COTS Software Selection: The Need to make Tradeoffs between System Requirements, Architectures and COTS/Components

Cornelius Ncube & Neil Maiden

Centre for HCI Design
City University London
Northampton Square
London EC1V OHB
Tel: 020 7477-8412
Fax: 020 7477-8859
E-mail: [c.ncube@soi., n.a.m.maiden@]city.ac.uk

Abstract

This short paper presents a new research agenda to address problems of COTS software selection in the forthcoming decade. It describes the increasing shift towards software engineering based on COTS software packages, the limitations of current COTS/component-based software engineering methods and research efforts, and proposes a new research agenda to address the problems which arise from a software engineering process based on the reuse of COTS software and software components. The focus of the proposed research is how can we develop complex software systems by integrating together different combinations of COTS software packages and software components.

1. The Changing Face of Software Engineering

There are few methods, guidelines or environments that support the selection of COTS and component software. This is an increasing problem. Forrester Research estimate that 70% of European software development will be component- or COTS-based by 2003. The coverage of the problem is also large. In business systems engineering, enterprise integration applications will need to be extended with business software components. In defence systems the paradigm is shifting towards more software and component procurement from third parties. In both business and defence systems engineering, the selection of COTS software and software components which meet customer requirements and, at the same time, fit the system architecture and do not give rise to undesirable emergent behaviour is THE problem. The solution is often a trade-off between the satisfaction of stakeholder requirements and architecture requirements. The need to be able to make such trade-offs, and the theoretical and applied research needed to achieve these trade-offs, are the foci of this paper.

The shift towards software procurement and outsourcing means that prime contractors for software systems will need to have significant new capabilities to undertake tasks previously undertaken by customers. These tasks include evaluating and managing total system risks, analysing trade-offs between system performance, delivery deadlines and whole life cycle costs, and facilitating effective communication with stakeholders. However, prime contractors lack effective tools and techniques for these crucial tasks. For many prime contractors the software development process is too solution-driven and compartmentalised around software components and COTS software packages to be applied to a system which is itself composed of complex inter-operating sub-systems. A step increase is required in the capabilities to model all of the perspectives of a complex system and its production to handle increasing system complexity, value-based trade-offs, conformance assessment of requirements, and risk and uncertainty management. Prime contractors need new processes for developing complex systems with COTS software and software components, capabilities to detect the weak points in the processes and how those processes might be improved. It also requires new integrated system capabilities for determining system requirements, for evaluating and implementing design trade-offs, and for handling risks and uncertainties associated with such trade-offs using COTS software components. The selection, evaluation and integration of COTS software and software components is critical in making trade-off decisions to optimise the final software system, however most COTS/component-based software engineering research does not recognise this.

The next section describes a current method for COTS software selection and discusses its weaknesses with respect to the need to deliver complex systems composed of many different COTS software and software components. The remainder of the paper describes part of a research agenda to deliver the capabilities needed by prime contractors.

2. Current Methods for COTS Software Selection

There are few methods currently available to support the selection and integration of COTS software and software components that meet customer requirements. One exception that we will outline here is PORE. The PORE (Procurement-Oriented Requirements Engineering) method guides a software development team to acquire customer requirements and select COTS software systems which satisfy these requirements (Ncube 1999). It supports an iterative process of requirements acquisition and COTS software selection. During each iteration, the software development team:

(1) acquires customer requirements which discriminate between the COTS software candidates;

(2) undertakes multi-criteria decision-making to identify candidates that are not compliant with these requirements;
(3) rejects COTS software candidates that are non-compliant with the requirements;
(4) explores the remaining COTS software candidates to discover possible new customer requirements which might discriminate further between these candidates.

The customer requirements to acquire in each iteration change throughout the process as the decisions that the team has to make change. The team continues these iterations until, in an ideal situation, one or more candidate COTS software which comply with all essential customer requirements remain. Figure 1 depicts this iterative process.

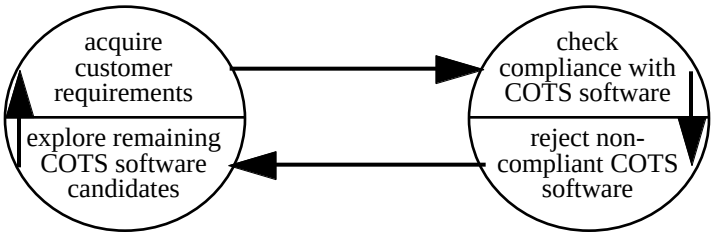

Figure 1. PORE's iterative process of requirements acquisition and COTS software selection.

PORE offers techniques to acquire customer requirements that enable the evaluation team to discriminate between COTS software candidates. PORE also offers techniques such as scenarios to discover, acquire and structure customer requirements to formulate test cases that are used to check COTS software compliance with the customer requirements. Other PORE techniques enable the team to discover and select cornerstone COTS software packages before peripheral packages which integrate with it. PORE also recommends that stakeholder representatives are present during COTS software demonstrations to ask important but unforeseen questions and to discover missing customer requirements. The complete description of the techniques offered in PORE is reported in Maiden & Ncube (1998).

Another feature of PORE is that it guides the software development team to make the decisions to select or reject COTS software candidates. Each iteration of the PORE process guides the team to acquire different customer requirements to reject COTS software that are not compliant with these requirements. The sequence of the decisions that PORE guides the team to make is shown in Figure 2. More simple decisions (e.g. to reject COTS software candidates that are not compliant with simple requirements such as cost) are made earlier in the process when there are more

candidates. More complex decisions (e.g. to reject software candidates for non-compliance with complex customer requirements such as system performance) are made later in the process when the COTS software is available to explore complex emergent system properties.

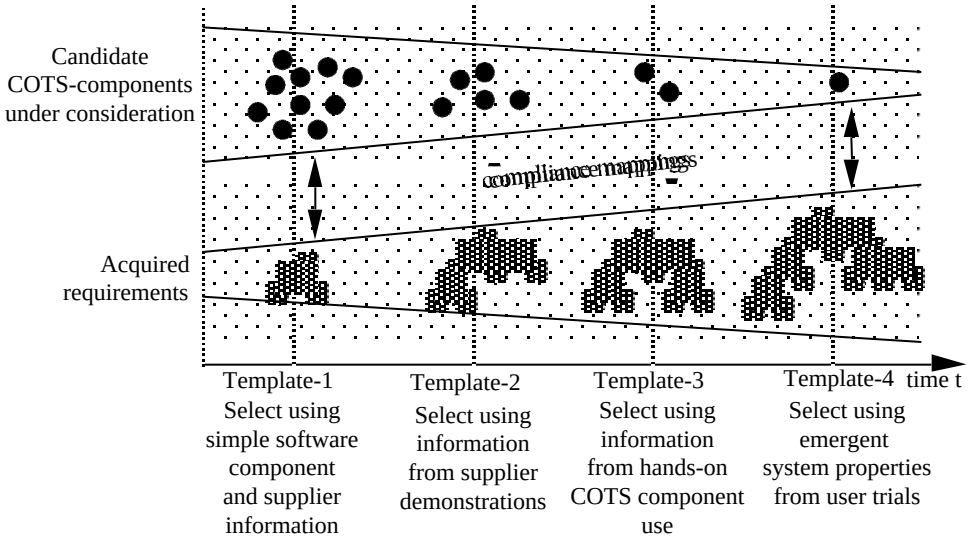

Figure 2: the sequence of critical decision-making stages to reject COTS software in PORE.

The PORE method has been evaluated on a number of case studies to be reported in Ncube & Maiden (2000), and has been shown to be useful and usable to customer organisations. However, the current limitations of PORE are clear. It only supports the selection of one COTS software package at a time, does not support integration of the COTS software and components to explore emergent system properties essential to test compliance with system requirements, and as a result does not scale to the types of systems which are commonplace in today's business and defence systems.

3. A Research Agenda for COTS-Based Systems Integration

Organisations need new methods, processes and modelling environments to undertake trade-off analyses that inform COTS and component selection. The envisaged research solution is large. The remainder of this paper will investigate some core intellectual problems which, when solved, can enable a wider solution. It will propose a new version of the PORE method extended with a modelling environment that will support trade-off analyses to enable COTS/component selection.

The new PORE method will have 2 extensions. First, it will have a modelling environment which will act as a 'test rig' to explore the impact of different COTS/component software on stakeholder and architectural requirements for a

system. Figure 3 depicts this ‘test rig’ environment. The vision is a “plug-and-play” environment in which different COTS/component combinations can be explored quickly to assess how system performance varies with these different combinations. The environment will use models of COTS/component software (possibly derived with reverse engineering techniques) rather than COTS software itself (Voas 1998). This has implication for libraries of COTS/component software. Figure 4 depicts this ‘plug-and-play’ environment. Second, the method will be extended with multi-criteria decision making techniques to enable software engineers to make trade-offs between the system performance and the cost and time to deliver this COTS/component combination. These trade-offs are a central component of systems engineering and systems integration. Let us explore these extensions in more detail.

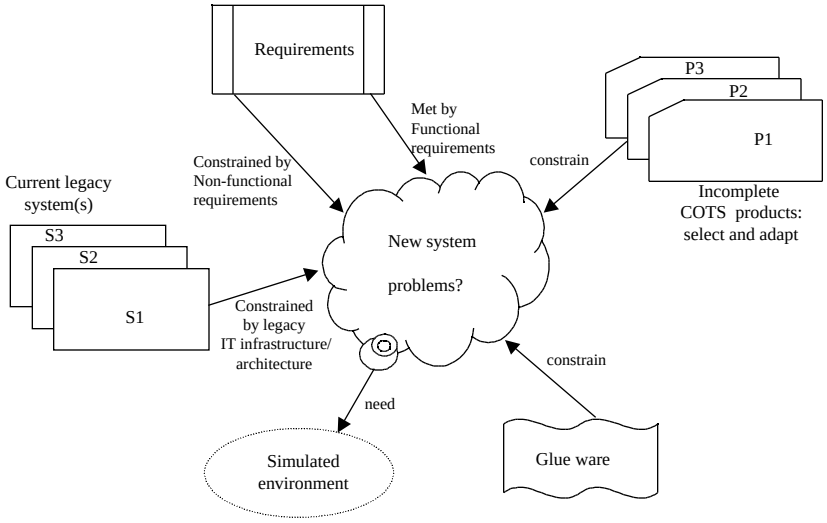

Figure 3: A modelling and simulation environment is needed in order to understand many competing, critical issues that arise during requirements engineering for COTS-based development. This environment will take as input customer requirements (both functional and non-functional), current legacy systems, candidate software products, and the glue ware software to explore the selection and integration of COTS products into their application environment. One critical success factor will be a plug-and-play environment as depicted in Figure 4 below.

The first extension is to categorise and model recurring problems which arise during COTS/component-based systems development as a basis for problem diagnosis in systems integration as depicted in Figure 4. There are 2 reasons for this. First, systems integration from COTS and component software is still a new area of research, and categorisation of systems integration problems and consequences will improve our understanding of detailed problems and hence drive an agenda for more specific

research opportunities. Second, this categorisation will provide a mechanism for diagnosing systems integration problems in COTS/component-based development.

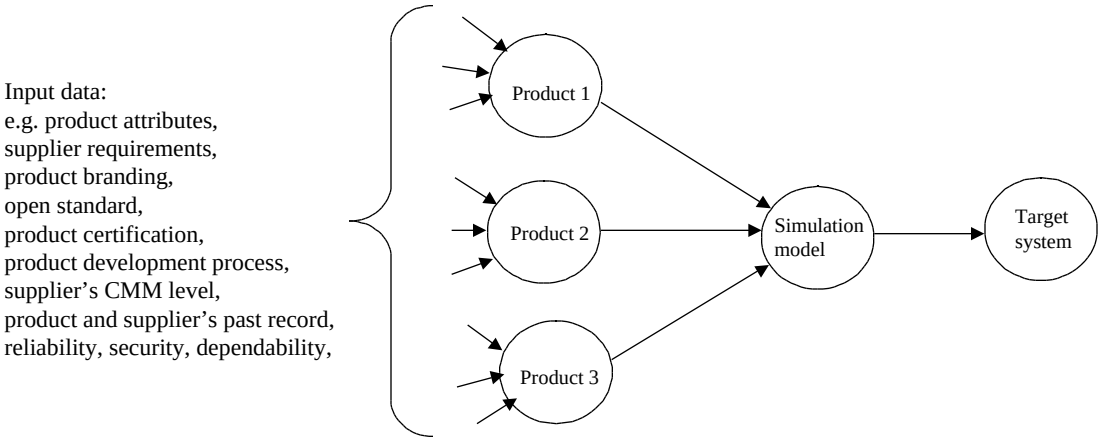

Figure 4: A modelling and simulation environment that allows plug-and-play of models of COTS software packages. Models of different COTS software products can be plugged in to explore their consequences. For example, this simulated environment would allow to evaluate and predict the reliability of COTS based systems and to build reliability models of these systems. Different configurations of COTS products can produce different reliabilities of the system and therefore, there is a need to choose the optimum configuration. The reliability prediction model can be based on or take as input, information about the products and their suppliers. This would allow to reason about uncertainty using diverse forms of evidence and then apply Bayes Theorem to predict reliability.

One advantage of a diagnostic approach is that it avoids the need for complete modelling of the COTS/component software, the integration of these components in a systems architecture, and system requirements which the system and its architecture must satisfy. Instead, the problem categorisation drives the diagnosis which, in turn, drives the essential modelling needed to make the diagnosis. The categorisation can be derived from both theoretical and empirical research. Relevant theories and research to draw on include systems theories, fault injection in software components (Voas 1998), models of software integration (Dean & Vidgar 1997) and computational viewpoints modelling (Spanoudakis & Finkelstein 1998, Kotonya & Sommerville 1997). The empirical research can use techniques which include laddering and card sorts (Maiden & Rugg 1996) to elicit categories of recurring problems from software engineering with experience of systems integration. The authors have considerable experience of this type of elicitation and modelling, for example the use of card sorts to elicit categories of problem domain from experienced

software engineers (Maiden & Hare 1998) and laddering to elicit categories of abnormal behaviour and state from naval warfare experts (Mavin 1999).

The second extension is to develop semantics for modelling COTS/component software and their properties which emerge from integration within an architecture. We propose a common object-oriented approach based on formal action semantics (Mellor & Wilkie 1999). COTS software and software components will be modelled with a restricted set of possible problem domain states derived from NATURE's domain models (Sutcliffe & Maiden 1998) and behaviours which indicate compliance to different types of non-functional requirement such as performance, reliability and usability. These restricted states and values are constrained using possible measurable fit criteria for system and architecture requirements. The essential test for the semantics is whether it is possible to determine the compliance of a system architecture and its COTS/component software to measurable system requirements. Therefore, semantics can be constrained by types of behaviours and states expressed in formal measurable fit criteria for each type of non-functional requirement.

One potential additional technique to apply to handle risk assessment associated with COTS software selection and integration is Bayesian Belief Networks (BBNs). We believe that BBNs offer an effective solution for providing quantitative risk assessment where there is a need to take into account of diverse, often uncertain, information. BBNs are graphs whose nodes represent uncertain variables, and whose arcs represent causal or influential relationships between variables. Associated with each node in a BBN is a probability table. This table provides the probability distribution for the node values conditional on each combination of parent values. BBNs can be used to model complex chains of reasoning with conditional dependencies. Research work [Fenton et al 1999] has demonstrated that engineers can benefit from BBNs without any need for specialist education in statistics and probability, provided that they are shielded from the underlying mechanics of Bayesian updating. We believe that BBNs can be applied to model candidate software system architectures. Different COTS software selections and integrations can be modelled as different instances of the BBNs, and changes to the architecture are modelled as simple changes to the BBN.

The third extension is to design and prototype a "test rig" modelling environment. The kernel of the hypothetical environment is a computational mechanism which enables modelling of system architectures with COTS/component software, diagnosis of problems in this architecture, inference of emergent behaviour and other properties of the architecture, and use of these emergent properties to test the architecture's compliance to measurable system requirements through scenarios. A possible architecture is shown in Figure 3.

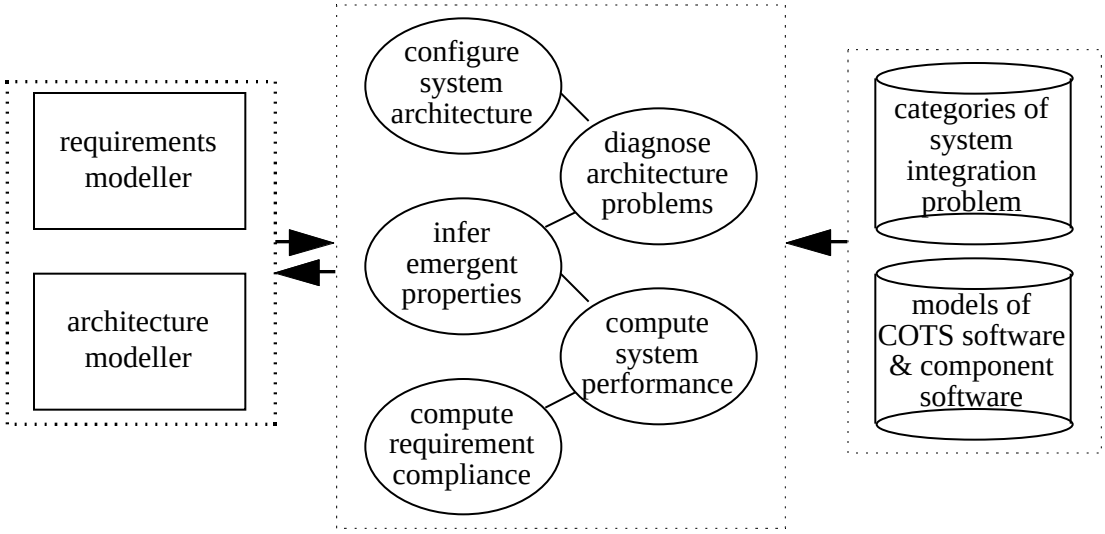

Figure 6. A possible prototype "test-rig" modelling architecture.

4. Conclusions

Admittedly this paper leaves a lot of open issues. It presents a state-of-the-art method for requirements engineering for COTS software selection, then critiques it and identifies possible new research directions and future solutions. As such we believe that it is a suitable paper to direct working group discussion. We look forward to discuss some of the issues that it raises in the workshop.

5. References

Dean C. J and Vigder R. M. 1997, System Implementation Using COTS Software, Proceedings of the 1997 Software Technology Conference (STC'97) Salt Lake City, May 1997.

Fenton N and Neil M, 1999: "A Critique of Software Defect Prediction Models" to appear in *IEEE Transactions Software Engineering*, 1999.

Kotonya G and Sommerville I, 1997, Integrating Safety Analysis and Requirements Engineering, Cooperative Systems Engineering Group, Lancaster University, Technical Report Ref: CSEG/18/1997.

Maiden N.A.M. & Hare M. (1998), Problem Domain Categories in Requirements Engineering, *International Journal of Human-Computer Studies*, 49, 281-304

Maiden N.A.M. & Ncube C., 1998, 'Acquiring Requirements for Commercial Off-The-Shelf Package Selection', IEEE Software, 15(2), 46-56.

Maiden N.A.M. & Rugg G., 1996, 'ACRE: Selecting Methods For Requirements Acquisition, Software Engineering Journal 11(3), 183-192.

Mavin A, 1999, A Method and Software Tool for Operational Scenario Analysis at Marconi Naval Systems, June 1999, City University

Mellor S and Wilkie I, 1999, Precise Action Semantics for UML, Proceedings of the 6th Annual Shlaer-Mellor User Group Conference, 14th – 15th September 1999, Carey's Manor Hotel, Brockenhurst.

Ncube C 1999, A Requirements Engineering Method for COTS-Based Systems Development, PhD Thesis, City University, December 1999.

Ncube C and Maiden N A. M. 2000, Selecting the Right COTS Software: Why Requirements are Important, to be published as a chapter in a book entitled: Component-Based Software Engineering: Putting the Pieces Together, an Addison-Wesley Longman Publication, summer 2000.

Spanoudakis G. and Finkelstein A. 1998, *A Semi-automatic process of Identifying Overlaps and Inconsistencies between Requirement Specifications*, In Proceedings of the 5th International Conference on Object-Oriented Information Systems (OOIS 98), pp. 405-424

Sutcliffe A.G. & Maiden N.A.M. (1998), The Domain Theory for Requirements Engineering, *IEEE Transactions on Software Engineering*, 24(3), 174-196.

Voas J. 1998, Maintaining Component-Based Systems, IEEE Software, vol. 15, no 14, pp 22-27, July/August 1998 Issue.