The infrastructure adheres to well established, widely used standards at all its interfaces. Open systems in engineering have come about as a response to the failure of systems which were glued together, and suffered from the glue analogy played upon earlier [1,4]. Open systems are automated in so far as a compiled interface is managed at runtime. The work by the Open Systems Group on CORBA [6], a well respected open system specification, supports a variety of automation techniques for the infrastructure. Even open systems suffer from an absence of good experimental frameworks and design techniques.

2.6 Automation and engineering

There is little doubt that other forms of engineering have increased their customer satisfaction as a result of automation. VLSI is an obvious field. but it's function span is much smaller than software systems. Software functionality is the functionality of the near future and the present; to be reliable it needs to have the level of automation, of high level control that motor cars and aircraft have in their chassis (or airframes, if one must) and

engines. The paradox is that the functionality of the component parts, including the engines, is now realised in software. Glue might hold them together.

2.7 Functional Performance

If systems are to be reconfigured dynamically, then work needs to be done on automating the determination of the functional performance of the resulting systems, and bounding that performance. One may not just call side effects an interesting new feature.

2.8 Autonomous components

I do not wish to be excessively anthropomorphic, but it seems to me that some of the more successful natural systems depend upon a considerable degree of autonomy for their success. Autonomy of components is a matter that we have not perhaps sufficiently addressed in considering system success.

3. Conclusion

Computer based systems cannot be considered engineered until automated techniques, comparable to the rest of the engineering world, are in common practice. We (at ECBS) would do well to fashion the goals and plans of such endeavours.

4. Acknowledgements

The discussions and patience of my colleagues at ECBS, and much fruitful (especially grape) work with Tim O'Neill and David Rowe, has contributed to this position.

5. References

- [1] Boeing Company, "Soaring into the Open", <http://www.opengroup.org/comm/case-studies/boeing.htm>
- [2] Bunge, M., Treatise on basic Philosophy, Vol 4. Ontology II: A World of Systems, D. Reidel Publishing, Boston, 1979
- [3] European Commissioner on Technology, Keynote Talk, ICSE, Nice, France, 1990.
- [4] Lawson, B, Leaney, J., and O'Neill, T., "Open Systems: towards safe, reliable systems", Proceedings of the 12th IEEE Conference and Workshop on Engineering of Computer-Based Systems, USA, 1999.
- [5] Leaney, J. , O'Neill, T., Rowe, D., "Effectiveness of Computer Based Systems: An Open System Measurement Example", Proceedings of the 14th IEEE Conference and Workshop on Engineering of Computer-Based Systems, Washington, USA, 2001.
- [6] Object Management Group, "The Common Object Request Broker: Architecture and Specification", Version 2.0., Framingham, MA, 1996. URL: <http://www.omg.org>