The Future of Software Integration: Self-integrating Systems

Steven R. Ray, Ph.D.
Chief, Manufacturing Systems Integration Division
National Institute of Standards & Technology
100 Bureau Drive, STOP 8260
Gaithersburg MD, USA, 20899-8260
Tel: (301) 975-3508; Fax (301) 258-9749
E-mail: steven.ray@nist.gov

Abstract

This paper outlines a visionary research program for systems integration that will improve the reliability and robustness of complex "systems of systems." This program, self-integrating systems, will significantly advance the science underlying computer communication and the engineering of software for network-centric systems. Key concepts representative of the notion of self-integrating systems include semantic interoperability, knowledge representation, information theory, and coordination models.

Problem Statement

The need to integrate large software systems into complex "systems of systems" is evident in multiple problem domains. Yet industry testimonials describe multi-million-dollar expenditures for unsuccessful software systems integration efforts. Despite the media promotion surrounding internet connectivity, middleware technologies, and standards like eXtensible Markup Language (XML), the difficulties of integrating large systems have likely increased given the heightened expectations for new capabilities in a world of ever-expanding technology choices. The current approaches to software integration are costly, time consuming, and frequently yield suboptimal results. The characteristics of current system integrations include:

- They are brittle, i.e., easily fail when faced with slight perturbations to the information transacted.
- They are difficult to maintain as the systems involved are upgraded.
- They are difficult to scale when the requirements for additional information content or additional constituent systems arise.

Current Solutions

Information has been flowing between software systems for decades; albeit with ever-increasing costs. When information transfers successfully without human intervention it is symptomatic of software integration. What systems integrators do is to try to reduce overhead costs and effort by making the information flow repeatable, with greater confidence of accuracy, and within the timeframe of need. Successful systems integration depends on whether:

- The hardware is compatible
- There is agreement about the communication protocols to be used
- There is understanding about how to transfer information among the information architectures involved.
- The software interrelationships are well defined
- The information at the interface between systems is in an open format, or mapped to adequate neutral-format standards
- The purpose of the communication is mutually understood.
- The terminology used is unambiguous in meaning

There are other considerations as well; business rules must be compatible, government regulations for commerce must be agreeable, the stakeholders representing the systems involved must truly want to share information, and so on.

Self-Integrating Systems

To introduce the concept of self-integrating systems, consider the following scenario: software system A is deployed on the internet with the requirement to interoperate with system B[1]. Focusing on semantic compatibility, system A sends an initial query message to B identifying the preferred communication and content languages as well as the adopted semantic definition for the languages. The languages and semantic definition are accessible in the environment. Systems A and B interact to negotiate appropriate common languages and semantic definitions. During the negotiations, the two systems adjust, adapt, and/or select appropriate parameters or components to establish semantic interoperability.

Building self-integrating systems is an ambitious software endeavor, similar in complexity to other visionary software pursuits. New scientific approaches will be needed to develop new theories of integrated system development, experimental apparatus (e.g. simulation and analytical experiments), and assessment approaches for the theories within the experimental domain. Multi-disciplinary engineering approaches will be needed to develop self-integrating systems, to test theoretical approaches, and to assess the scalability, efficiency, and effectiveness of these new approaches. To arrive at such engineering approaches, it will be necessary to have the following:

- Methods for clear specification and communication of integration problems
- An effective means to accumulate integration-design knowledge, i.e. lessons learned
- A set of metrics supported by a theoretical foundation for gauging semantic equivalence
- Knowledge-based models that are expressive (both domain-specific and domain-independent), formal, and easy-to-use to capture semantics of different domains. A domain-specific approach to self-integrating systems may be helpful in managing the complexities of the issue
- Intention, goal, plan, beliefs, and experience models on top of knowledge-based models to share domain-specific concepts
- Machine learning and automated adaptation methods to provide for an efficient means of self-integration.
- Coordination/Collaboration/Negotiation theories and technologies that allow for flexible integration of systems at the level of task and process definitions
- Smart human-computer interfaces to enable development of usable, semantic-based, descriptions as a basis for developing ontologies.

Conclusions

In the face of the increasing pace of business and the increasing reliance on network-centric systems, we have little choice but to move forward with new technology approaches that can accommodate this shifting landscape. Practically speaking, the software development community will have to change its practices because new requirements and growth of complexity are outpacing the ability to keep up. It will not be practical to convene a team of all relevant players each time a new distributed application is designed. We need to endow software development systems and the resulting applications with the ability to solve many of these systems integration problems autonomously.

[1] While the integration architecture may depend on the specific situation (i.e., peer-to-peer, mediator-based, client-server, component-framework), we will assume the general case of peer-to-peer integration.