COTS Integration and Evaluation: Introduction

Robert C. Seacord
Grace A. Lewis
Russ Bunting
Software Engineering Institute
Carnegie Mellon
4500 Fifth Avenue, Pittsburgh, PA, 15213

1 BACKGROUND

Enterprise systems, as well as other types of software systems, are increasingly being developed largely from COTS (commercial-off-the-shelf) components. While a COTS-based systems approach is often considered essential to building highly functional, competitive systems this approach also poses many challenges. Foremost among these is the identification of compatible components that meet the functional requirements of the system under development. Compatibility in this sense is a measure of how easily various components can be integrated. A COBOL module running on a Unisys platform would not be considered highly compatible with a VisualBasic application, for example.

2 ENSEMBLE SELECTION

Currently, identifying and evaluating commercial components is largely a manual task. System integrators must first perform a market survey to identify COTS components that meet the functional requirements. For example, a system may have requirements for three components: c1, c2, and c3, as shown in Figure 1. The first component (c1) is an XML database. The second component (c2) is a rules engine and the third component (c3) is an XML parser. Some number of candidate components may now be identified that provide similar functionality. For example, components x1, x2 and x3 may all satisfy the functional requirements of component c1, component y1 may satisfy the functional requirements of c2 and z1, z2 and z3 may satisfy the functional requirements of c3. The system integrator must discover an ensemble—a collection of compatible components—that satisfies the functional requirements.

Evaluation of the individual software components does not guarantee that the selected components are mutually compatible. This makes it necessary to evaluate ensembles of components. When multiple components are found that match the requirements, however, the number of possible ensembles becomes exponential. This situation becomes even worse when you consider that there may be multiple versions of each component.

Evaluating all possible component ensembles by hand is, of course, prohibitively expensive. In practice, system integrators must rely on experience to select components with a high degree of compatibility. The result is that the component space is largely unexplored, and the possibility that an optimal component ensemble has been selected is low.

3 AIM OF THIS WORKSHOP

The goal of this workshop was to identify an approach for automating this process. Workshop participants worked on a reference architecture for a system to perform the search and evaluation functionality. The envisioned system consists of a searchable repository of components specifications, including attributes relevant to integration issues such as the available language bindings, supported protocols, and platform requirements. Users of the system provide a Manifest that defines the functional requirements of the components and interactions among components, as well as any overriding constraints on how the components will be integrated (i.e., they all must provide a Java language binding). Components that match the requirements specified in the Manifest can then be evaluated for compatibility based on a repository of software engineering integration rules. Product ensembles, ranked according to compatibility, are returned to the user for further evaluation.

It is hoped that by automating part of the process by which components are discovered and evaluated some savings in evaluation and development costs can be obtained, and that more components can be considered more quickly than might have otherwise been possible (leading to the selection of best-fit components that may not have been previously considered.

Component brokers, who market third-party components, may initially stand up the component search engine. Component consumers will use this service to discover components that can be easily integrated into their systems and begin to rely on these component brokers. Component producers may then feel compelled to generate component specifications that are compatible with this approach. Ideally this will result in a snowball effect that will result in a standard component repository and a standard component specification format.